MICROS Y LABORATORIOS

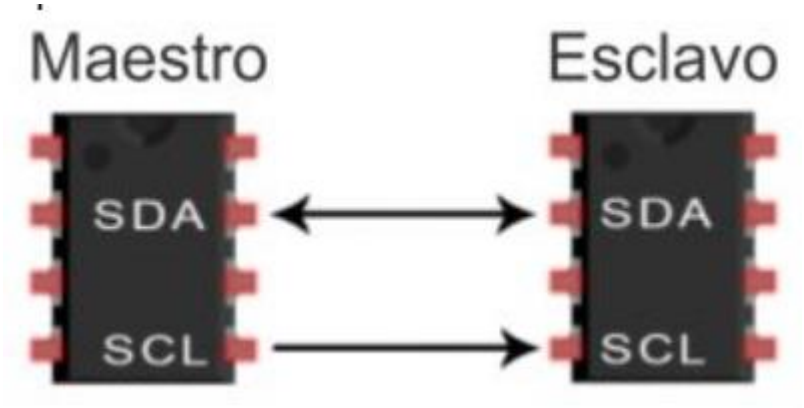
SESIÓN 30. I2C

PROTOCOLO I2C

- I2C significa Circuito Interintegrado (Por sus siglas en Inglés Inter-Integrated Circuit)
- Es un protocolo de comunicación serial desarrollado por Phillips Semiconductors en la década de los 80s. Básicamente se creó para poder comunicar varios chips al mismo tiempo dentro de los televisores.
- El protocolo I2C toma e integra lo mejor de los protocolos SPI y UART.
- Con el protocolo I2C se pueden tener a varios maestros controlando uno o múltiples esclavos. Esto puede ser de gran ayuda cuando se van a utilizar varios microcontroladores para almacenar un registro de datos hacia una sola memoria o cuando se va a mostrar información en una sola pantalla.

PROTOCOLO I2C

El protocolo I2C utiliza sólo dos vías o cables de comunicación, así como también lo hace el protocolo UART.



- **SDA – Serial Data:** Es la vía de comunicación entre el maestro y el esclavo para enviarse información.
- **SCL – Serial Clock:** Es la vía por donde viaja la señal de reloj.

- ✓ I2C es un protocolo de comunicación serial.
- ✓ El protocolo I2C envía información a través de una sola vía de comunicación. La información es enviada bit por bit de forma coordinada.
- ✓ I2C es un protocolo síncrono.
- ✓ Al igual el protocolo SPI, el protocolo I2C trabaja de forma síncrona. Esto quiere decir que el envío de bits por la vía de comunicación SDA está sincronizado por una señal de reloj que comparten tanto el maestro como el esclavo a través de la vía SCL.

PROTOCOLO I2C

- **Condición de Inicio – Start:** La vía SDA cambia de un nivel de voltaje Alto a un nivel de voltaje Bajo, antes de que el canal SCL cambie de Alto a nivel Bajo.
- **Condición de Paro – Stop:** La vía SDA ahora cambia de un nivel de voltaje Bajo a Alto, después de que la vía SCL cambia de Bajo a Alto.
- **Trama de Dirección – Addres Frame:** Es una secuencia única que va de los 7 a los 10 bits. Este sección (Frame) se envía a cada Esclavo, y va a identificar al Esclavo con el que el Maestro se quiere comunicar.

PROTOCOLO I2C

- **Bit para Lectura/Escritura A – Read/Write Bit A:** Es un bit de información enviado a los Esclavos. Por medio de este bit el Maestro indica si le va enviar información al Esclavo (Nivel Bajo de voltaje = Escritura), o si el Maestro quiere solicitarle información al Esclavo (Nivel Alto de Voltaje = Lectura).

- **Bit ACK/NACK :** Después de cada sección (Frame) de información enviada en un mensaje, hay un bit acknowledge/no-acknowledge (reconocido/no-reconocido). Esto ayuda a identificar si la información fue enviada correctamente. En seguida de que se envía un Frame, si este fue recibido con éxito, se retorna un bit ACK al remitente. Si la información no fue recibida con éxito, se retorna un bit NACK.

EJEMPLOS / EJERCICIOS

EJEMPLO: Diseñar un programa que permita enviar y recibir un dato a través del modulo I2C.

SOLUCIÓN:

- **Puertos y pines necesarios:**

- **MODULO I2C = Modulo 2**
  - **PF1->SCL (reloj)**
  - **PF0->SDA (datos)**
  - **Interrupciones deshabilitadas**

```
RCC->AHB1ENR  |=  0x00000020; // Encender reloj puertoF
```

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

```
GPIOF->AFR[0] |=  0x00000044; // seleccion de la funcion alterna 4 del puerto F(I2C) para  PF1-SCL,PF0-SDA -> I2C2
```

Table 12. STM32F745xx and STM32F746xx alternate function mapping (continued)

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
		SYS	TIM1/2	TIM3/4/5	TIM8/9/10/11/LPTIM1/CEC	I2C1/2/3/4/CEC	SPI1/2/3/4/5/6	SPI3/SAI1	SPI2/3/USART1/2/3/UART5/SPDIFRX	SAI2/USART6/UART4/5/7/8/SPDIFRX	CAN1/2/TIM12/13/14/QUADSPI/LCD	SAI2/QUADSPI/OTG2_HS/OTG1_FS	ETH/OTG1_FS	FMC/SDMMC1/OTG2_FS	DCMI	LCD	SYS
Port E	PE14	-	TIM1_CH4	-	-	-	SPI4_MOSI	-	-	-	-	SAI2_MCK_B	-	FMC_D11	-	LCD_CLK	EVENTOUT
	PE15	-	TIM1_BKIN	-	-	-	-	-	-	-	-	-	-	FMC_D12	-	LCD_R7	EVENTOUT
	PF0	-	-	-	-	I2C2_SDA	-	-	-	-	-	-	-	FMC_A0	-	-	EVENTOUT
	PF1	-	-	-	-	I2C2_SCL	-	-	-	-	-	-	-	FMC_A1	-	-	EVENTOUT
	PF2	-	-	-	-	I2C2_SMBA	-	-	-	-	-	-	-	FMC_A2	-	-	EVENTOUT
	PF3	-	-	-	-	-	-	-	-	-	-	-	-	FMC_A3	-	-	EVENTOUT

```
GPIOF->MODER   |=   0x0000000A; // PF0,PF1 => en modo alterno
```

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

```
GPIOF->OTYPER |=  0x0003;      // Open drain
```

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

```
GPIOF->PUPDR |=  0x5;
```

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

```
GPIOF->OSPEEDR   |=  0xC;
```

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

```
RCC->APB1ENR   |= 0x00400000;        // Enable clock for I2C2 bit 22
```

5.3.13 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

Bit 22 **I2C2EN:** I2C2 clock enable

This bit is set and cleared by software.

0: I2C2 clock disabled

1: I2C2 clock enabled

```
RCC->DCKCFGR2   |= 0x80000;                    //Reloj de frecuencia del I2C2
```

5.3.26 RCC dedicated clocks configuration register (DCKCFGR2)

Address: 0x90h

Reset value: 0x0000 0000h

Access: no wait state, word, half-word and byte access

This register allows to select the source clock for the 48MHz, SDMMC, HDMI-CEC, LPTIM1, UARTs, USARTs and I2Cs clocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SDMMC1SEL	CK48MSEL	CECSEL	LPTIM1SEL		I2C4SEL		I2C3SEL		I2C2SEL		I2C1SEL	
			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
UART8SEL		UART7SEL		USART6SEL		UART5SEL		UART4SEL		UART3SEL		UART2SEL		UART1SEL	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 19:18 **I2C2SEL**: I2C2 clock source selection

Set and reset by software.

00: APB1 clock (PCLK1) is selected as I2C2 clock

01: System clock is selected as I2C2 clock

10: HSI clock is selected as I2C2 clock

11: reserved

```
// Disable the I2Cx peripheral
I2C2->CR1 &= ~I2C_CR1_PE; //deshabilita SCL y SDA para el periferico
while (I2C2->CR1 & I2C_CR1_PE);
```

30.7.1 Control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	Res.	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

Bit 0 **PE**: Peripheral enable

0: Peripheral disable

1: Peripheral enable

Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window

Figure 288. I2C initialization flowchart

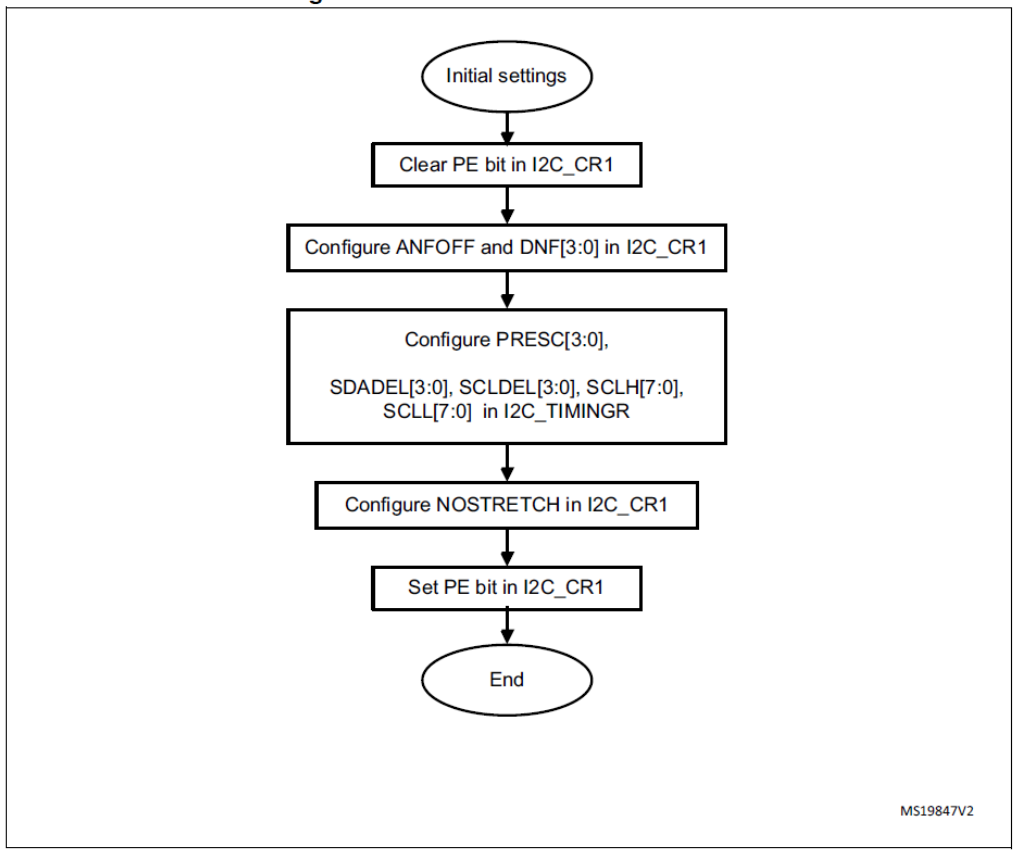
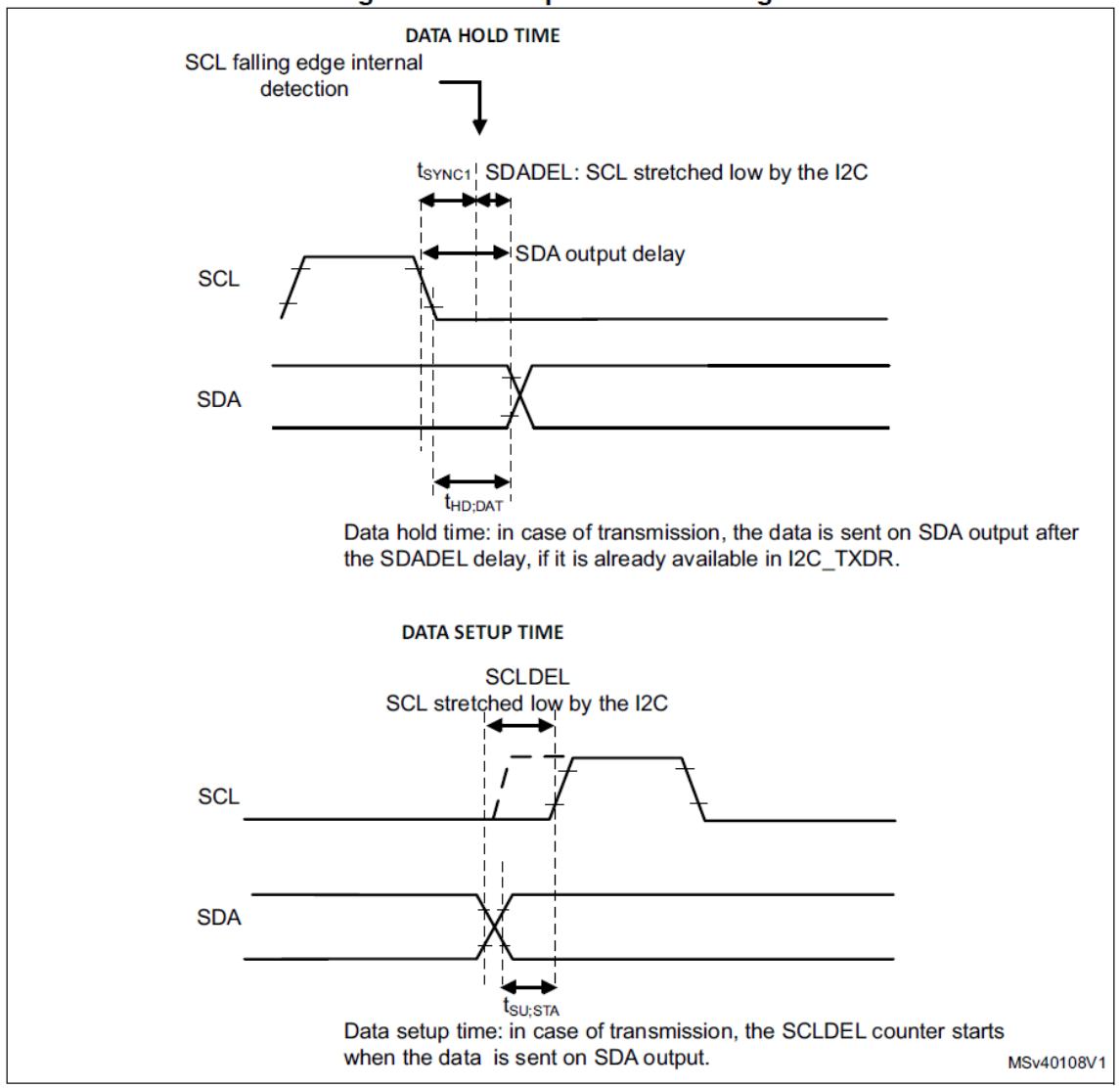


Figure 287. Setup and hold timings



```
I2C2->TIMINGR |= 0x30420F13;          //I2C2 Timing config
```

```
11000001000010000011110010011
```

30.7.5 Timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

PRESC = 3
SCLDEL = 4
SDADEL = 2
SCLH = 15
SCLL=19

11				0000				0100				0010			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw								rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw								rw							
00001111								00010011							

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 923*) and for SCL high and low level counters (refer to *I2C master initialization on page 938*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

$$t_{PRESC} = (PRESC + 1) * t_{I2CCLK}$$
$$t_{PRESC} = (3 + 1) * 62{,}5ns$$
$$t_{PRESC} = 250ns$$

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

$$t_{SCLDEL} = (SCLDEL + 1) * t_{PRESC}$$
$$t_{SCLDEL} = (4 + 1) * 250ns$$
$$t_{SCLDEL} = 1{,}25us$$

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

$$t_{SDADEL} = (SDADEL) * t_{PRESC}$$
$$t_{SDADEL} = (2) * 250ns$$
$$t_{SDADEL} = 500ns$$

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

$$t_{SCLH} = (SCLH + 1) * t_{PRESC}$$
$$t_{SCLH} = (15 + 1) * 250ns$$
$$t_{SCLH} = 4us$$

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

$$t_{SCLL} = (SCLH + 1) * t_{PRESC}$$
$$t_{SCLL} = (19 + 1) * 250ns$$
$$t_{SCLL} = 5us$$

PRESC = 3
SCLDEL = 4
SDADEL = 2
SCLH = 15
SCLL=19

I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window

Table 161. Examples of timings settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL} (1)	~100 µs (2)	~10 µs (2)	~2500 ns (3)	~1000 ns (4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

```
I2C2->CR2 |= 0x0202207E;
```

```
10000000100010000001111110
```

30.7.2 Control register 2 (I2C_CR2)

NBYTES = 2

SADD = 0X7E -> Solo es valido para este ejemplo

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

0 1 0 00000010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_ WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									

0 0 1 0 0 0 0001111110

Bits 7:1 **SADD[7:1]**: Slave address bit 7:1 (master mode)

In 7-bit addressing mode (ADD10 = 0):

These bits should be written with the 7-bit slave address to be sent

In 10-bit addressing mode (ADD10 = 1):

These bits should be written with bits 7:1 of the slave address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

Bit 0 **SADD0** : Slave address bit 0 (master mode)

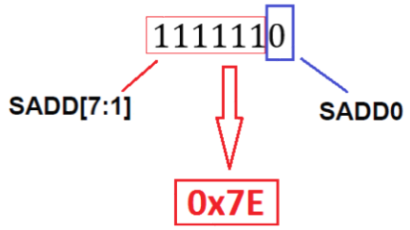
In 7-bit addressing mode (ADD10 = 0):

This bit is don't care

In 10-bit addressing mode (ADD10 = 1):

This bit should be written with bit 0 of the slave address to be sent

Note: Changing these bits when the START bit is set is not allowed.



PCF8574A

0	1	1	1	A2	A1	A0

$$Si\ A2, A1\ y\ A0 = 1, entonces$$
$$la\ dirección\ del\ esclavo\ es = 0111111 = 0x3F$$

NBYTES = 2

SADD = 0X7E -> Solo es valido para este ejemplo

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCF bit in the I2C_ICR register.

0: No Start generation.

1: Restart/Start generation:

– If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.

– Otherwise setting this bit will generate a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set. In 10-bit addressing mode, if a NACK is received on the first part of the address, the START bit is not cleared by hardware and the master will resend the address sequence, unless the START bit is cleared by software

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in slave mode or when the RELOAD bit is set.

```
// Use 7-bit addresses
I2C2->CR2 &=~ I2C_CR2_ADD10; //Modo maestro receptor, solo recibe 7 de 10 bits de comunicación
// Enable auto-end mode
I2C2->CR2 |= I2C_CR2_AUTOEND; //autofinalización activada
```

30.7.2 Control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_ WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									

Bit 11 **ADD10**: 10-bit addressing mode (master mode)

0: The master operates in 7-bit addressing mode,

1: The master operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

```
// Disable the analog filter
I2C2->CR1 |= I2C_CR1_ANFOFF; //filtro de ruido desactivado
// Disable NOSTRETCH
I2C2->CR1 |= I2C_CR1_NOSTRETCH; //nostretch desactivado
```

30.7.1 Control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	Res.	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.

0: Clock stretching enabled

1: Clock stretching disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

Bit 12 **ANFOFF:** Analog noise filter OFF

0: Analog noise filter enabled

1: Analog noise filter disabled

Note: This bit can only be programmed when the I2C is disabled (PE = 0).

```
// Enable the I2Cx peripheral
I2C2->CR1 |= I2C_CR1_PE; //habilita SCL y SDA para el periferico
```

30.7.1 Control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBD EN	SMBH EN	GCEN	Res.	NOSTR ETCH	SBC
								rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

Bit 0 **PE**: Peripheral enable

0: Peripheral disable

1: Peripheral enable

Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

Función de inicialización del modulo I2C:

```
void I2C2_Init (void)   {
    RCC->AHB1ENR  |=  0x00000020; // Encender reloj puertoF
    GPIOF->AFR[0] |=  0x00000044; // seleccion de la funcion alterna 4 del puerto F(I2C) para  PF1-SCL,PF0-SDA -> I2C2
    GPIOF->MODER  |=  0x0000000A; // PF0,PF1 => en modo alterno
    GPIOF->OTYPER |=  0x0003;     // Open drain
    GPIOF->PUPDR |=  0x5;
    GPIOF->OSPEEDR   |=  0xC;

    RCC->APB1ENR  |= 0x00400000;       // Enable clock for I2C2 bit 22
    RCC->DCKCFGR2  |= 0x80000;                //Reloj de frecuencia del I2C2

    // Disable the I2Cx peripheral
    I2C2->CR1 &= ~I2C_CR1_PE; //deshabilita SCL y SDA para el periferico
    while (I2C2->CR1 & I2C_CR1_PE);

    //I2C2->TIMINGR
    I2C2->TIMINGR |= 0x30420F13;         //I2C2 Timing config at t2clk=1/FHSI

    I2C2->CR2 |= 0x0202207E;

    // Use 7-bit addresses
    I2C2->CR2 &=~ I2C_CR2_ADD10; //Modo maestro receptor, solo recibe 7 de 10 bits de comunicación
    // Enable auto-end mode
    I2C2->CR2 |= I2C_CR2_AUTOEND; //autofinalización activada
    // Disable the analog filter
    I2C2->CR1 |= I2C_CR1_ANFOFF; //filtro de ruido desactivado
    // Disable NOSTRETCH
    I2C2->CR1 |= I2C_CR1_NOSTRETCH; //nostretch desactivado

    // Enable the I2Cx peripheral
    I2C2->CR1 |= I2C_CR1_PE; //habilita SCL y SDA para el periferico
}
```

Función para enviar un dato por el modulo de I2C:

```
void I2C2_Write (char Dat){
//I2C2 Initialization
    I2C2->CR2 |= (1UL << 13);  //bit de START  bit 13
    while (!(I2C2->ISR & I2C_ISR_TXIS));
    I2C2->TXDR = LCD_I2C_Address;
    while (!(I2C2->ISR & I2C_ISR_TXIS));
    I2C2->TXDR = Dat;
    while (!(I2C2->ISR & I2C_ISR_TXE));
    while (!(I2C2->ISR & I2C_ISR_STOPF));
    I2C2->ISR &= ~I2C_ISR_STOPF;
}
```

30.7.7 Interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r							r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).

Note: This bit is cleared by hardware when PE=0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE=0.

Bit 5 **STOPF**: Stop detection flag

This flag is set by hardware when a Stop condition is detected on the bus and the peripheral is involved in this transfer:

– either as a master, provided that the STOP condition is generated by the peripheral.

– or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE=0.

Función para recibir un dato por el modulo de I2C:

```
//FUNCION PARA RECIBIR
char I2C2_Read (char Adr){

  I2C2->CR2 |= (1UL<<13);    //bit de Start
    while (!(I2C2->ISR & I2C_ISR_TXIS));
    I2C2->TXDR = Adr;
    while (!(I2C2->ISR & I2C_ISR_TXE));
    while (!(I2C2->ISR & I2C_ISR_STOPF));
    I2C2->ISR &= ~I2C_ISR_STOPF;
    I2C2->CR2=0;

    return (Dato);
}
```

Por medio de la tabla de registros que se encuentra en la hoja de datos de cada sensor, solicitar la información necesaria ingresado el valor en el argumento de entrada de la función.

Programa Principal: Ejemplo para configurar una LCD controlada por un modulo conversor I2C

```
int main ()  {

    I2C2_Init();          // initializar  I2C2

    LCD_I2C_Ini();        // inicializar LCD

    LCD_I2C_WriteData('A');  //Enviar letra de prueba
  while(1) {

  }//cierra while
}//cierra maina
```

Preguntas