MICROS 32 BITS
STM – I2C

ROBINSON JIMENEZ MORENO

LUISA FERNANDA GARCIA VARGAS

UNIVERSIDAD MILITAR NUEVA GRANADA

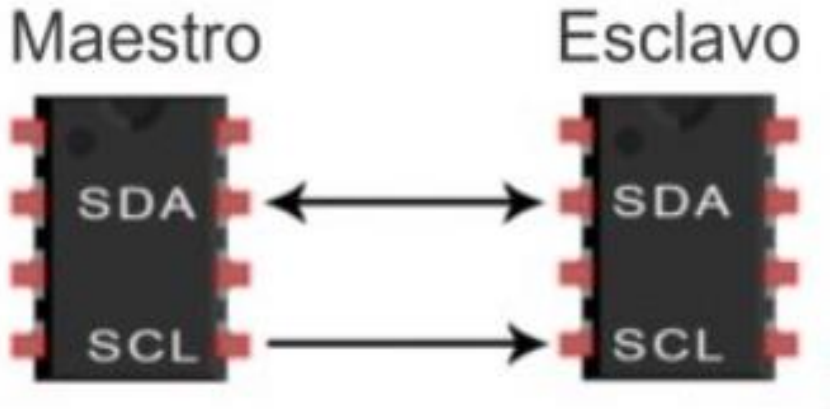
INGENIERÍA MECATRÓNICA UMNG

https://www.i2c-bus.org/

I2C o Circuito Interintegrado (Inter-Integrated Circuit) es un protocolo de comunicación serial desarrollado por Phillips Semiconductors en la década de los 80s. Se creó para poder comunicar varios chips al mismo tiempo dentro de los televisores.

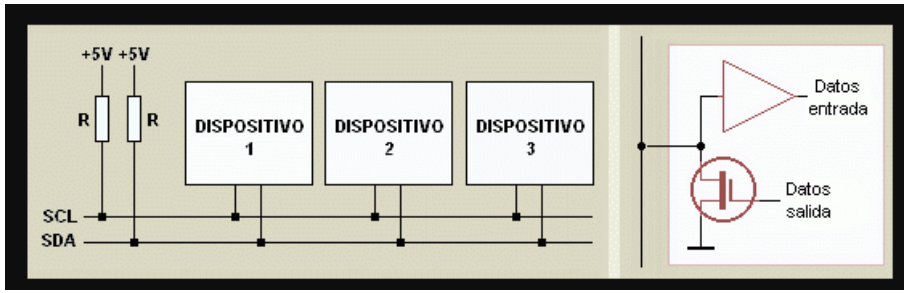
Con el protocolo I2C es posible tener a varios maestros controlando uno o múltiples esclavos. Esto puede ser de gran ayuda cuando se van a utilizar varios microcontroladores para almacenar un registro de datos hacia una sola memoria o cuando se va a mostrar información en una sola pantalla.

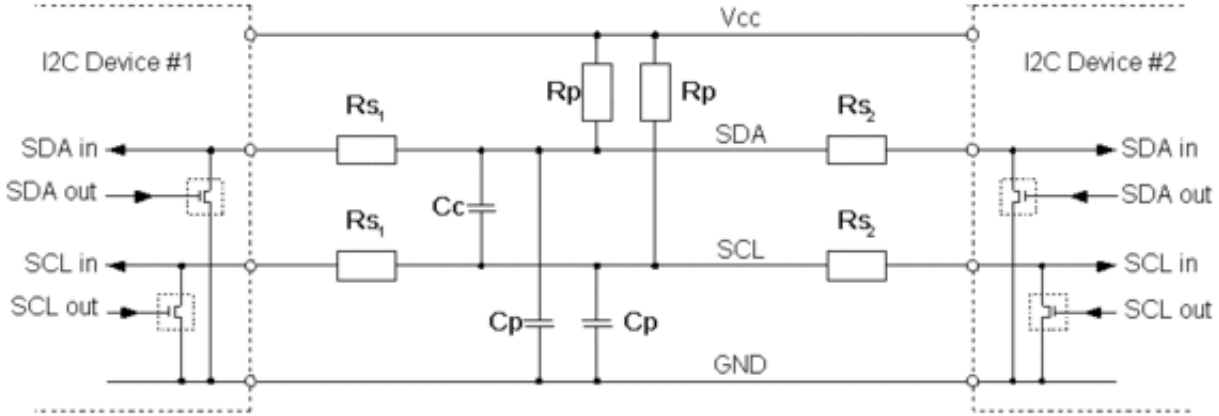
El protocolo I2C utiliza sólo dos vías o cables de comunicación, así como también lo hace el protocolo UART.



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.

https://teslabem.com/nivel-intermedio/fundamentos/





The image shows a simplified equivalent circuit diagram for an I2C connection between two devices (master or slave) containing all relevant factors for I2C.

VCC	I2C supply voltage, typically ranging from 1.2 V to 5.5 V
GND	Common ground
SDA	Serial data (I2C data line)
SCL	Serial clock (I2C clock line)
Rp	Pull-up resistance (a.k.a. I2C termination)
Rs	Serial resistance
Cp	Wire capacitance
Cc	Cross channel capacitance

https://www.i2c-bus.org/

http://robots-argentina.com.ar/didactica/wp-content/uploads/Comunicacion_busI2Cblk.gif

I2C es un protocolo de comunicación serial: envía información a través de una sola vía de comunicación. La información es enviada bit por bit de forma coordinada. **Trabaja de forma síncrona,** 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.

Velocidad máxima	Modo estándar (Sm) = 100kbps
	Modo rápido (Fm) = 400kbps
	Modo High Speed (Fm+) = 3.4Mbps
	Modo Ultra Fast (Hs-mode) = 5Mbps

La información viaja en mensajes divididos en tramas de datos. Cada mensaje lleva una trama con una dirección la cuál transporta la dirección binaria del esclavo al que va dirigido el mensaje, y una o más tramas que llevan la información del mensaje. También el mensaje contiene condiciones de inicio y paro, lectura y escritura de bits, y los bits ACK y NACK.

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.

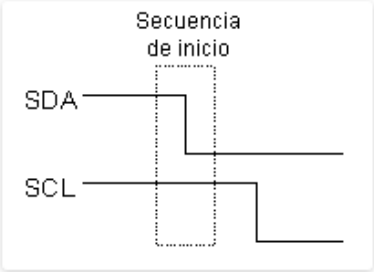
Condición de Paro – Stop: La vía SDA ahora cambia de un nivel de voltaje Bajo a Alto, después de que la vía SCL cambia de Bajo a Alto.

Trama de Dirección – Addres Frame: Es una secuencia única que va de los 7 a los 10 bits. Esta sección (Frame) se envía a cada Esclavo, y va a identificar al Esclavo con el que el Maestro se quiere comunicar.

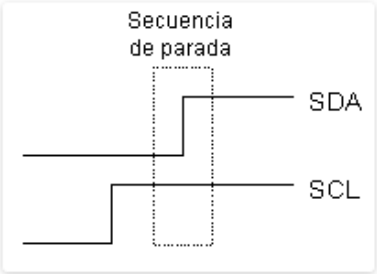
Bit para Lectura/Escritura A – Read/Write Bit A: Es un bit de información enviado a los Esclavos. Por medio de este bit el Maestro indica si le va a enviar información al Esclavo (Nivel Bajo de voltaje = Escritura), o si el Maestro quiere solicitarle información al Esclavo (Nivel Alto de Voltaje = Lectura).

Bit ACK/NACK – : Después de cada sección (Frame) de información enviada en un mensaje, podemos notar que lleva 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.

https://teslabem.com/nivel-intermedio/fundamentos/



La condición inicial, de bus libre, es cuando ambas señales están en estado lógico alto. En este estado cualquier dispositivo maestro puede ocuparlo, estableciendo la condición de inicio (start). Esta condición se presenta cuando un dispositivo maestro pone en estado bajo la línea de datos (**SDA**), pero dejando en alto la línea de reloj (**SCL**).

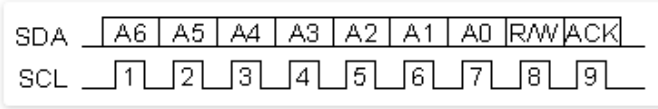


En el caso contrario, cuando el bit de lectura/escritura estaba a nivel lógico alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar los datos. Luego de cada byte recibido el dispositivo maestro (quien está recibiendo los datos) genera un pulso de reconocimiento.

El dispositivo maestro puede dejar libre el bus generando una condición de parada (o detención; stop en inglés).

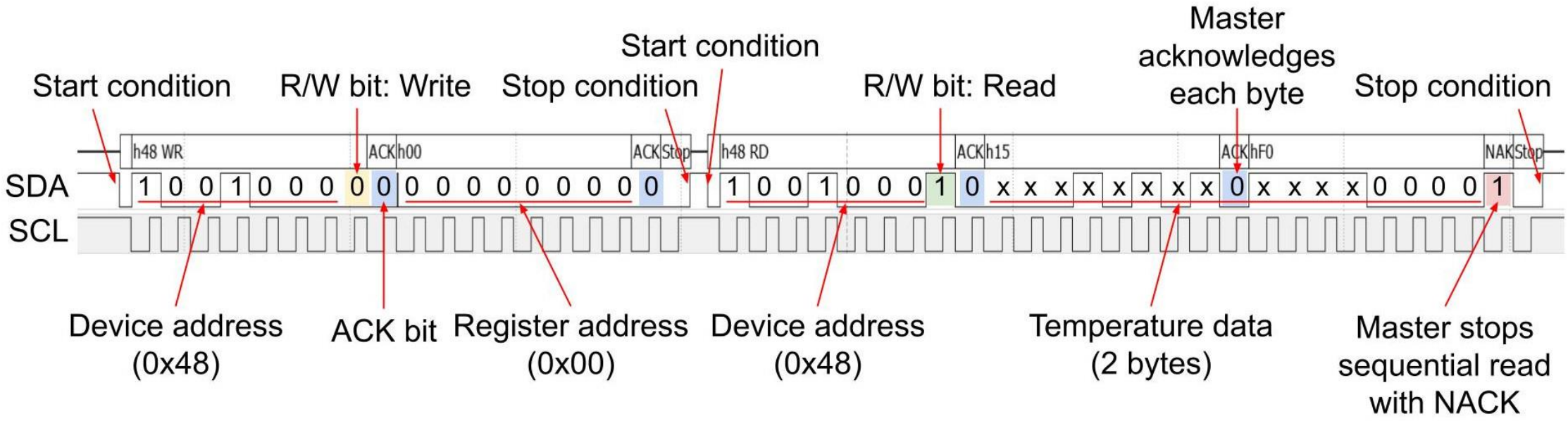
El primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (A0-A6) está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de reconocimiento (ACK) en bajo le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos.

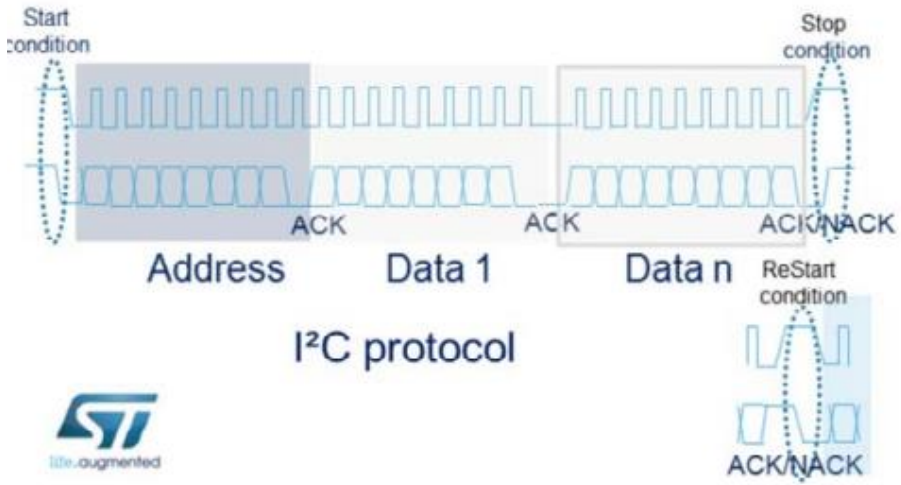


Si el bit de lectura/escritura (R/W) fue puesto en esta comunicación a nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo. Esto se mantiene mientras continúe recibiendo señales de reconocimiento, y el contacto concluye cuando se hayan transmitido todos los datos.

http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/



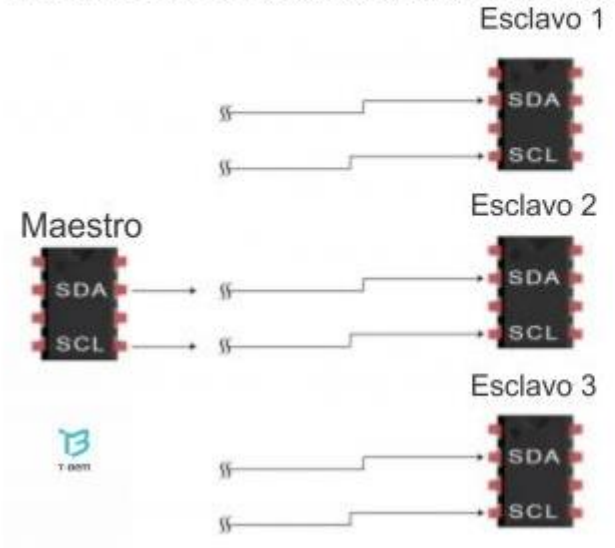
https://www.digikey.be/en/maker/projects/getting-started-with-stm32-i2c-example/ba8c2bfef2024654b5dd10012425fa23



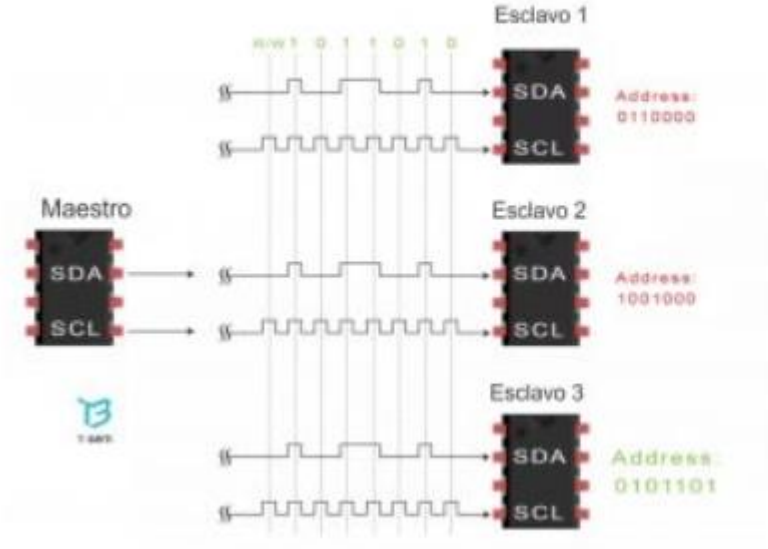
Pasos de ejecución del I2C.

Pasos del I2C en la ejecución.

1.- El Maestro envía la condición de Inicio (Start) a cada Esclavo que esté conectado en la vía SDA, cambia el nivel de voltaje a Bajo, y deja la vía SCL en estado Alto.

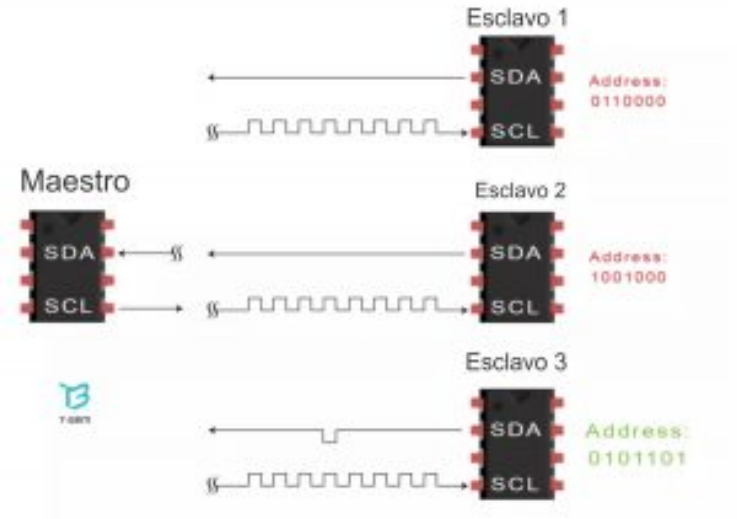


2.- El Maestro envía la dirección de 7 a 10 bits, a cada uno de los Esclavos para identificar al Esclavo con el que se quiere comunicar.

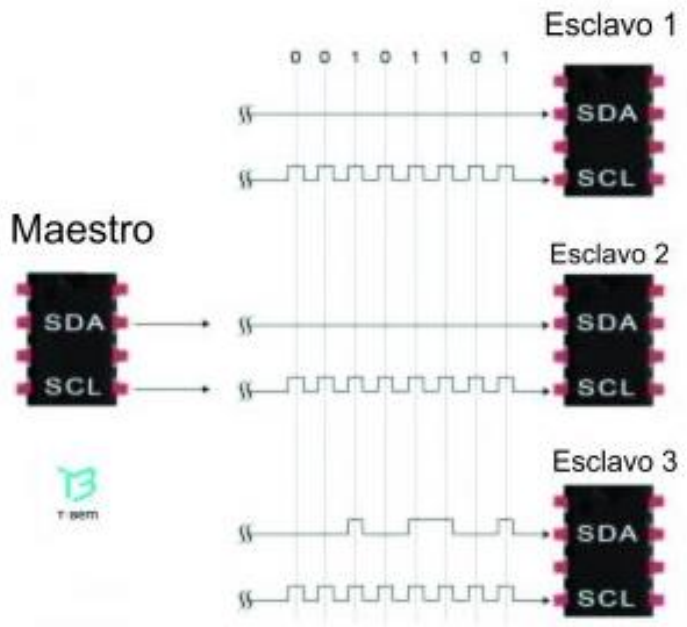


https://teslabem.com/nivel-intermedio/fundamentos/

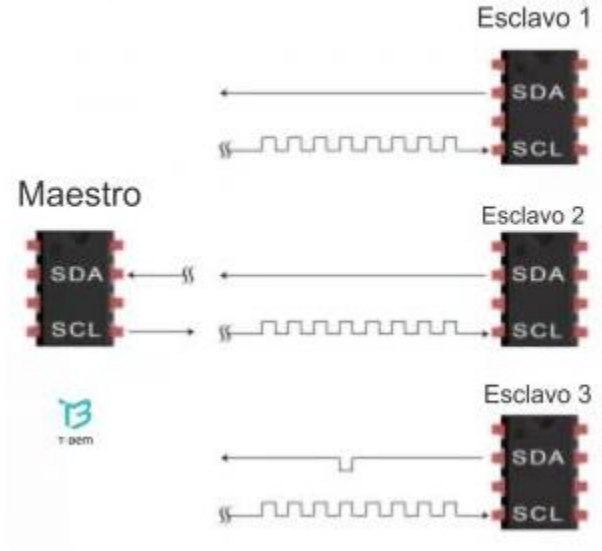
3.- Cada Esclavo recibe la dirección y la compara con su propia dirección. Si la dirección coincide, el Esclavo envía un bit ACK, y pone la vía SDA en nivel Bajo de voltaje. Si la dirección no es la misma, entonces los Esclavos no hacen nada y dejan la vía SDA en el mismo nivel de voltaje Alto.



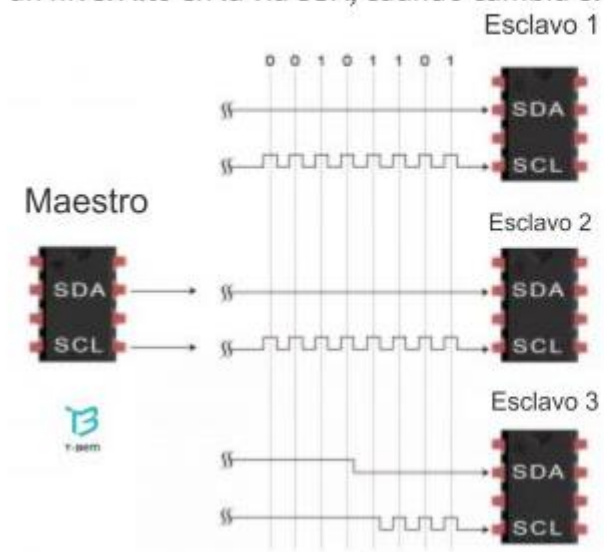
4.- El Maestro envía o recibe los Frames de información.



5.- Después de que cada Frame de información fue enviado, el dispositivo que recibe (ya sea Esclavo o Maestro) va a enviar un bit ACK al remitente, para notificarle que la información se recibió exitosamente.



6.- Para concluir la transmisión de información, el Maestro envía al Esclavo la condición de paro (Stop) con un nivel Alto en la vía SDA, cuando cambia el estado de SCL a Alto.



I2C main features

- I^2C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset

I2C features[1]	I2C1	I2C2	I2C3	I2C4
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X

The I2C is clocked by an independent clock source which allows to the I2C to operate independently from the PCLK frequency.

This independent clock source can be selected from the following three clock sources:

- PCLK1: APB1 clock (default value)
- HSI: high speed internal oscillator
- SYSCLK: system clock

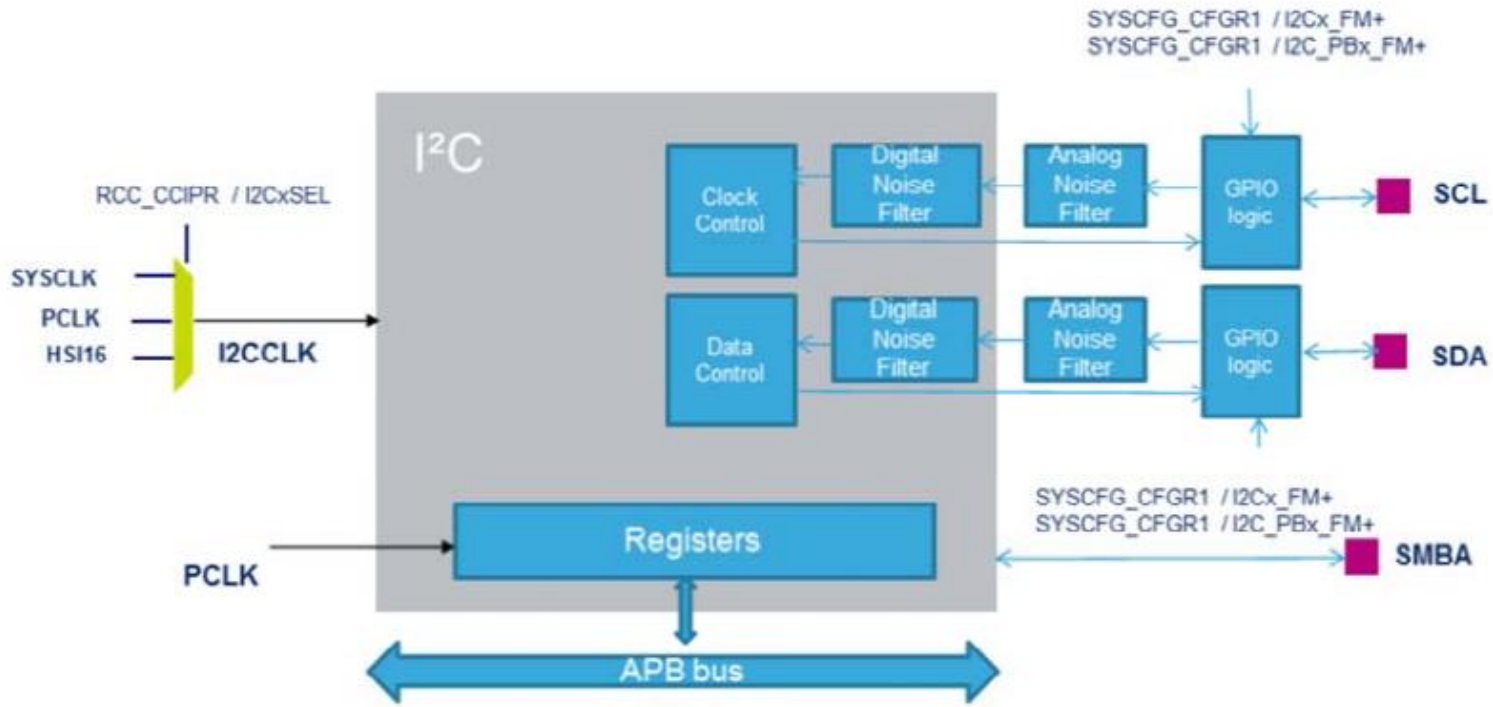
I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2C bus.

This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

Block diagram



Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

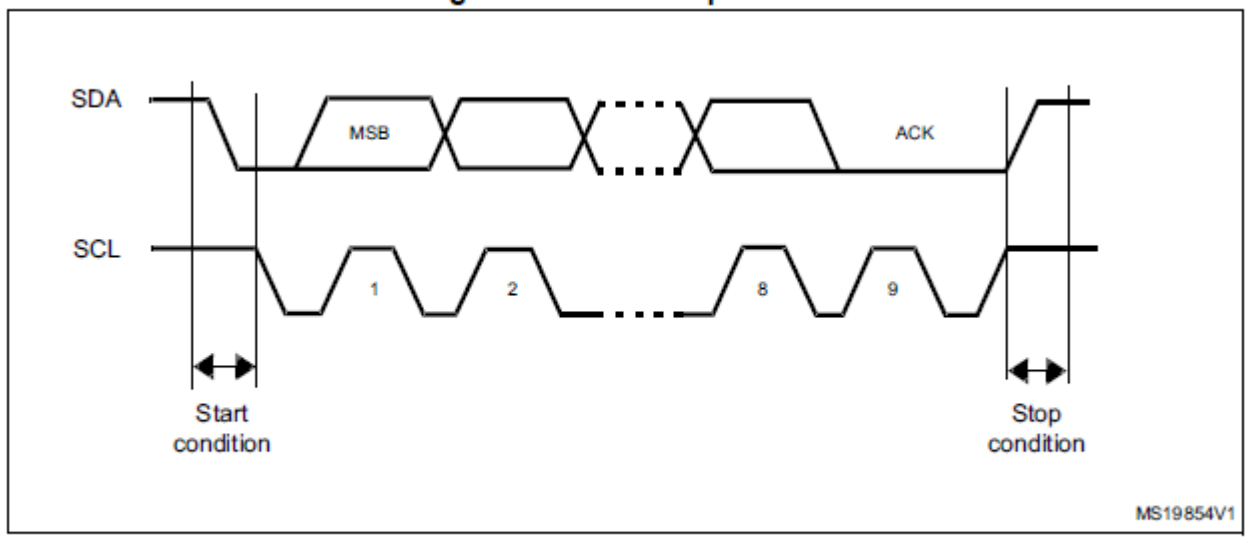
Communication flow

In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

Figure 304. I^2C bus protocol

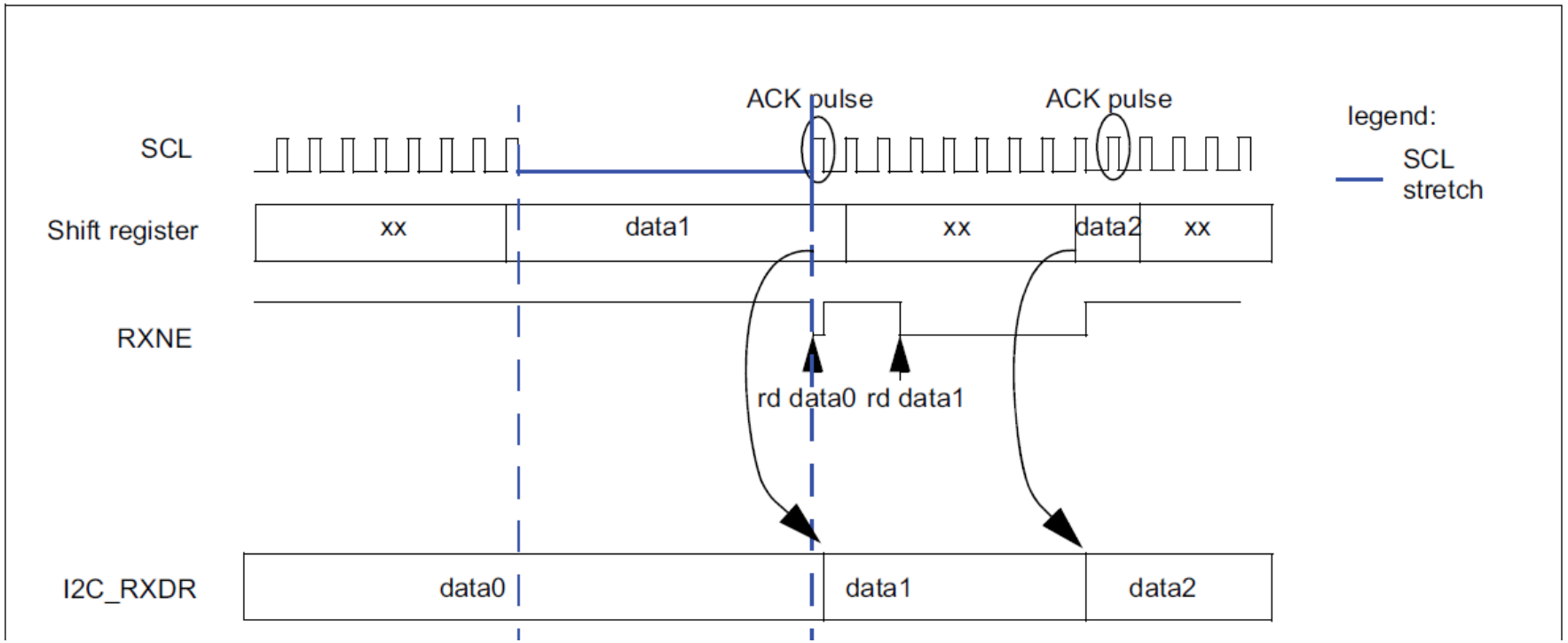


Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

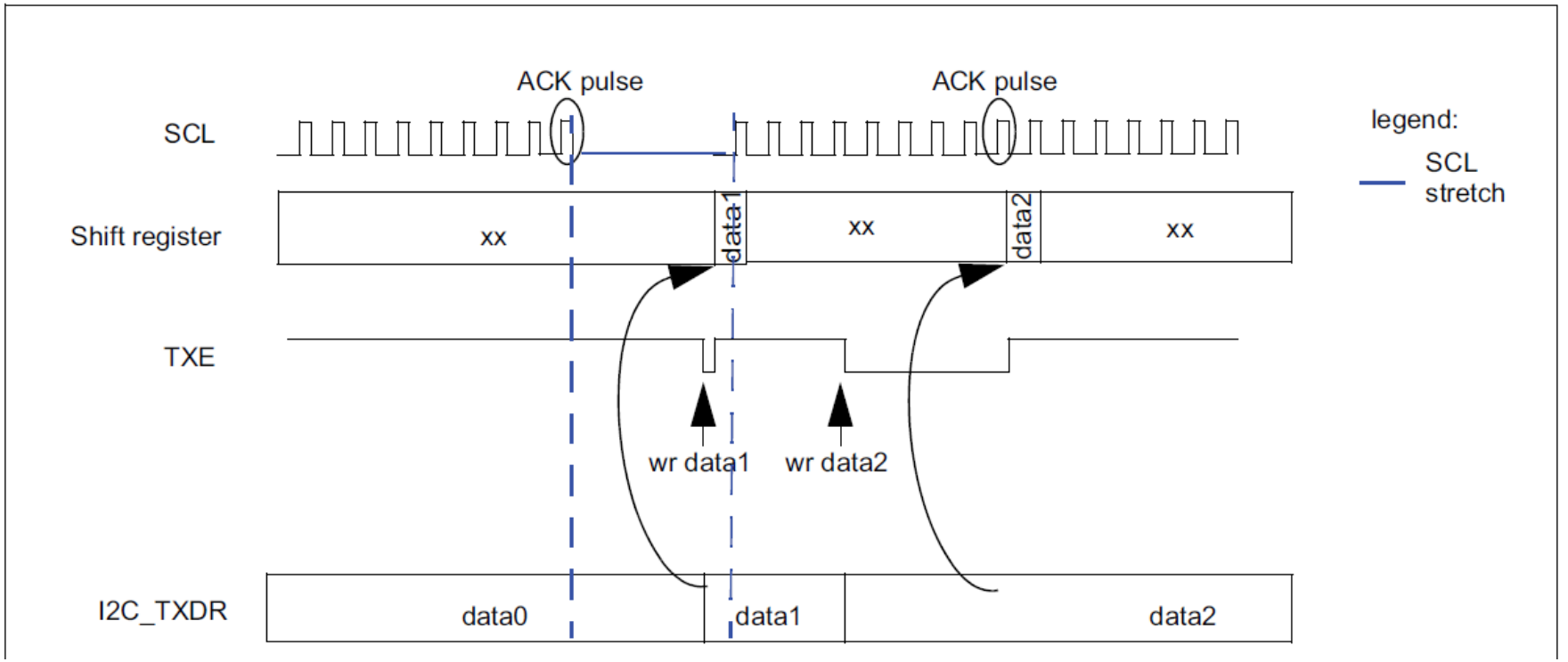
Figure 307. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 308. Data transmission



Ejemplo:

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

Puertos y Pines:

- PF1-SCL (Reloj).
- PF0-SDA (Datos).
- Modulo I2C

RCC_AHB1ENR peripheral clock register.

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

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

Asignación de Puertos PF0 y PF1.

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

Table 12. STM32F745xx and STM32F746x

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6
		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
Port E	PE14	-	TIM1_CH4	-	-	-	SPI4_MOSI	-
	PE15	-	TIM1_BKIN	-	-	-	-	-
	PF0	-	-	-	-	I2C2_SDA	-	-
	PF1	-	-	-	-	I2C2_SCL	-	-
	PF2	-	-	-	-	I2C2_SMBA	-	-
	PF3	-	-	-	-	-	-	-

GPIOx_Moder port moder register

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

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

GPIOx_OTYPER port output type register.

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

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

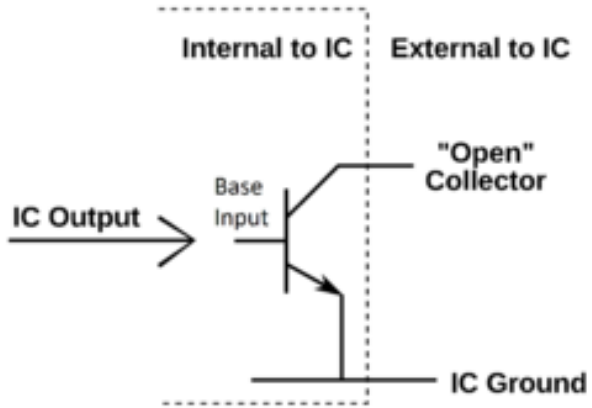
Bits 31:16 Reserved, must be kept at reset value.

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



GPIOx_PUPDR port pull-up/pull-down register.

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

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

GPIOx_OSPEEDR port output speed register.

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

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

SCL SDA

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 peripheral clock enable register.

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

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

DCKCFGR2 dedicated clocks configuration register.

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

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

Control Register (I2C_CR1)

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

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

Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I^2C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows to suppress spikes with a programmable length of 1 to 15 I2CCLK periods.

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	rw	rw	rw

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).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter will filter spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to15 t_{I2CCLK}
Note: If the analog filter is also enabled, the digital filter is added to the analog filter.
This filter can only be programmed when the I2C is disabled (PE = 0).

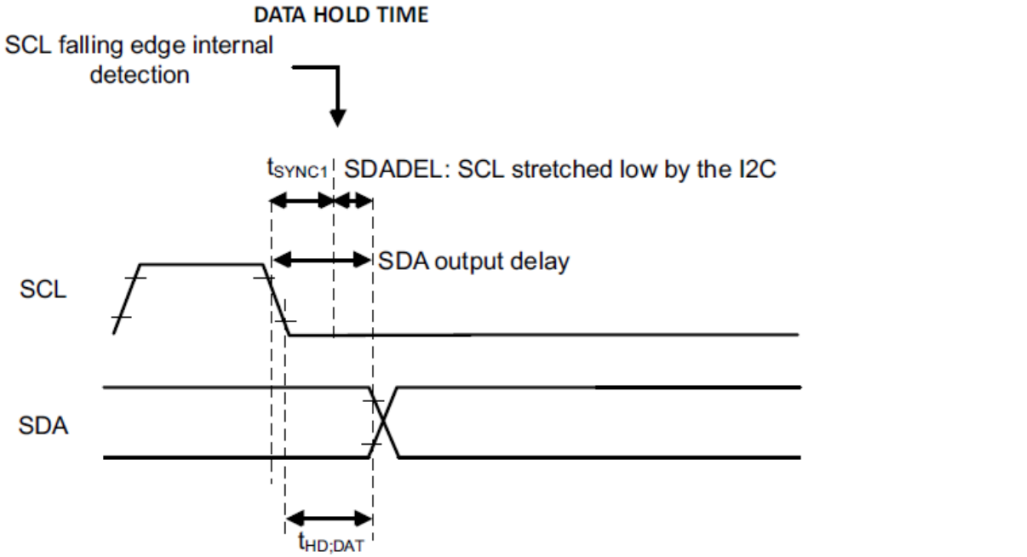
Configuración de tiempos:

Los tiempos deben ser configurados de tal forma que permitan la transferencia y configuración de información usada en los módulos maestros y esclavos. Esto se logra mediante la configuración del PRESC, SCLDEL, SDADEL, en el registro I2C_TIMINGR.

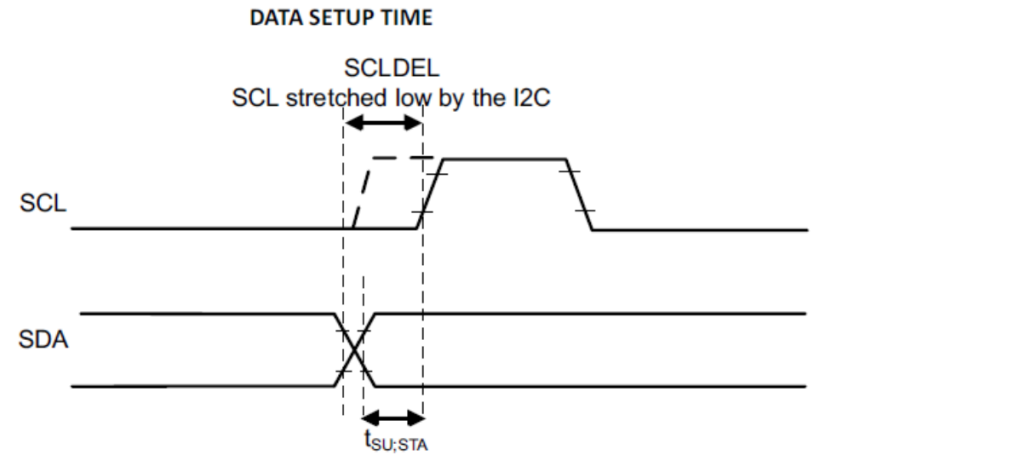
Table 161. Examples of timings settings for f_{I2CCLK} = 16 MHz

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

Figure 287. Setup and hold timings



Data hold time: in case of transmission, the data is sent on SDA output after the SDADEL delay, if it is already available in I2C_TXDR.

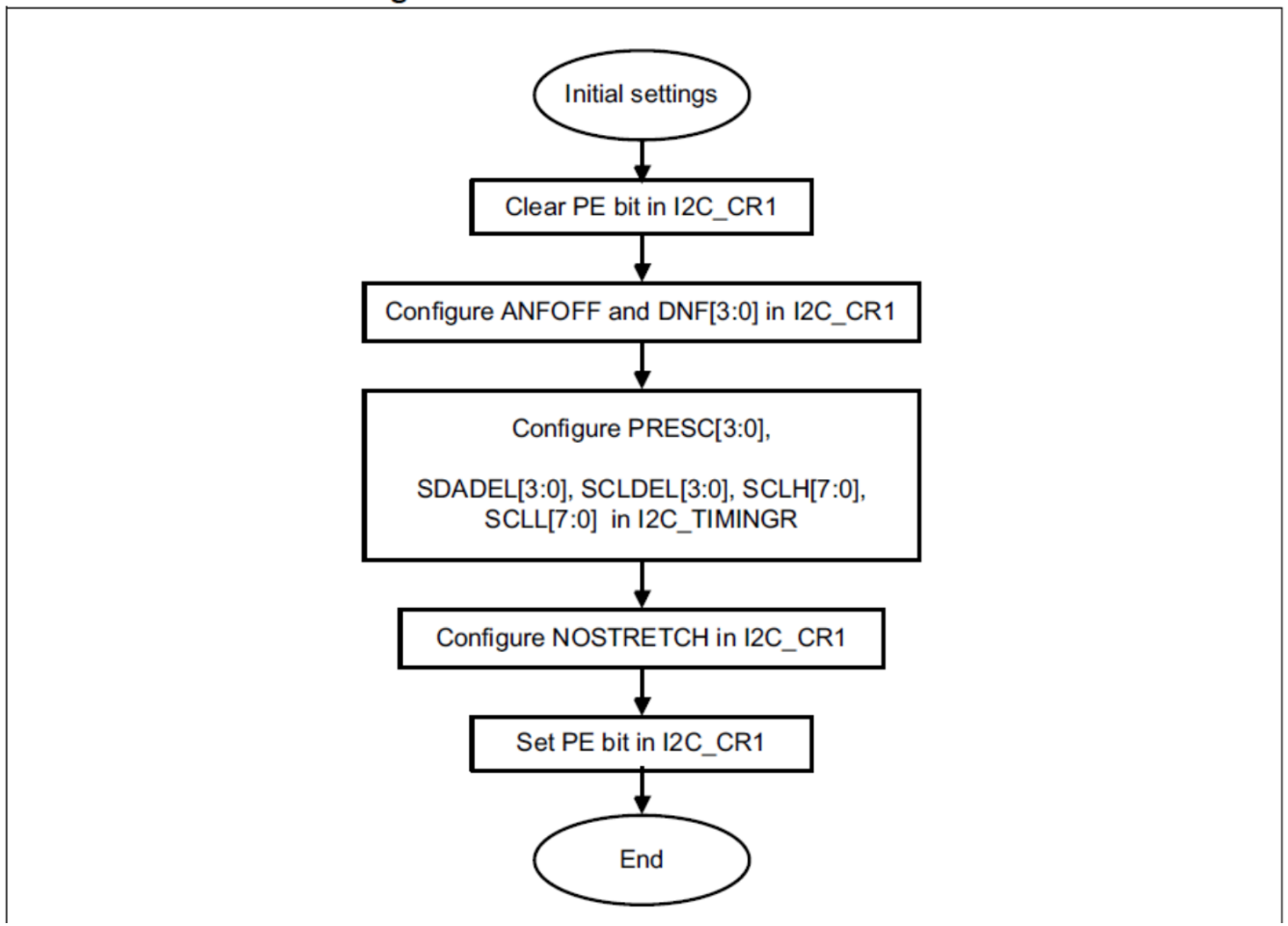


Data setup time: in case of transmission, the SCLDEL counter starts when the data is sent on SDA output.

MSv40108V1

Los tiempos deben ser configurados de tal forma que permitan la transferencia y configuración de información usada en los módulos maestros y esclavos. Esto se logra mediante la configuración del PRESC, SCLDEL, SDADEL, en el registro I2C_TIMINGR.

Figure 288. I2C initialization flowchart



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

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			

11 0000 0100 0010

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

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

11000001000010000011110010011

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.

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

30.7.2 Control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

NBYTES = 2

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

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

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

10000000100010000001111110

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

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.

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.

Bits 9:8 **SADD[9:8]**: Slave address bit 9:8 (master mode)

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

These bits are don't care

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

These bits should be written with bits 9:8 of the slave address to be sent

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

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.

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.

```
// 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
```

30.7.1 Control register 1 (I2C_CR1)

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

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).

30.7.1 Control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

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

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.

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.

```
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;
}
```

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

In case of an arbitration loss, the master automatically switches back to slave mode acknowledge its own address if it is addressed as a slave.

Simple master mode management

For payloads <= 255 bytes: only 1 write action needed!

- START = 1
- SADD: slave address
- RD_WRN: transfer direction
- NBYTES = N: number of bytes to be transferred
- **AUTOEND** = 1: STOP automatically sent after N data

AUTOEND	Description
0: Software end mode	End of transfer software control after NBYTES bytes of data are transferred: Transfer Complete (TC) flag is set and an interrupt generated if enabled. A Restart or Stop condition can be requested by software
1: Automatic end mode	Stop condition is automatically sent after NBYTES bytes of data are transferred

```
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
}
```

```
//---------------------I2C2-------------------------------------

  GPIOF-> AFR[0]  |=0X00000044; //AF4: PF1->SCL - PF0->SDA |I2C2)
  GPIOF-> MODER   |=0X0000F00A; //PF0 Y PF1 en modo alternante - PF6 Y PF7 Analog
  GPIOF-> OTYPER  |=0X0003;     //Open drain
  GPIOF-> PUPDR   |=0X00000005; //Pull-up
  GPIOF-> OSPEEDR |=0X0000000C; //PF0->SDA: Low speed - PF1->SCL: Very high speed
```

```
//------------------Funcion para enviar---------------------

void I2C2_Write (char Dat)
  {
    //Envia el dato el maestro (I2C2)
    I2C2->CR2 |= (1UL<<13); //bit de START bit 13
    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;
  }
```

```
//-------------------Funcion para recibir----------

char I2C4_Read ()
  {
    //Recibe el dato el esclavo (I2C4)
    I2C4->CR2 |= (1UL<<13);  //bit de START
    while (!(I2C4->ISR & I2C_ISR_RXNE)){};
    Dato = I2C4->RXDR;
    while (!(I2C4->ISR & I2C_ISR_STOPF));
    I2C4->ISR &= ~I2C_ISR_STOPF;
    I2C4->CR2=0;
    return Dato;
  }
```

30.4.7 I2C slave mode

I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional slave addresses are required, the 2nd slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

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 16 **SBC**: Slave byte control
This bit is used to enable hardware byte control in slave mode.
0: Slave byte control disabled
1: Slave byte control enabled

Slave Byte Control mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

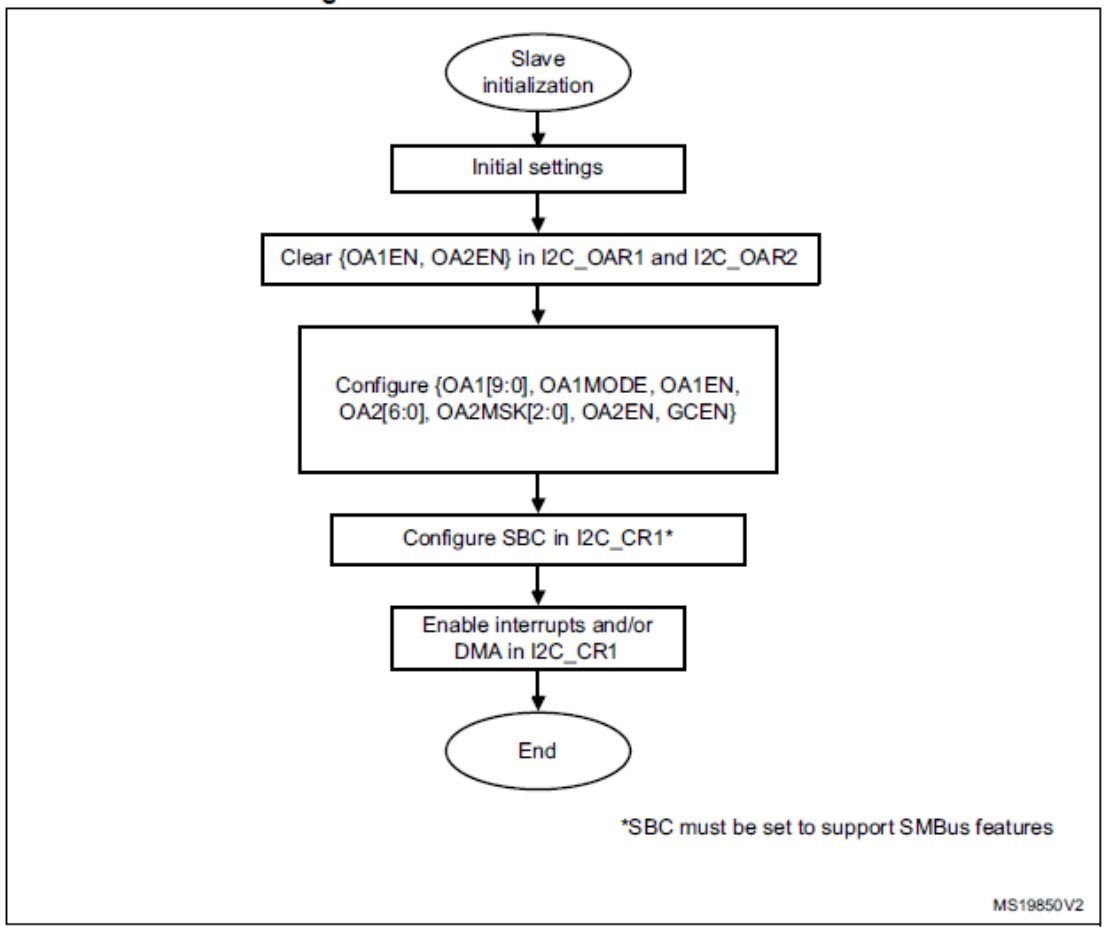
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

Bit 15 **NACK**: NACK generation (slave mode)
The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE=0.
0: an ACK is sent after current received byte.
1: a NACK is sent after current received byte.
Note: Writing '0' to this bit has no effect.
This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.
When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.
When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Figure 309. Slave initialization flowchart



30.7.3 Own address 1 register (I2C_OAR1)

Address offset: 0x08

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 15 **OA1EN**: Own Address 1 enable
0: Own address 1 disabled. The received slave address OA1 is NACKed.
1: Own address 1 enabled. The received slave address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own Address 1 10-bit mode
0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.
Note: This bit can be written only when OA1EN=0.

Bits 9:8 **OA1[9:8]**: Interface address
7-bit addressing mode: do not care
10-bit addressing mode: bits 9:8 of address
Note: These bits can be written only when OA1EN=0.

Bits 7:1 **OA1[7:1]**: Interface address
7-bit addressing mode: 7-bit address
10-bit addressing mode: bits 7:1 of 10-bit address
Note: These bits can be written only when OA1EN=0.

Bit 0 **OA1[0]**: Interface address
7-bit addressing mode: do not care
10-bit addressing mode: bit 0 of address
Note: This bit can be written only when OA1EN=0.

30.7.4 Own address 2 register (I2C_OAR2)

Address offset: 0x0C

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bit 15 **OA2EN**: Own Address 2 enable
0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN=0.

As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN=0.

Bit 0 Reserved, must be kept at reset value.

```
/--------------------I2C2 Maestro----------------------------

 //Disable I2C2 peripheral
 I2C2->CR1 &= ~I2C_CR1_PE;
 while(I2C2->CR1 & I2C_CR1_PE);
 //I2C2->TIMINGR
 I2C2 -> TIMINGR |=0X30420F13;
 I2C2 -> CR2     |=0X0201207E;  //SADD = 0X7E -  N BYTES =2
 // Use 7-bit addresses
 I2C2 -> CR2 &=~ I2C_CR2_ADD10; //Modo maestro receptor, solo
 // 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 I2C2 peripheral
 I2C2 -> CR1 |= I2C_CR1_PE; //habilita SCL y SDA para el peri
```

```
//  ***** CONFIGURACION I2C4 (ESCLAVO) *****
I2C4 -> CR1 &= ~I2C_CR1_PE;              // DESACTIVA EL I2C4 PARA CONFIGURAR
while (I2C4 -> CR1 & I2C_CR1_PE);        // ESPERA HASTA QUE ESTE DESACTIVADO
I2C4 -> OAR1 &= ~I2C_OAR1_OA1EN;         // DESACTIVA EL REGISTRO OA1EN PARA CONF
while (I2C4 -> OAR1 & I2C_OAR1_OA1EN){} // ESPERA HASTA QUE ESTE DESACTIVADO
I2C4 -> OAR1 |= 0XFE;                    // DIRECCION QUE VA TENER EL ESCLAVO
I2C4 -> TIMINGR |= 0X30420F13;           // PRESC -> 3 - SCLDEL -> 4 - SDADEL ->
I2C4 -> CR2 |= 0X02012000;               // # BYTES = 2
I2C4 -> CR2 &= ~I2C_CR2_ADD10;           // MAESTRO RECEPTOR
I2C4 -> CR2 |= I2C_CR2_AUTOEND;          // AUTOFINALIZACION ACTIVADA
I2C4 -> CR1 |= I2C_CR1_ANFOFF;           // FILTRO DE RUIDO DESACTIVADO
I2C4 -> CR1 |= I2C_CR1_NOSTRETCH;        // NOSTRETCH DESACTIVADO
I2C4 -> CR1 |= I2C_CR1_SBC;              // HABILITA EL BIT ESCLAVO
I2C4 -> OAR1 |= I2C_OAR1_OA1EN;          // ACTIVAMOS EL OA1EN
I2C4 -> CR1 |= I2C_CR1_PE;               // HABILITA EL I2C4
```