Sistemas Embebidos

Bus I2C

Ángel Perles

r06

Contenido

- Objetivos
- Introducción
- Señales SDA y SCL
- Direccionamiento
- Ejemplo sensor AMS TSL2561
- STM32L4xx I2C
  - Características
  - Master mode
  - Ejemplo HAL
  - CubeMX
- Ejemplo con el AMS TSL2561
- Ejemplo con el AMS TSL2561
  - Módulo de ADAFruit

Objetivos

- Conocer el bus serie I2C
- Emplearlo con microcontroladores STM32
- Aplicarlo a un sensor real

Introducción

- I2C es un estándar “de facto” para comunicaciones entre chips
- Desarrollado hace un montón de años por Philips (NXP)
- Características:
  - Comunicación serie **síncrona**
  - Tipo maestro/esclavo**s**. Posibilidad multimaestro
  - No balanceada -> poca distancia, nivel PCB
  - Dos líneas
    - Serial Data Line (SDA)
    - Serial Clock Line (SCL)
  - Cada dispositivo tiene con una dirección prefijada
- Más detalles en la especificación oficial
  - copia en https://www.disca.upv.es/aperles/asignatures/sist_emb/UM10204.pdf

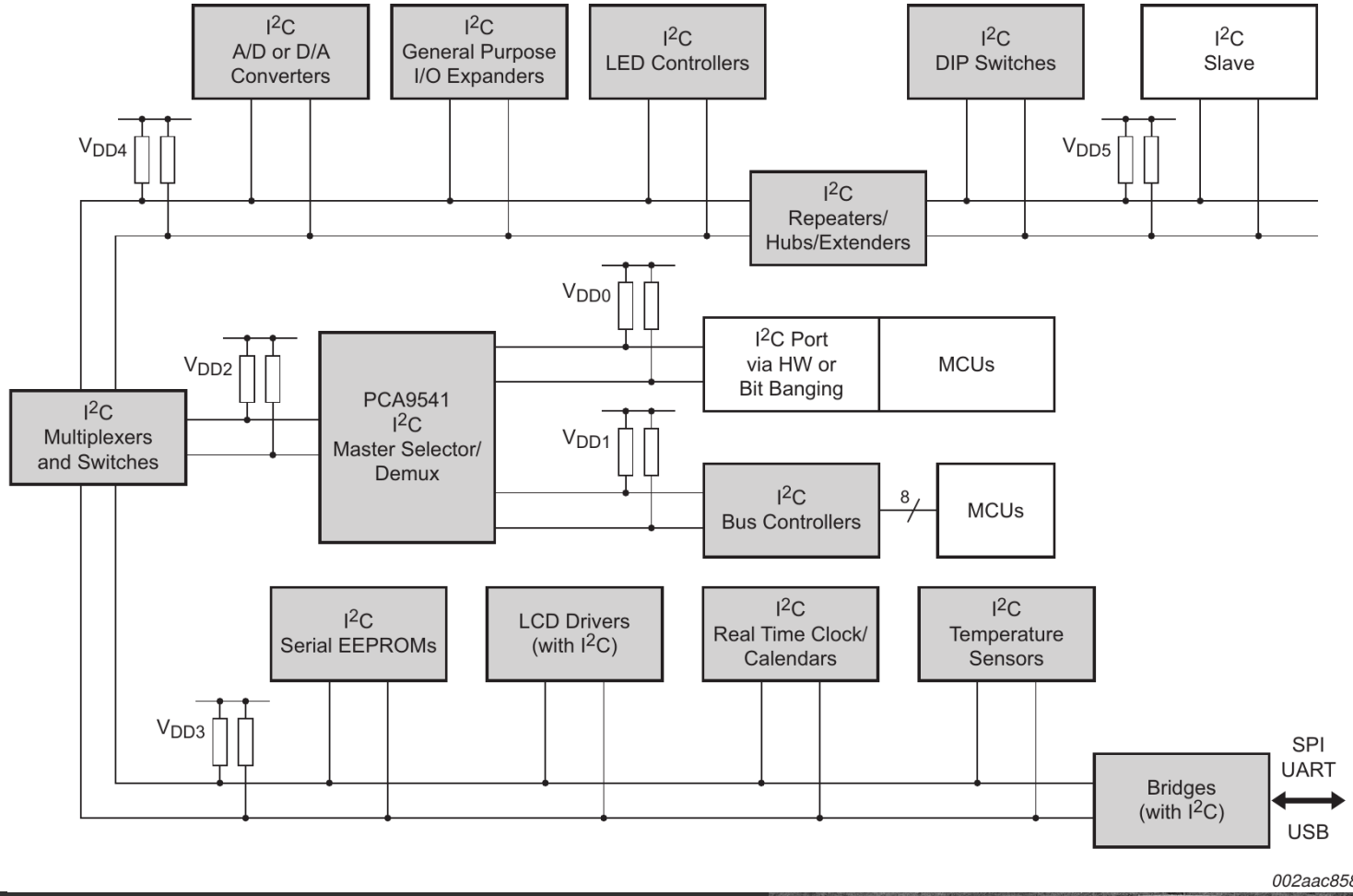
Introducción

- Revisiones incorporando mayor velocidad máxima
- Y otras características

Versiones	Velocidad máxima
Standard mode	100 kbps
Fast mode	400 kbps
High speed mode	3,4 Mbps
Ultra fast mode	5 Mbps

Introducción

- Diagrama con muchas de las opciones



Introducción

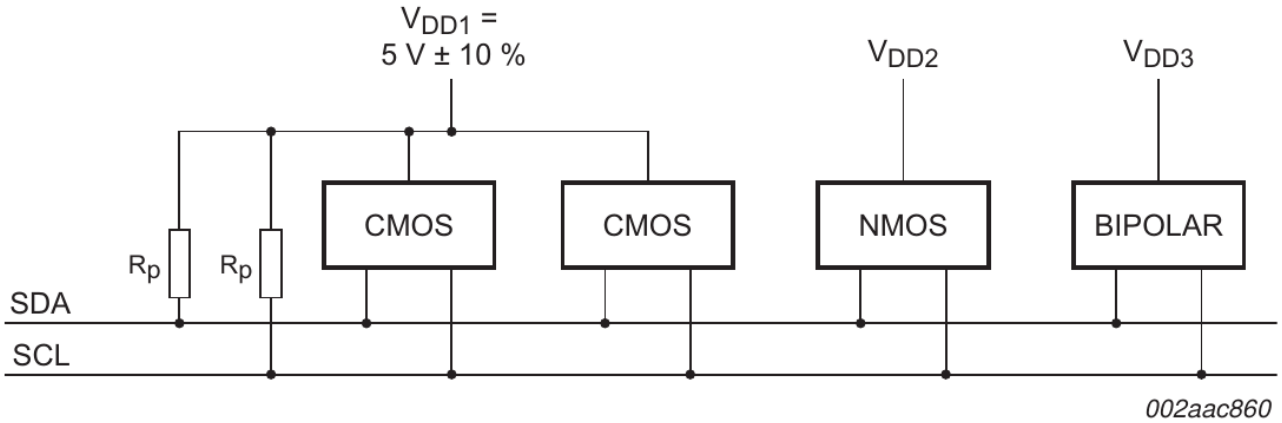
- Terminología

Table 1. Definition of I²C-bus terminology

Term	Description
Transmitter	the device which sends data to the bus
Receiver	the device which receives data from the bus
Master	the device which initiates a transfer, generates clock signals and terminates a transfer
Slave	the device addressed by a master
Multi-master	more than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	procedure to ensure that, if more than one master simultaneously tries to control the bus, only one is allowed to do so and the winning message is not corrupted
Synchronization	procedure to synchronize the clock signals of two or more devices

Señales SDA y SCL

- Bus en colector/drenador abierto
- Resistencias de pull-up
- El bus se comporta como una AND cableada
- <u>Solo</u> el maestro genera SCL



V_{DD2}, V_{DD3} are device-dependent (for example, 12 V).

Fig 3. Devices with various supply voltages sharing the same bus

Señales SDA y SCL

- Los cambios de nivel se meten en el estado bajo de reloj
- En el estado alto no debe modificarse el dato
- 1 ciclo de reloj -> 1 dato (bit)

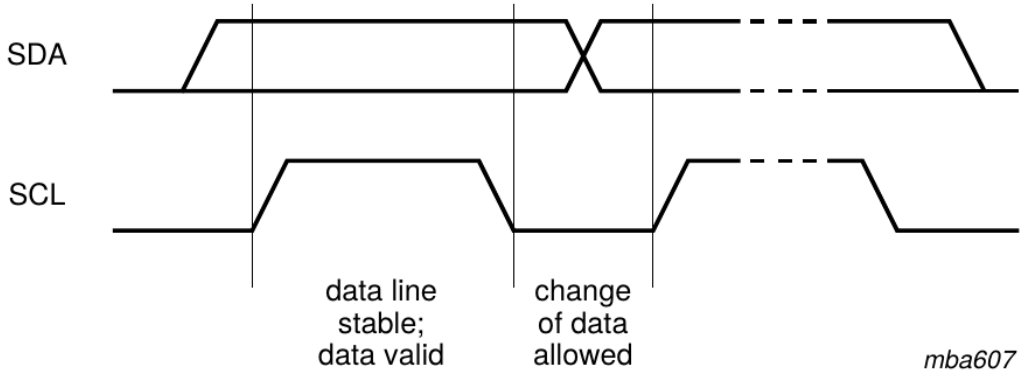


Fig 4. Bit transfer on the I²C-bus

Señales SDA y SCL

- La transacciones empiezan con un START y terminan con un STOP
- Una transacción son 8 bits …(huy, se me ha escapado una mentira)
- 1 ciclo de reloj -> 1 dato (bit)

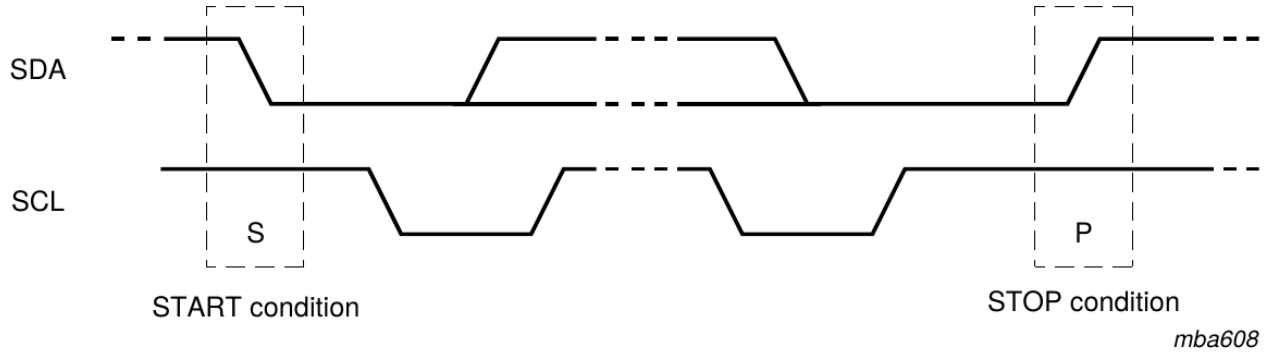


Fig 5. START and STOP conditions

- FÍJATE: que el “start” y el “stop” incumplen traspa anterior

Señales SDA y SCL

- Formato de la transferencia
- El esclavo puede reconocer (ACK) la recepción

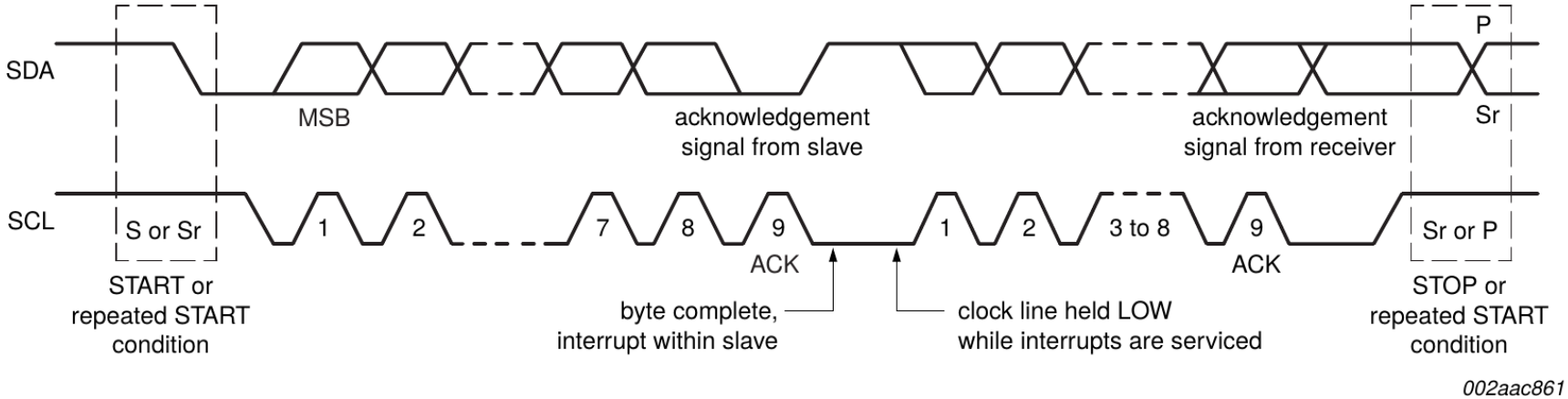


Fig 6. Data transfer on the I²C-bus

Direccionamiento

- Es decir, como selecciono con quién hablo
- Modo 7-bits + **R/W**

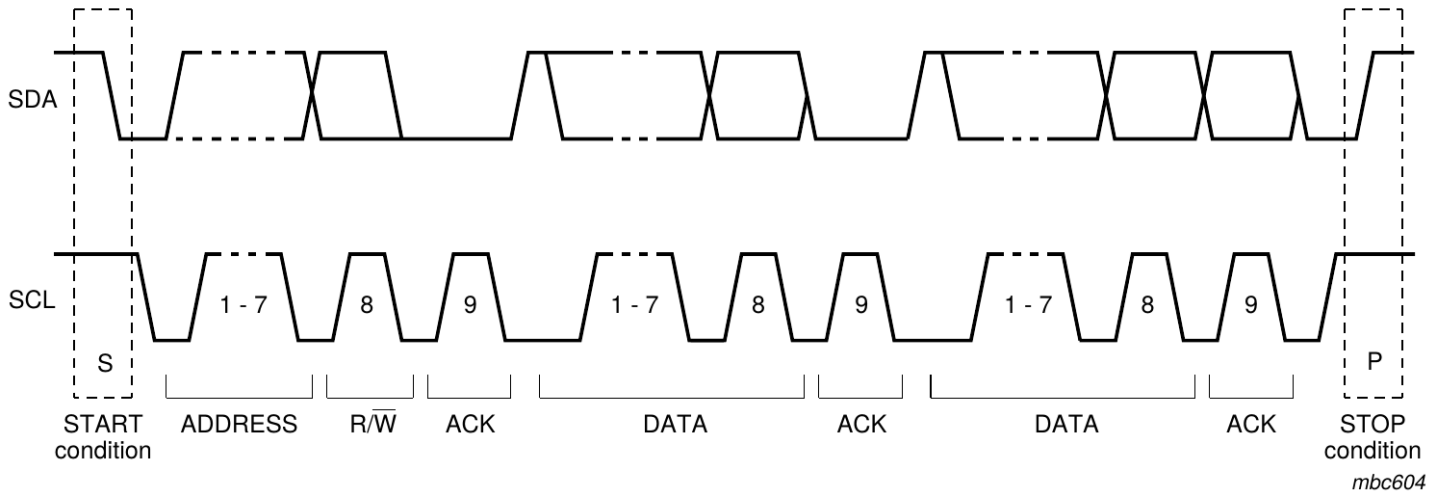


Fig 9. A complete data transfer

Fig 10. The first byte after the START procedure

Direccionamiento

- Modo 10-bits + **R/W**

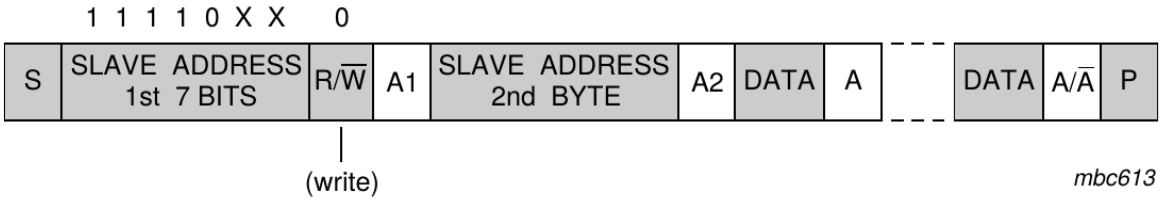
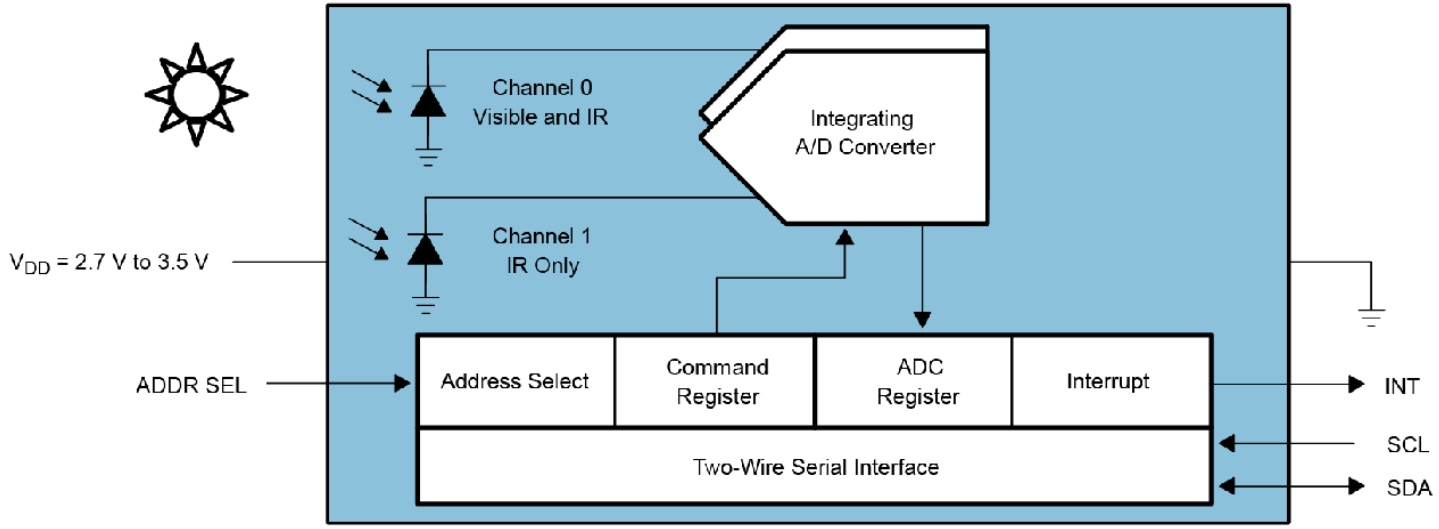


Fig 14. A master-transmitter addresses a slave-receiver with a 10-bit address

Ejemplo: AMS TSL2561

- https://ams.com/TSL2561
- Argghhhhh!!!! Lo han descatalogadooooo !!!
- Descargad y mirad el datasheet … 10 minutos



- Y van preguntas …

STM32L4xx I2C: Características

- De donde siempre (Reference manual)

39.2 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
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

- Actividad: revisa en UM10204 (I2C bus …) lo que dice de la rev03

STM32L4xx I2C: Características

- Veamos que I2C tiene nuestro micro

Table 227. STM32L475xx/476xx/486xx devices I2C implementation

I2C features[1]	I2C1	I2C2	I2C3
7-bit addressing mode	X	X	X
10-bit addressing mode	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X

Table 227. STM32L475xx/476xx/486xx devices I2C implementation (continued)

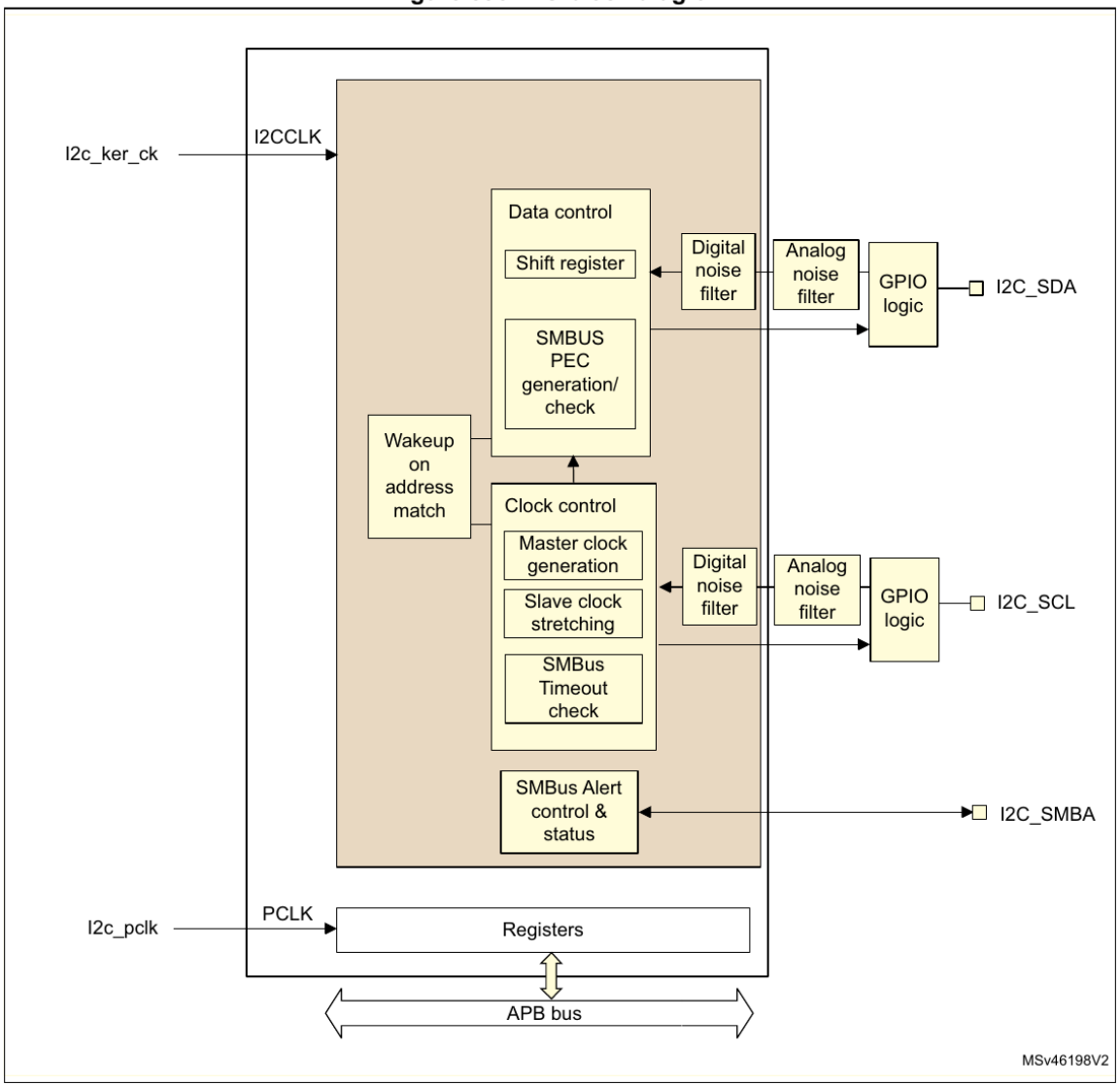
I2C features[1]	I2C1	I2C2	I2C3
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X	X	X
Independent clock	X	X	X
Wakeup from Stop 0 mode	X	X	X
Wakeup from Stop 1 mode	X	X	X
Wakeup from Stop 2 mode	-	-	X
SMBus/PMBus	X	X	X

1. X = supported.

STM32L4xx I2C: Características

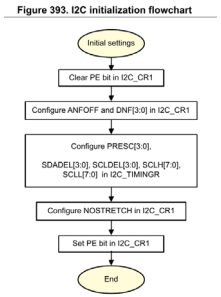
- ¡Que bien diseñan hw los de St!
- Una pena que los manuales sean infum…

Figure 390. I2C block diagram



STM32L4xx I2C: Características

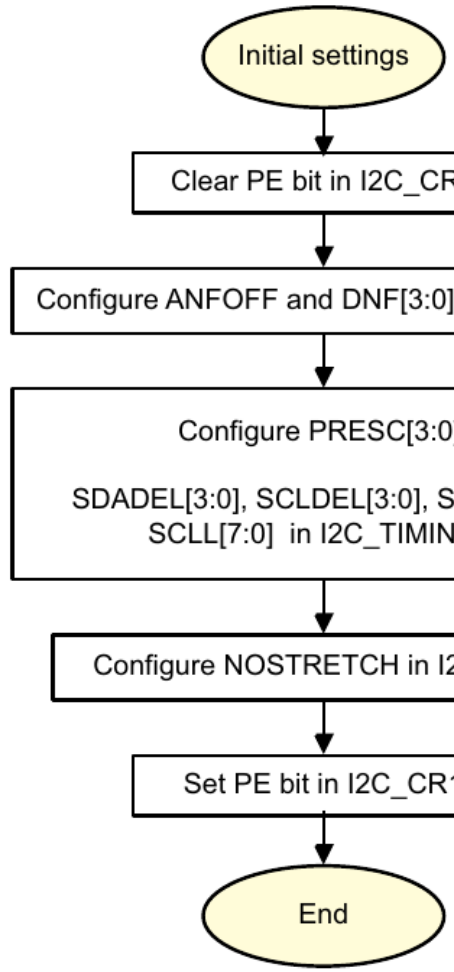
- Explico lo que puedo … a destacar
- Relojes. Han de funcionar los dos
  - El del bus (i2C_pclk)
  - El del dispositivo (i2c_ker_clk). Muy flexible y bastantes reglas
    - STM32CubeMX nos puede ayudar mucho
- Hay que seleccionar el modo antes de operar
  - Slave transmitter, slave receiver, master transmitter, master receiver
- Hay que configurar todos los “timmings” a partir del reloj
  - Vamos, muy parecido al tema de timers
    - STM32CubeMX nos puede ayudar mucho



STM32L4xx I2C: Características

- Y en este orden

Figure 393. I2C initialization flowchart



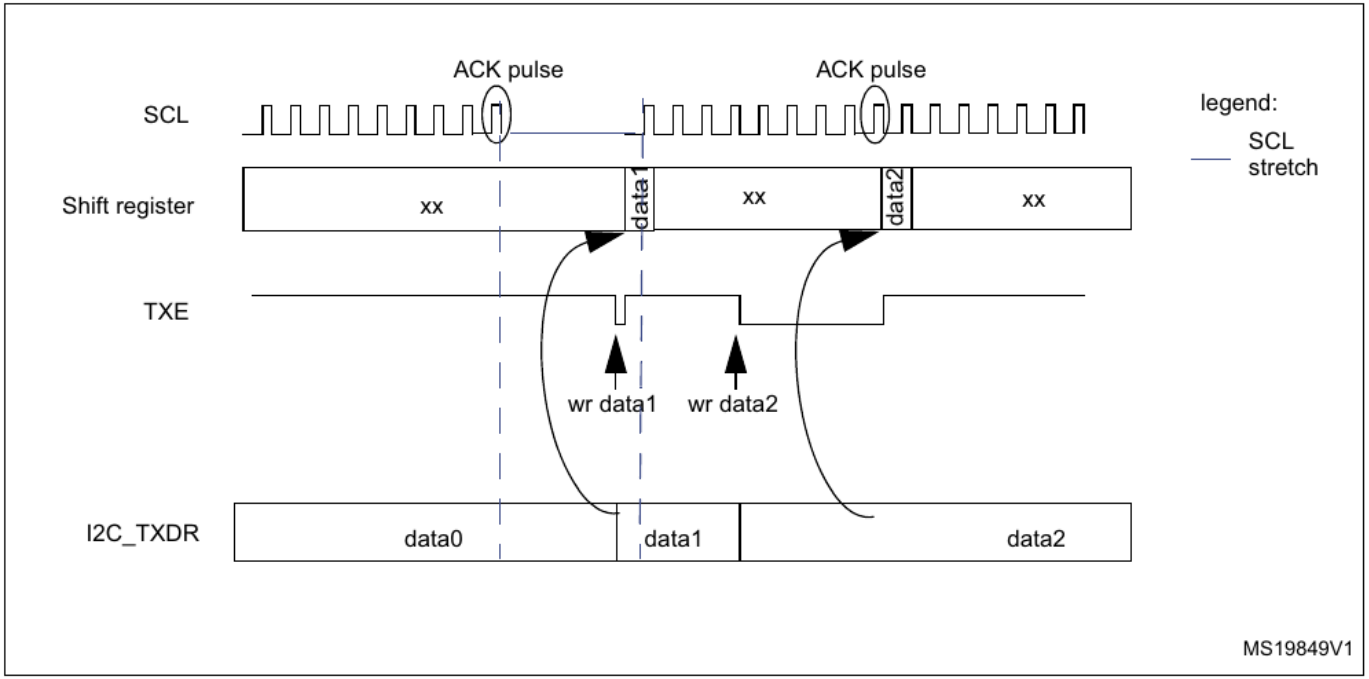
STM32L4xx I2C: Características

- El Hw asiste la RX y la TX mediante contadores y distintos modos

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 395. Data transmission

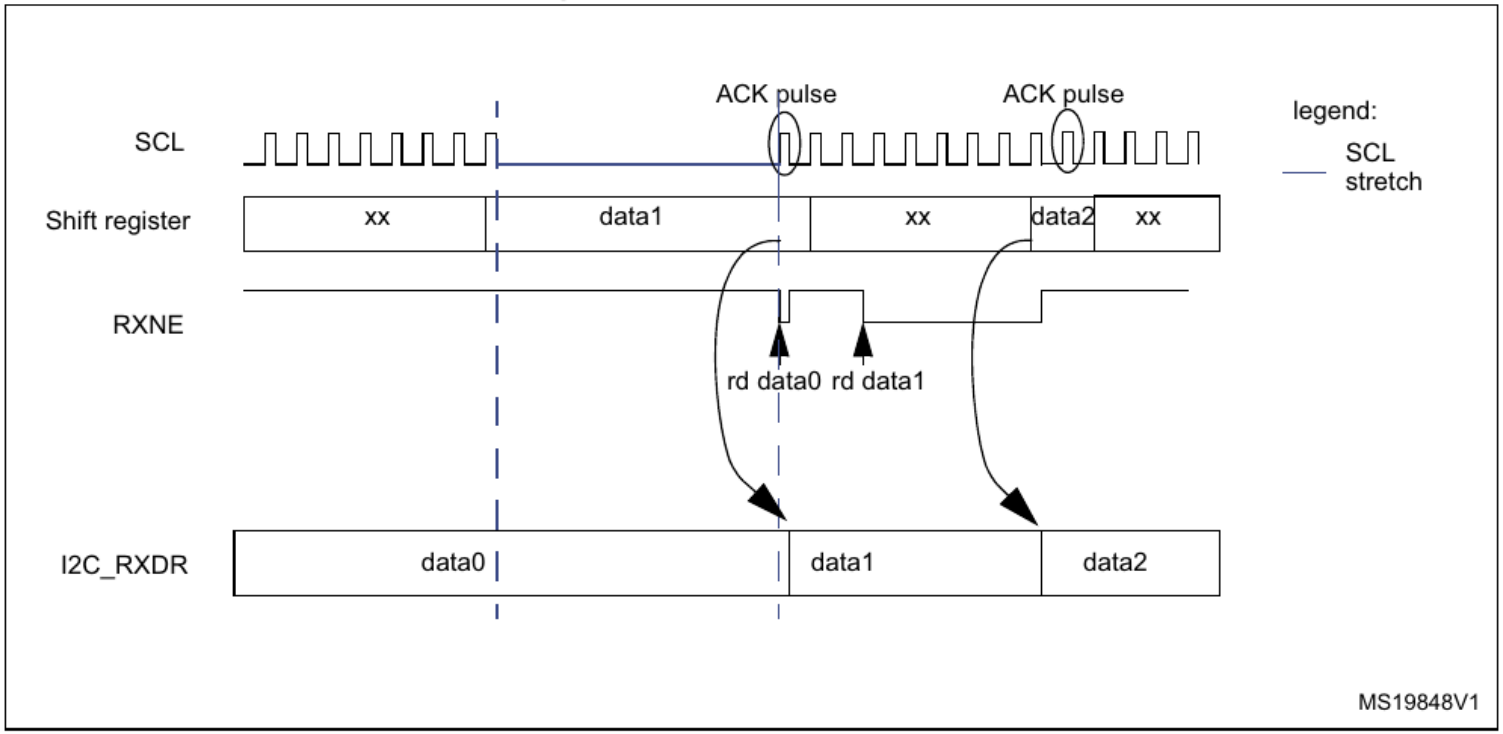


STM32L4xx I2C: Características

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 394. Data reception



STM32L4xx I2C: Características

- Y montones de interrupciones … hell!

Figure 420. I2C interrupt mapping diagram

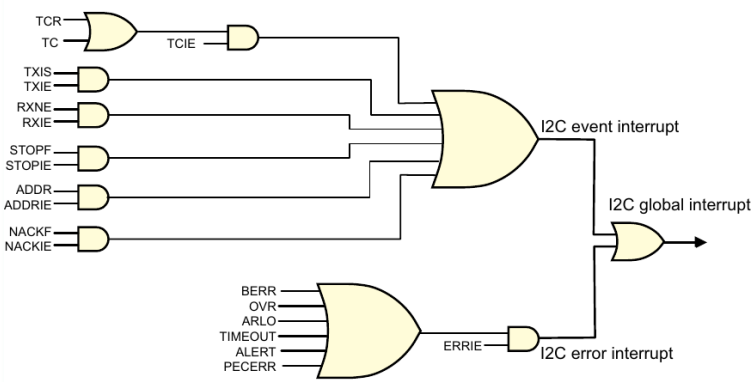


Table 241. I2C Interrupt requests

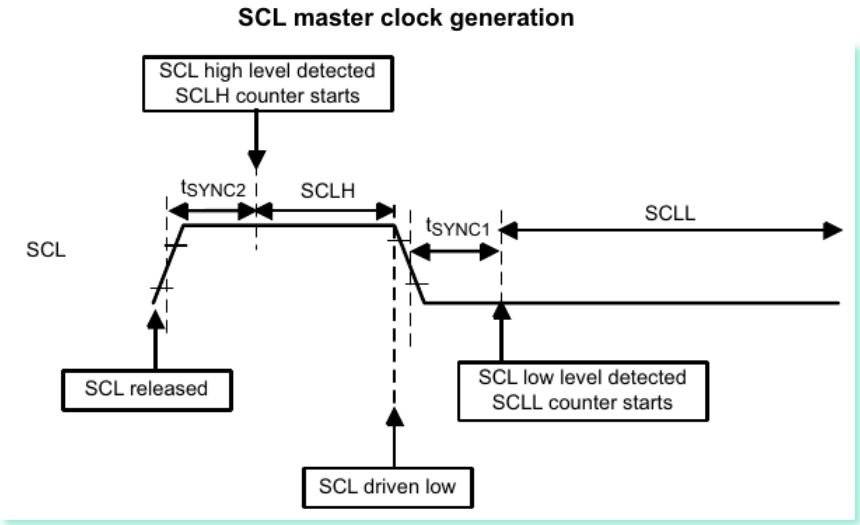
Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] ≠ 0	TCIE
Transfer complete	TC	Write START=1 or STOP=1	
Address matched	ADDR	Write ADDRCF=1	ADDRIE
NACK reception	NACKF	Write NACKCF=1	NACKIE
Bus error	BERR	Write BERRCF=1	ERRIE
Arbitration loss	ARLO	Write ARLOCF=1	
Overrun/Underrun	OVR	Write OVRCF=1	
PEC error	PECERR	Write PECERRCF=1	
Timeout/t_{LOW} error	TIMEOUT	Write TIMEOUTCF=1	
SMBus Alert	ALERT	Write ALERTCF=1	

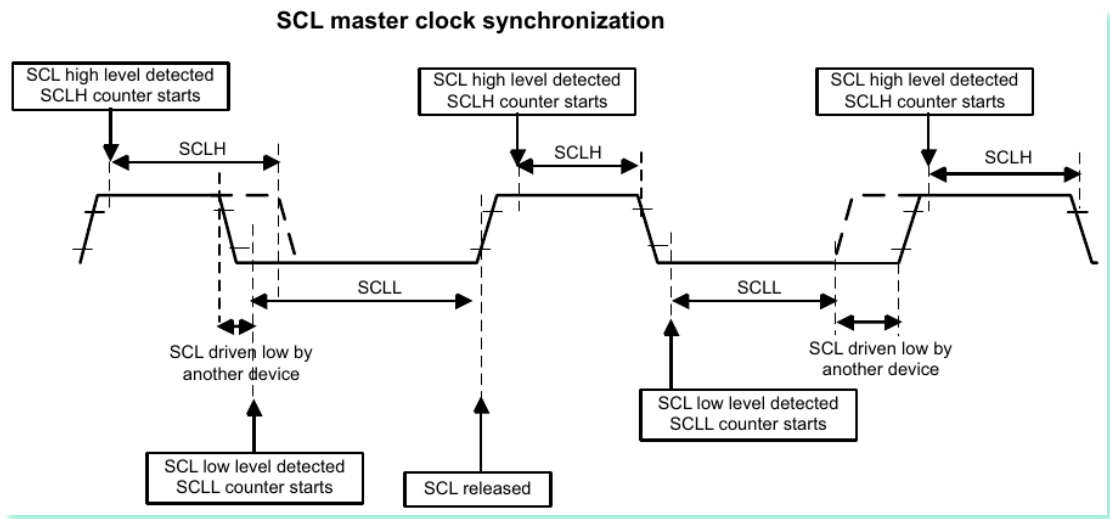
- Sería bueno informarse sobre el paradigma *productor-consumidor*
- ¿Una asignatura de microkernels? ¿En STR veis algo de eso?

STM32L4xx I2C: Master mode

- "Timming"

Figure 403. Master clock generation





STM32L4xx I2C: Master mode

- “Timming” -> ejemplos de ayuda de St

Table 232. Examples of timing settings for f_{I2CCLK} = 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
t_{SCLL}	200x250 ns = 50 µs	20x250 ns = 5.0 µs	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
t_{SCLH}	196x250 ns = 49 µs	16x250 ns = 4.0µs	4x125ns = 500ns	4x125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

- Jolines … esperemos que el “Cube” ayude

STM32L4xx I2C: Master mode

- Inicialización. Lectura.

Figure 404. Master initialization flowchart

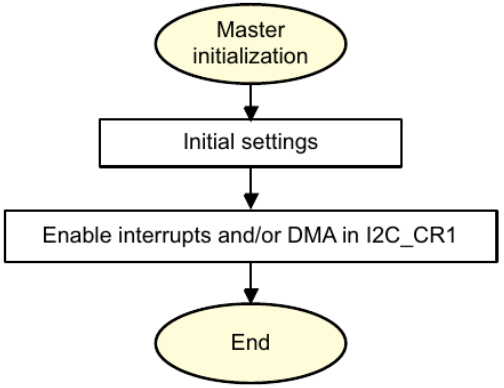
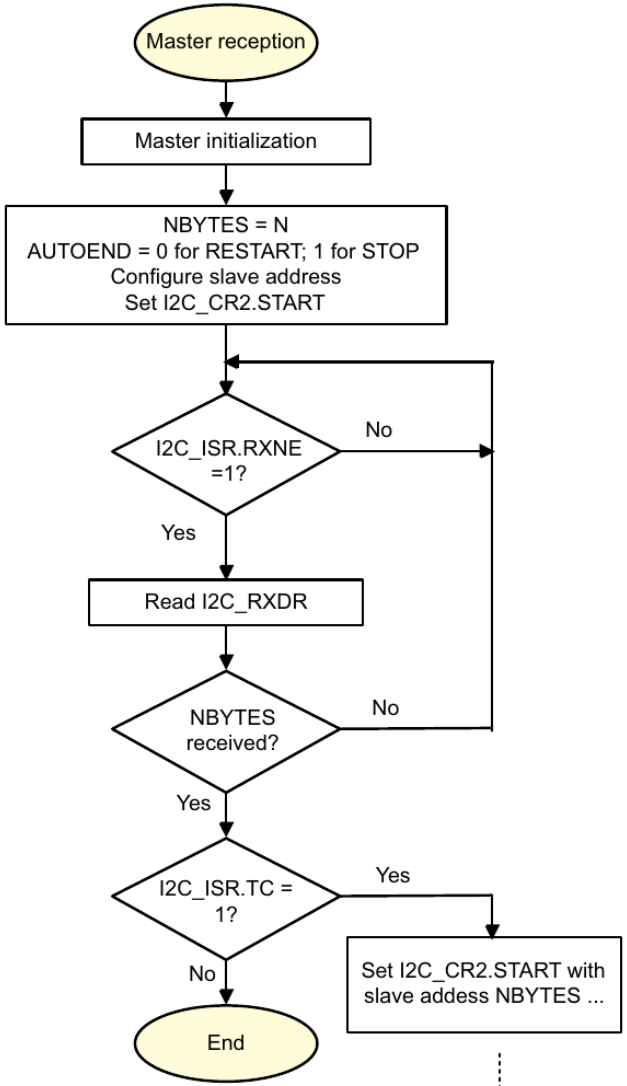
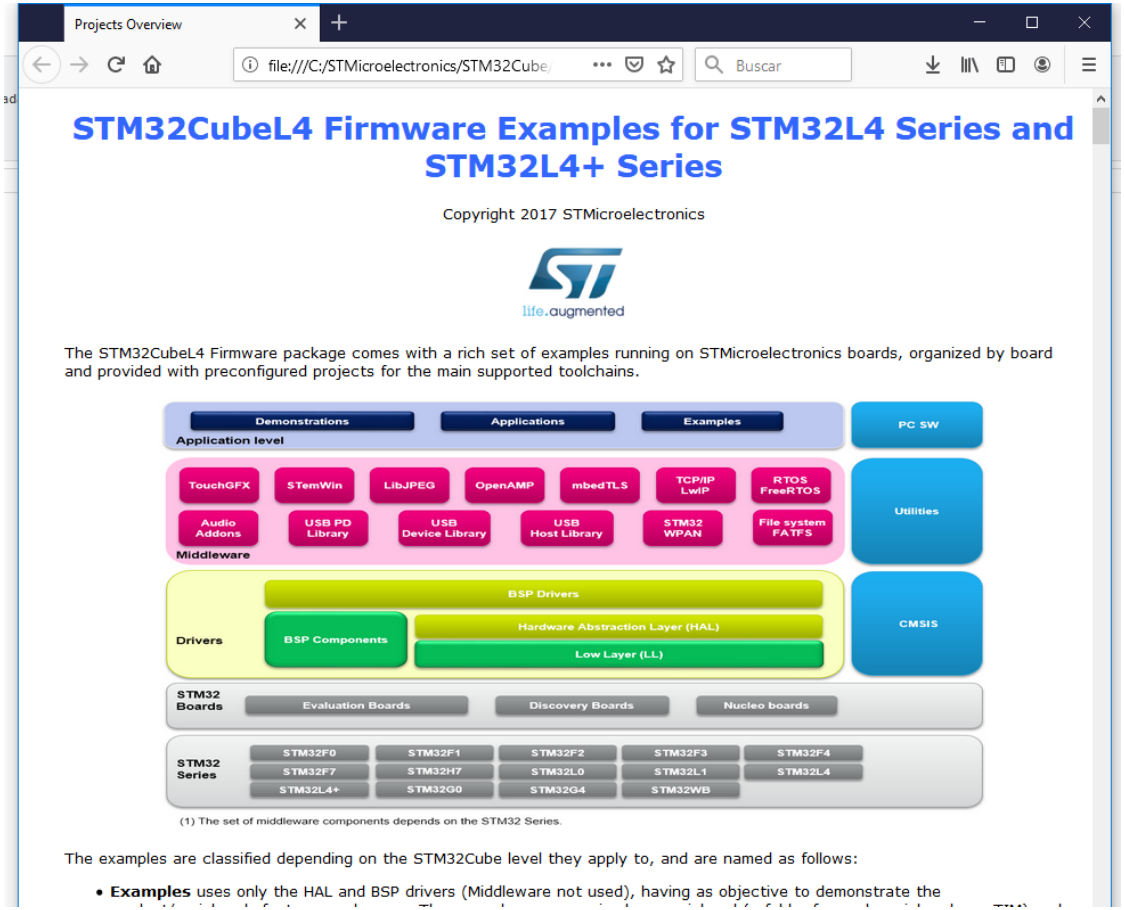


Figure 410. Transfer sequence flowchart for I2C master receiver for N≤255 bytes



STM32L4xx I2C: Ejemplo HAL

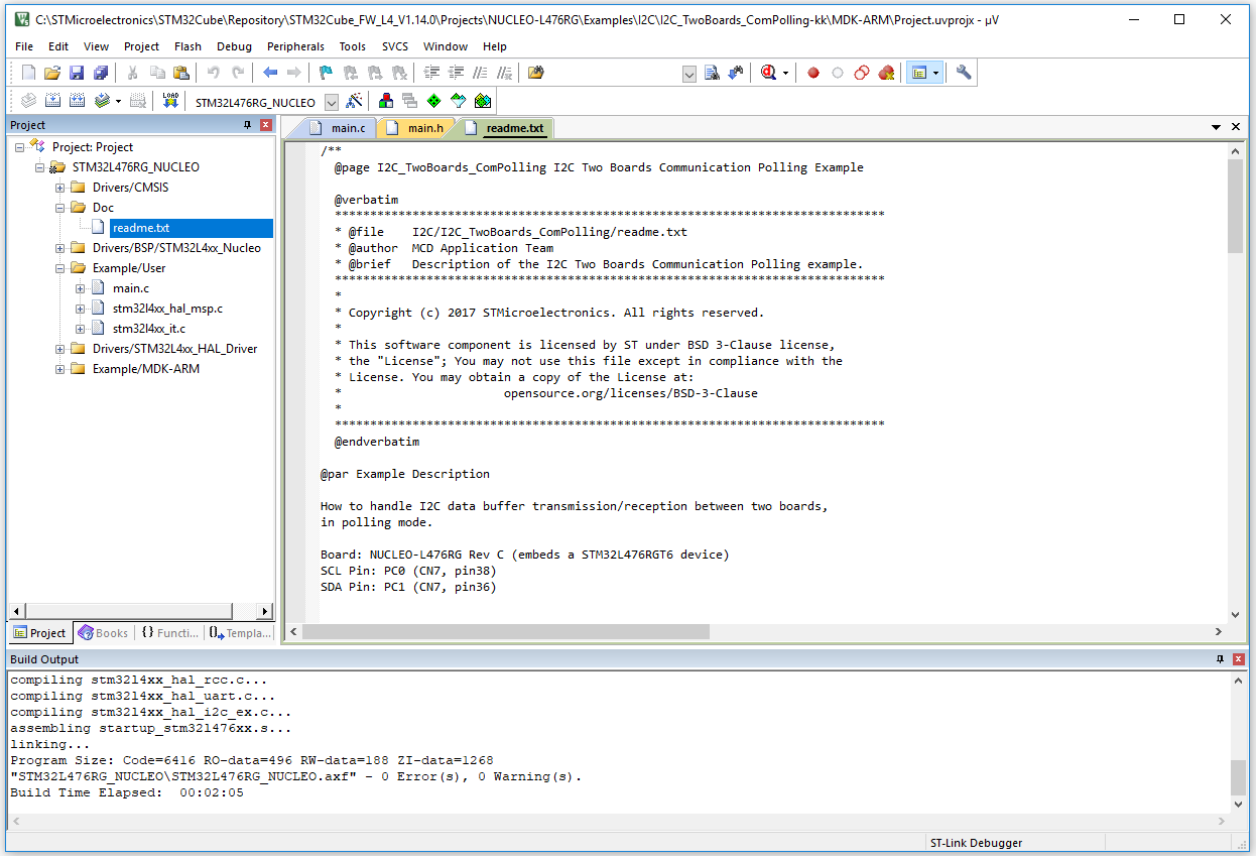
- Una manera buena de aprender es echar un vistazo a los ejemplos del firmware “Cube”
- Si no está para nuestra placa, puede estar para otra



- Veamos el archivo “html” con la explicación de los ejemplos (dentro de “projects”) (y copiar esto también al tema de ARM)

STM32L4xx I2C: Ejemplo HAL

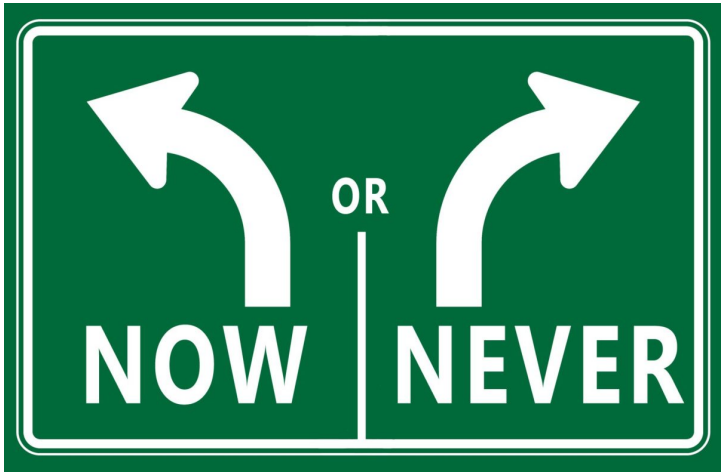
- El más sencillo parece “I2C_TwoBoards_ComPolling”
- Readme
- Licencia …
- Doxygen …
- Secciones …
- Cambio “optimización”
- Configuro “defines” MASTER …
- No me gustan los main.h
- … vamos muchas de las cosas que estáis sufriendo con el microproyecto (Mundo real)
- **Se aprende mucho de los ejemplos**

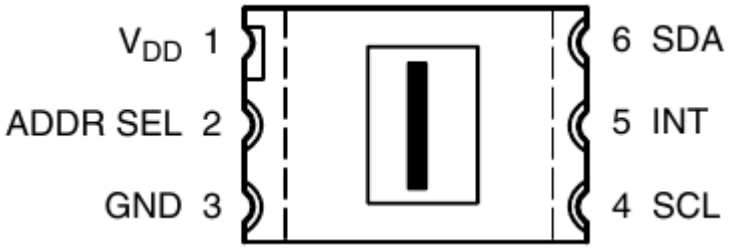


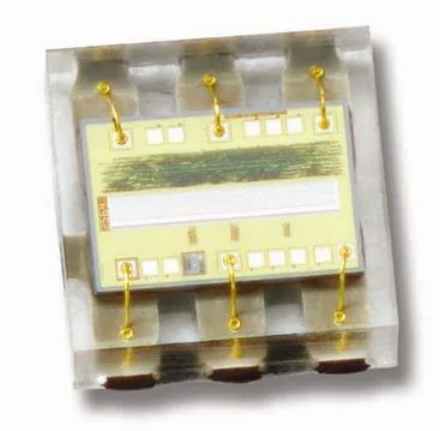
- Buscad la documentación de las funciones usadas

Ejemplo con el AMS TSL2561

- ¡Quiero leer un puñetero sensor ya!

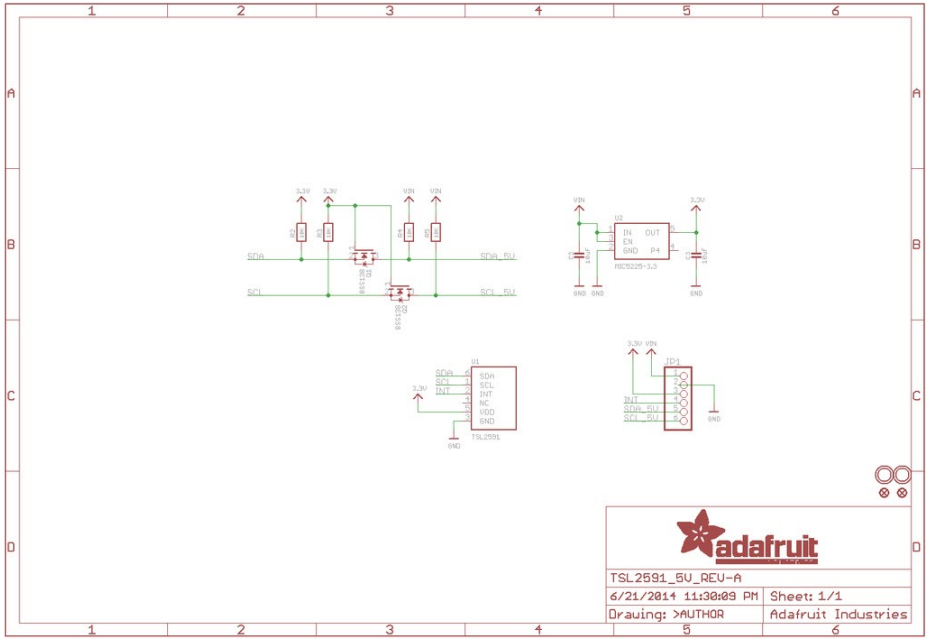


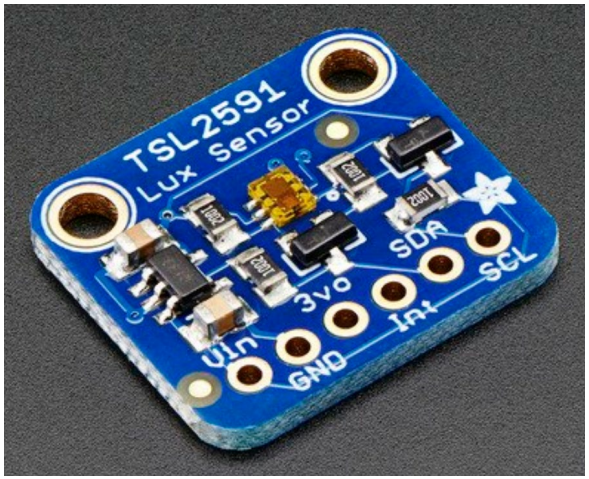




Ejemplo con el AMS TSL2561

- Podéis comprar un módulo para Arduino en Adafruit … buscadlo





- Actividad
  - ¿Qué pines de ese módulo hay que usar?
  - ¿Qué dirección I2C tiene el módulo? ¿Es de 7 o de 10 bits?
  - ¿Qué velocidad I2C se puede emplear con el módulo?

Ejemplo con el AMS TSL2561

- Veamos si CubeMX nos ayuda … directo con la Nucleo-L476RG

New Project from a Board

MCU/MPU Selector | Board Selector | Cross Selector

Board Filters

Part Number Search: NUCLEO-L476RG

Vendor

Type

MCU/MPU Series

Other

Price = 14.0

Oscillator Freq. = 0 (MHz)

Peripheral

Accelerometer		
Analog I/O	0	0
Arduino Form Factor	0	0
Audio Line In	0	0
Audio Line Out	0	0
Button	0	2
CAN	0	0
Camera		
Compass		
Custom Form Factor	0	0
Digital I/O	0	108
Ethernet		
Gyroscope		
IrDA		
Joystick		
LCD Display (Graphics)	0	0
LCD Display (Segment)	0	0
LED	0	3
Light Sensor		
Magnetometer		

Large Picture | Docs & Resources | Datasheet | Buy | Start Project

NUCLEO-L476RG

STM32 L4

STMicroelectronics NUCLEO-L476RG Board Support and Examples

ACTIVE Active
Product is in mass production

Unit Price (US$) : 14.0

Mounted device: STM32L476RGTx

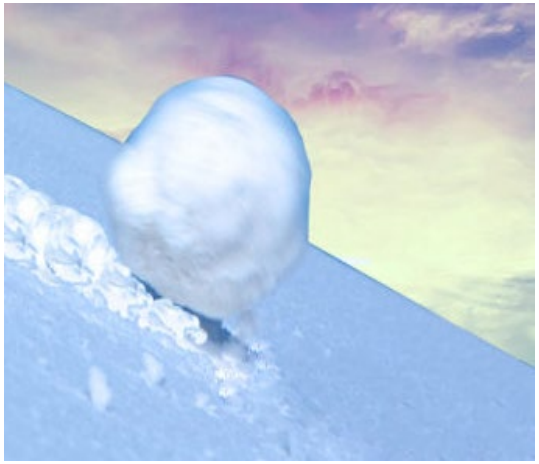
The STM32 Nucleo-64 boards provide an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the external SMPS significantly reduces power consumption in Run mode.
The Arduino™ Uno V3 connectivity support and the ST morpho headers allow the easy expansion of the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields.
The STM32 Nucleo-64 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.
The STM32 Nucleo-64 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

Features

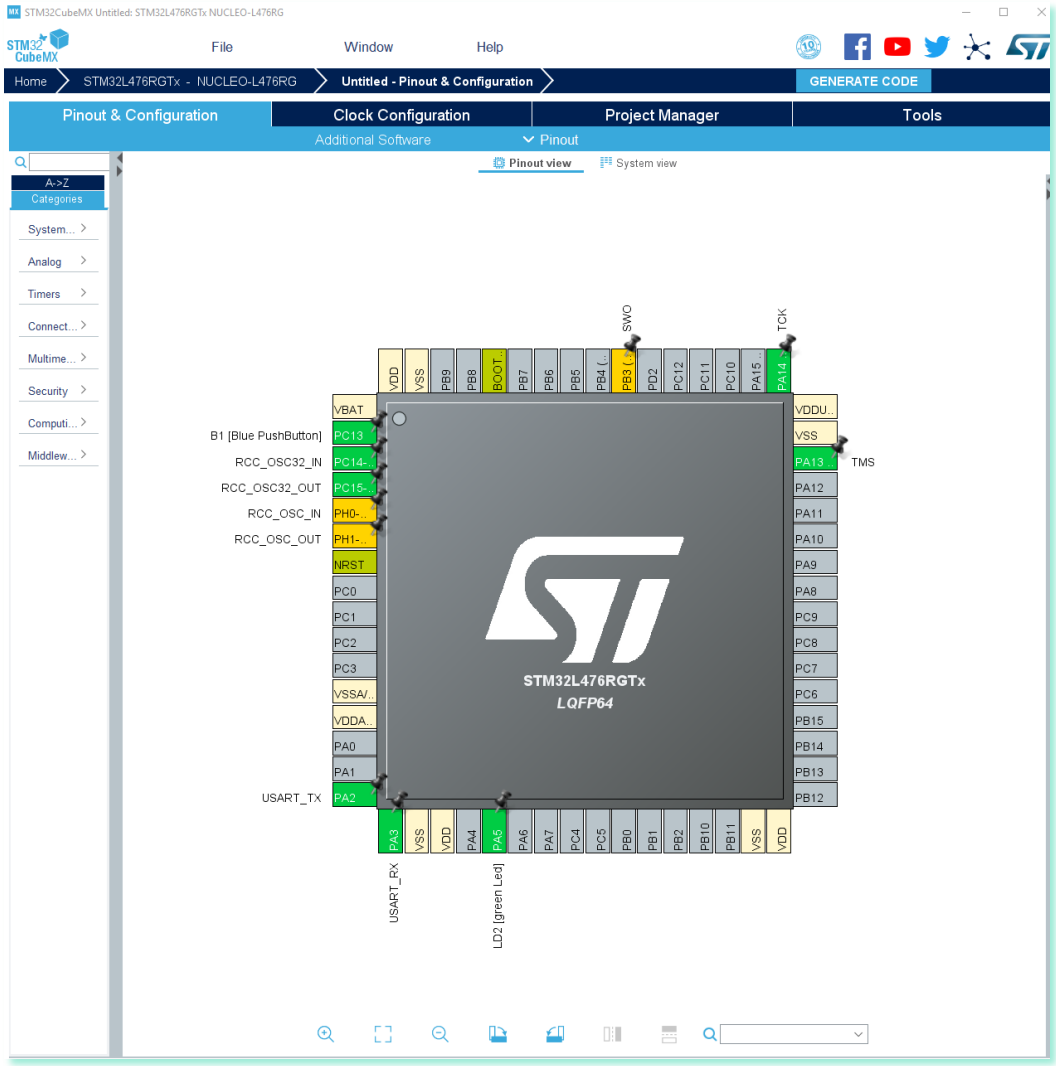
- On-board ST-LINK/V2-1
- USB VBUS, ext. VIN, ext. 5V, ext +3.3V
- STMicroelectronics Morpho connector : (2 x 38)

Boards List: 1 item

*	Overview	Part No	Type	Marketing St...	Unit Price (U...	Mounted De...	MCU/MP...	Kit Contents	Included i...
★		NUCLEO-L...	Nucleo64	Active	14.0	STM32L476...	STM32...		



Ejemplo con el AMS TSL2561



- Pito, pito … -> I2C1
- Standard mode
- GPIO no-pull (el sensor ya las lleva)
- ¡Anda! genera todo el lio de timings, filtros y cosas raras
- Project manager … lo vemos

Ejemplo con el AMS TSL2561

- A ver qué ha hecho el CubeMX … anda, lo típico de las HAL
  - No he querido meter descripción HAL I2C en este tema ni futuros

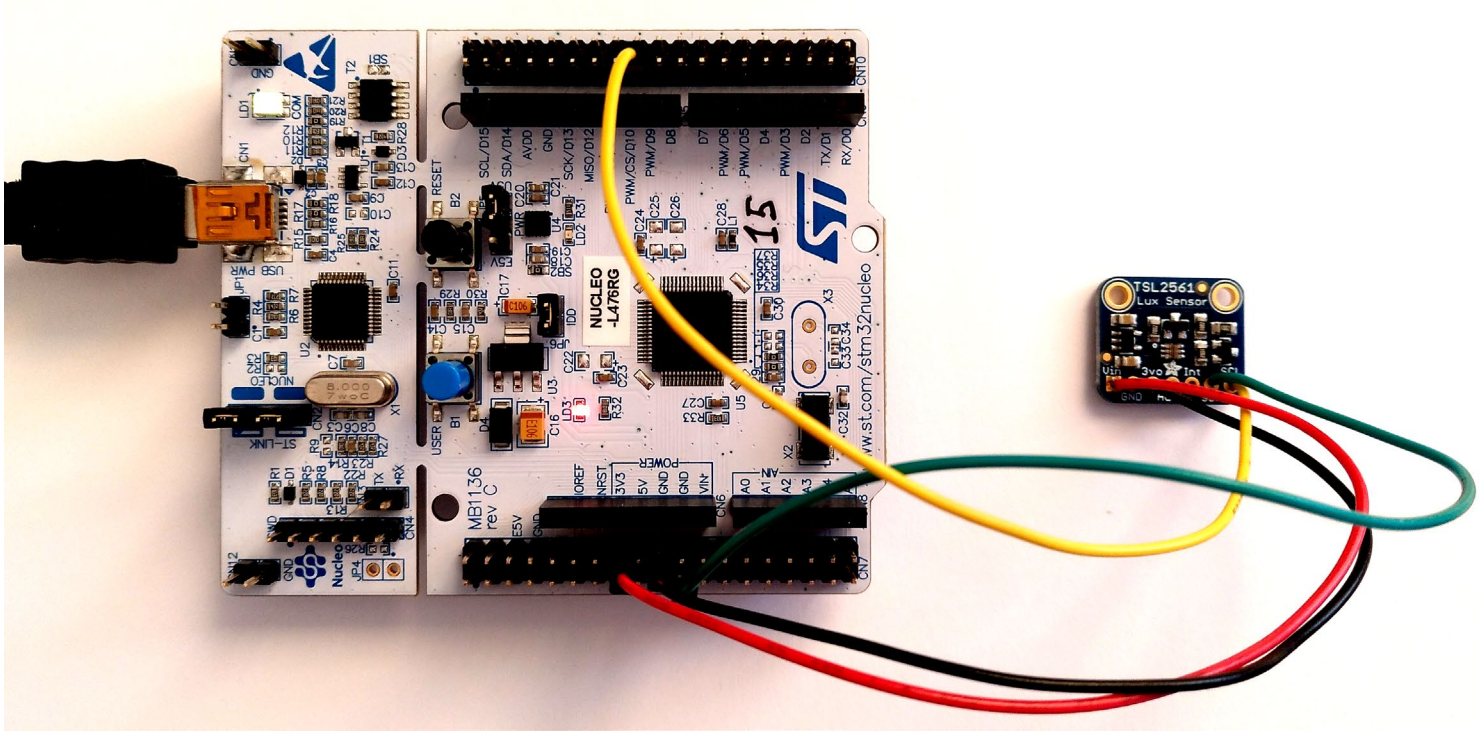
```
/* I2C1 init function */
static void MX_I2C1_Init(void)
{

  hi2c1.Instance = I2C1;
  hi2c1.Init.Timing = 0x10909CEC;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure Analogue filter
    */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1, I2C_ANALOGFILTER_ENABLE) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
```

Ejemplo con el AMS TSL2561

- Montamos el módulo …
  - Vin a 3V3 (CN7 pin 16), GND a GND (CN7 pin 20)
  - SCL a PB6 (CN10 pin 17), SDA a PB7 (CN7 pin 21)



Ejemplo con el AMS TSL2561

- Primera prueba ejemplo

```
#define I2C_ADDRESS 0x39
#define RXBUFFERSIZE 2

static I2C_HandleTypeDef hi2c1;
static uint8_t aRxBuffer[RXBUFFERSIZE];
```

```
while (1)
{
   HAL_Delay(1000);

   if (HAL_I2C_Master_Receive(&hi2c1, ((uint16_t)I2C_ADDRESS)<<1,
       (uint8_t *)aRxBuffer, RXBUFFERSIZE, 10000) != HAL_OK)
   {
       if (HAL_I2C_GetError(&hi2c1) != HAL_I2C_ERROR_AF)
       {
          while(1);
       }
   }
}
```

Ejemplo con el AMS TSL2561

- Preguntas suponiendo que estáis en el nivel 2
  - ¿Cómo sé que funciona?
  - ¿Qué cambio para que no me engañe?
  - ¿Y como veo lo que estoy recibiendo?
  - Realmente ¿qué leo y escribo?
    - -> volver al origen -> datasheet TSL2561 -> ¿ahora se entiende la idea?

The End

(Cajón de sastre)

- Bip, bip