Curso de Sensores en Plataforma Arduino

Departamento de Electrónica

Introducción al protocolo I2C

D. Julio Pastor Mendoza (pastor@depeca.uah.es)
D. Pedro Revenga de Toro (revenga@depeca.uah.es)
Profesores del Departamento de Electrónica (UAH)

Universidad de Alcalá

Índice de la Lección

- Introducción
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

Índice de la Lección

- Introducción
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

Introducción

Interconexión de dispositivos mediante bus serie

Ventajas

- Pocos cables de interconexión
- Componentes con encapsulado reducido
- Tarjetas reducidas
- Conexión de dispositivos a distancia

Inconvenientes

- Velocidad inferior a un bus paralelo
- Disponibilidad de circuitos que soporten el bus

Parámetros generales

- Número de hilos de conexión
- Velocidad (bits/segundo)
- Distancia máxima y número de dispositivos
- Protocolo de acceso al medio compartido
- Política de direccionamiento

Introducción

Origen del bus I²C (Inter Integrated Circuits Bus)

- Desarrollado por Philips a principios de los 80 como medio de interconexión entre una CPU y dispositivos periféricos dentro de la electrónica de consumo.
  - Simplificar las conexiones entre los periféricos (pistas, decodificadores, ..)
  - Aumentar de la inmunidad al ruido
  - Control de sistemas de audio y vídeo (baja velocidad)
- Actualmente diseñan dispositivos basados en I²C muchos fabricantes
  - Xicor, SGS-Thomson, Siemens, Intel, TI, Maxim, Atmel, Analog Devices

Protocolo de Comunicación I^2C

Índice de la Lección

- Introducción
- Características del bus I^2C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I^2C
- Conclusiones
- Bibliografía

Características del bus I²C

- Bus de comunicación síncrono
  - La comunicación es controlada por una señal de reloj común
- Bus formado por 2 hilos
  - SDA (***S****erial* ***DA****ta Line*): datos
  - SCL (***S****erial* ***CL****ock line*): reloj
  - También es necesaria una referencia común de masa
- Velocidad de transmisión
  - *Standard*: hasta 100 Kbits/s
  - *Fast*: hasta 400 Kbits/s
  - *High-speed*: hasta 3,4 Mbits/s
- Cada dispositivo del bus tiene una dirección única
- Distancia y número de dispositivos
  - Limitado por la capacidad del bus (inferior a 400pF). Normalmente 2 o 3 metros
- Protocolo de acceso al bus:
  - Maestro – esclavo
  - I²C soporta protocolo multimaestro

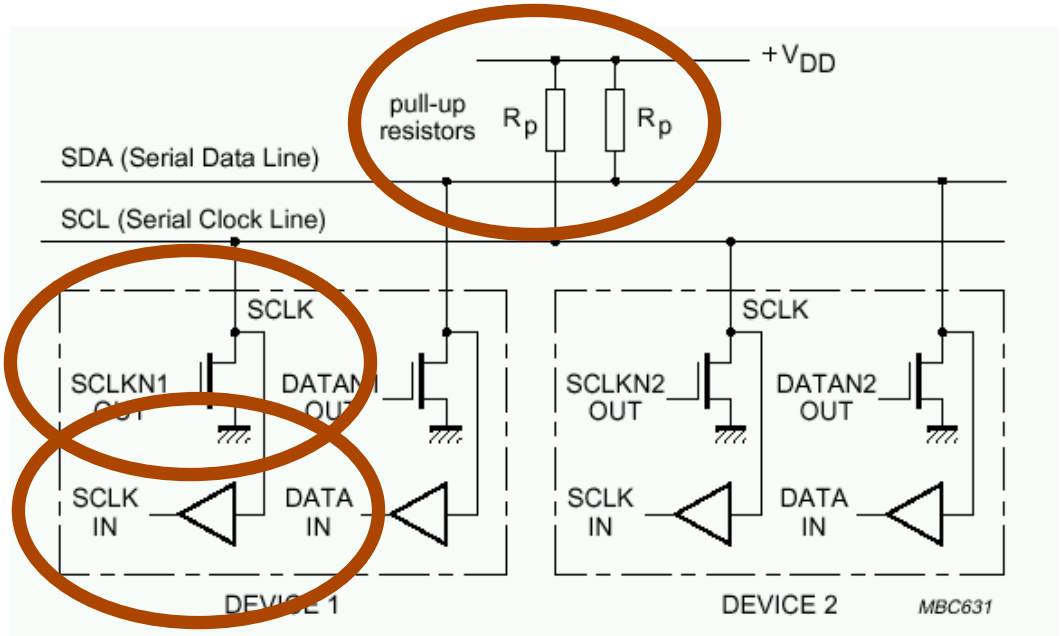
Índice de la Lección

- Introducción
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

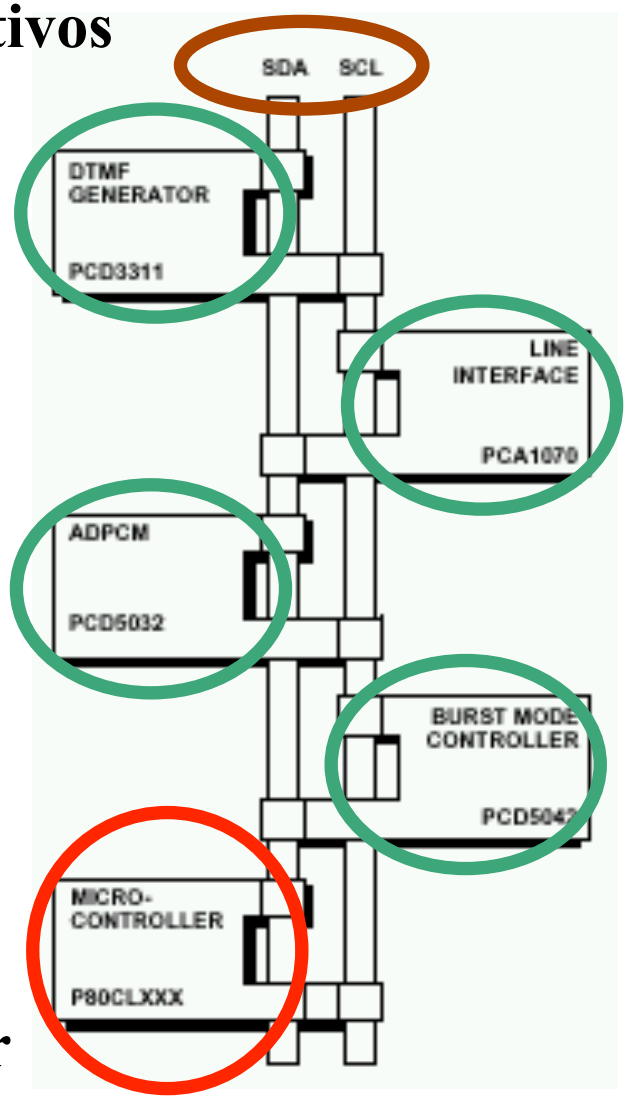
Conexión de dispositivos al bus: nivel físico

Conexión en bus

- **Todos los dispositivos conectados a las mismas líneas**
- Las salidas deben ser en colector o drenador abierto



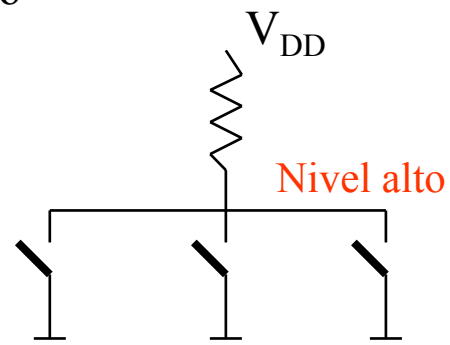
Dispositivos



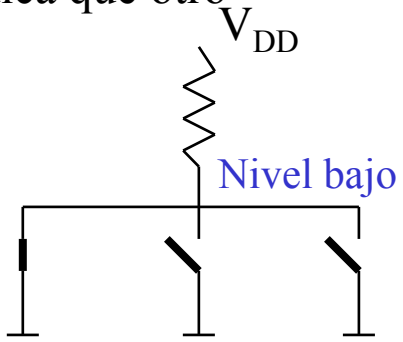
Microcontrolador

Características de una conexión en colector abierto

- Permite conectar varias fuentes de datos a un mismo hilo
- Nivel alto en el bus
  - Si ningún dispositivo accede al bus
  - Si ningún dispositivo transmite un cero
- Nivel bajo en el bus
  - Si un dispositivo pone un nivel bajo

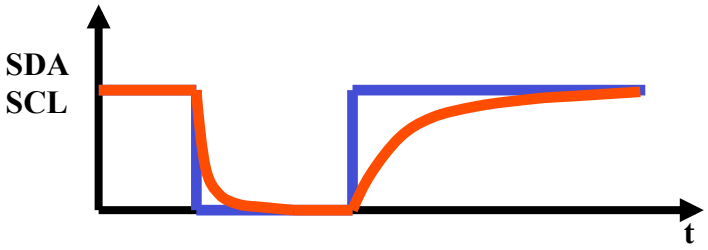


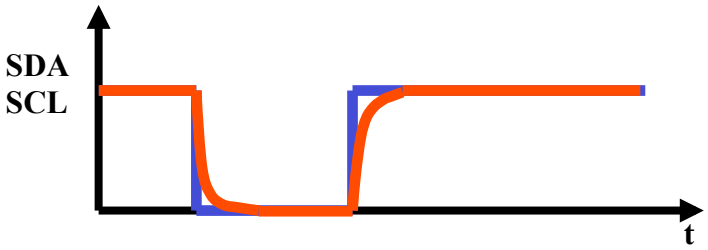
- Si dos dispositivos escriben a la vez siempre prevalecen los ceros
  - AND cableada
- Si un dispositivo escribe un nivel alto pero lee un cero indica que otro dispositivo está también accediendo al bus

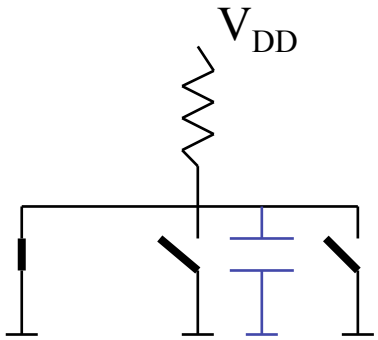


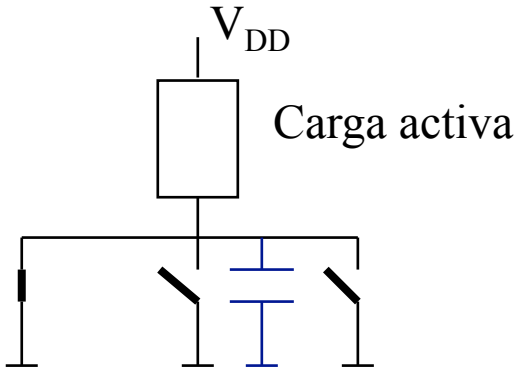
Inconvenientes de la conexión en colector abierto

- Las capacidades de la línea se cargan a través del *pull-up*
- Se puede solucionar utilizando una carga activa en lugar de un resistor









Protocolo de Comunicación I²C

Índice de la Lección

- Introducción a la comunicación entre dispositivos mediante un bus serie
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

Protocolo de acceso al medio: maestro - esclavo

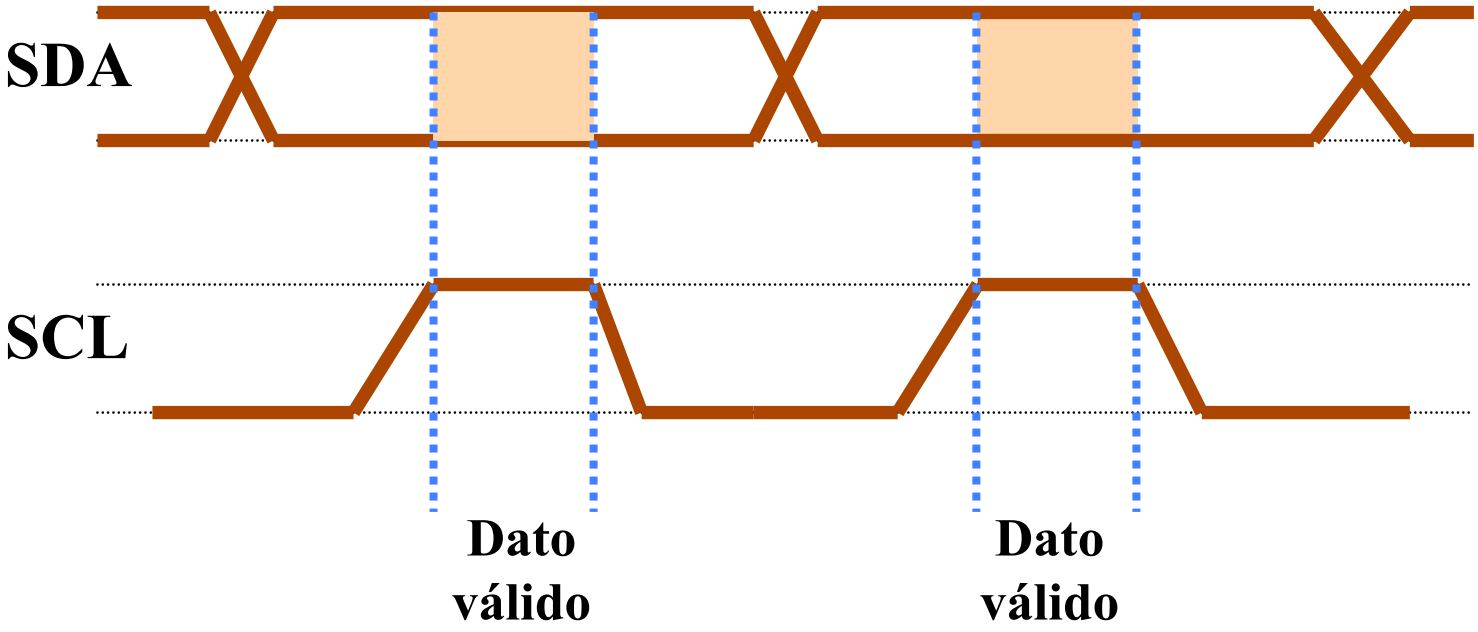
- El maestro controla la comunicación
  - Genera la señal de reloj del bus (SCL)
  - Inicia y termina la comunicación
  - Direcciona a los esclavos
  - Establece el sentido de la comunicación
- El protocolo requiere que cada byte de información sea confirmado por el destinatario

Nomenclatura

- Emisor: Dispositivo que envía datos al bus
- Receptor: Dispositivo que recibe datos del bus
- Maestro: Dispositivo que inicia una transferencia, genera las señales de reloj y termina la transferencia
- Esclavo: Dispositivo direccionado por un maestro

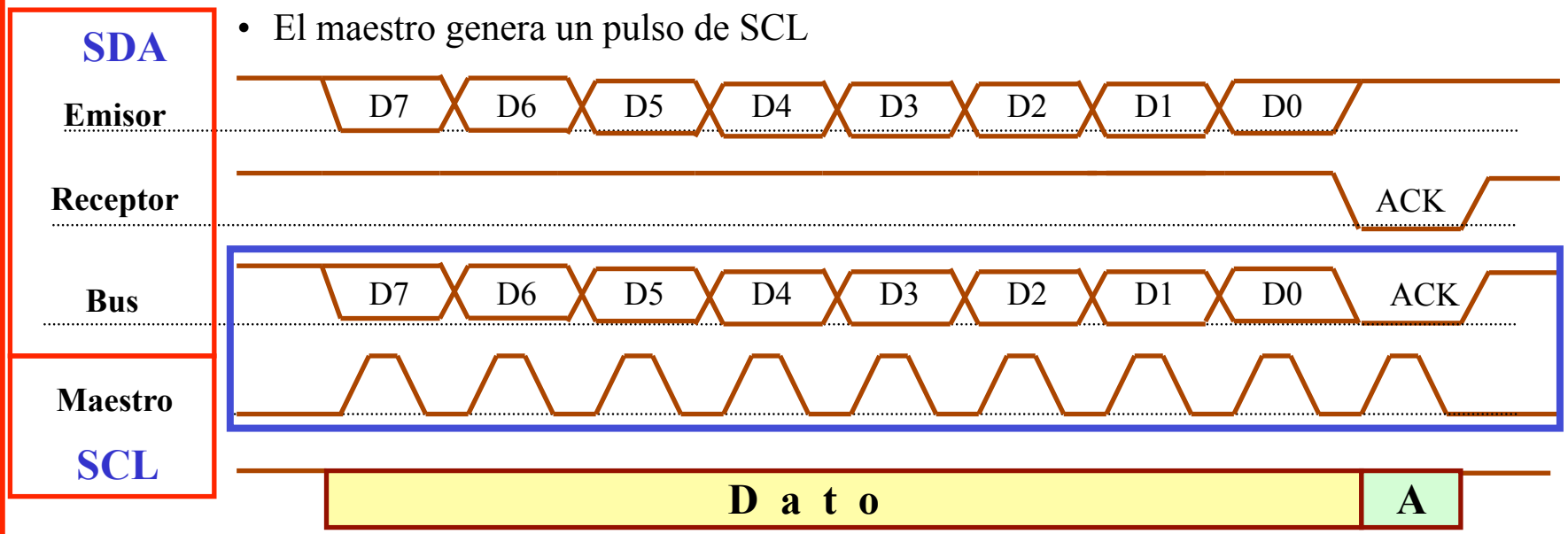
Transmisión de bits

- Los bits de datos van por SDA
- Por cada bit de información es necesario un pulso de SCL
- Los datos sólo pueden cambiar cuando SCL está a nivel bajo



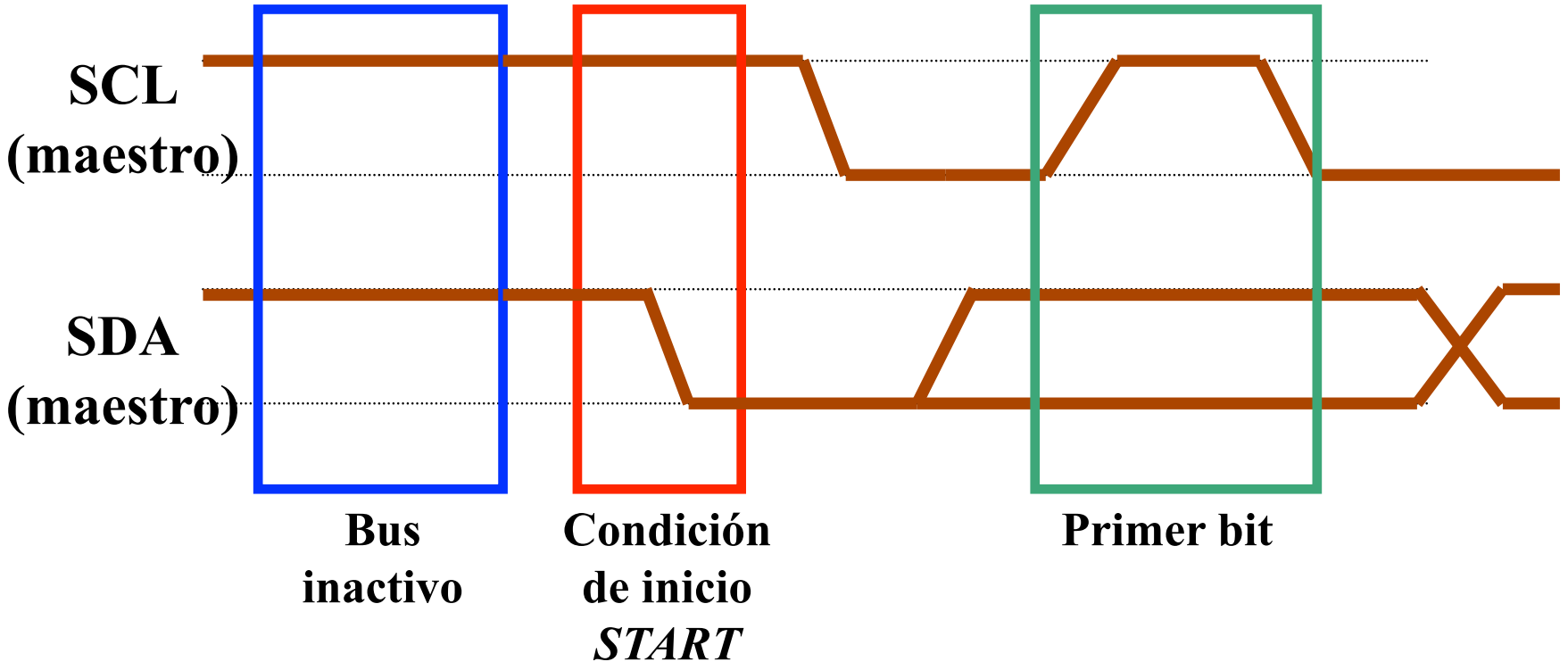
Transmisión de datos

- **La unidad básica de transmisión en el byte**
- **Las transferencias de datos son de 8 bits**
- **Cada byte enviado requiere una respuesta de confirmación**
  - ACK: el destinatario (maestro o esclavo) mantiene SDA a nivel bajo durante un tiempo de bit (si no lo hace → NACK)
  - El maestro genera un pulso de SCL



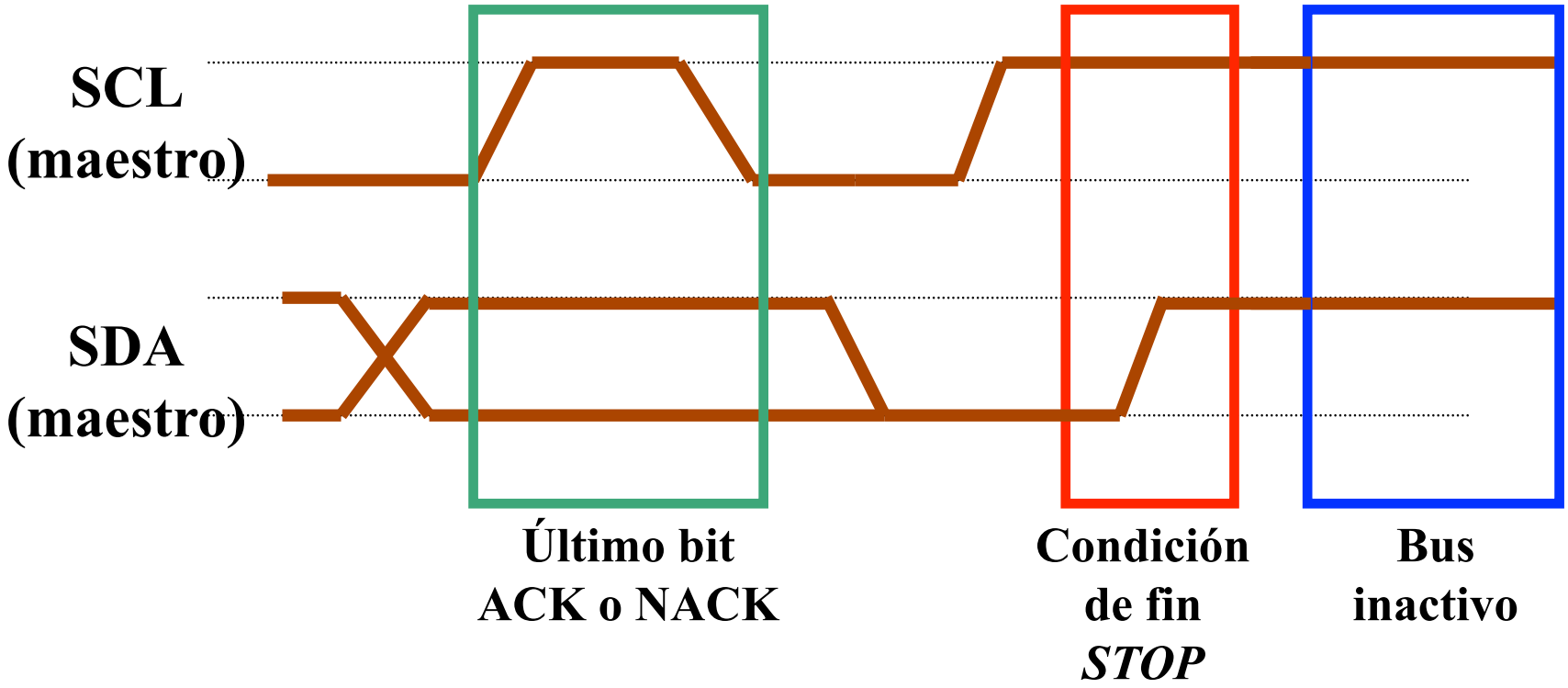
Inicio de transmisión

- La transmisión la inicia el maestro
- Flanco de bajada en SDA con SCL a nivel alto
- Cuando nadie accede al bus hay un nivel alto en SCL y SDA



Finalización de transmisión

- La transmisión la finaliza el maestro
- Flanco de subida en SDA con SCL a nivel alto

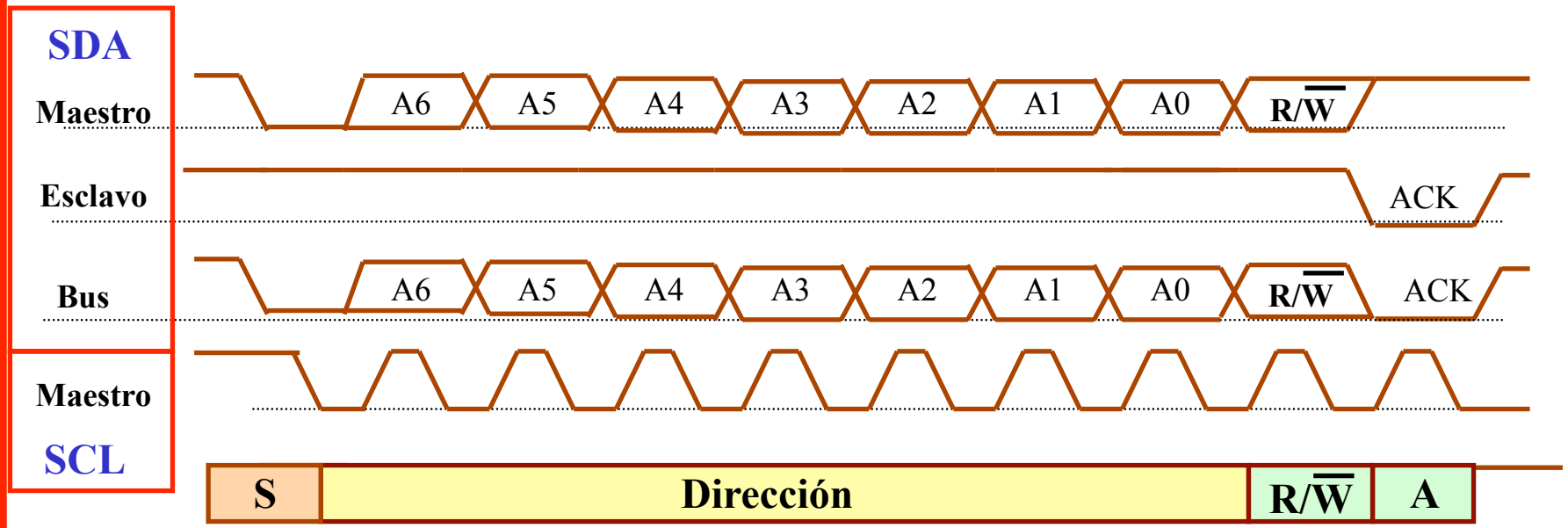


Intercambio de datos

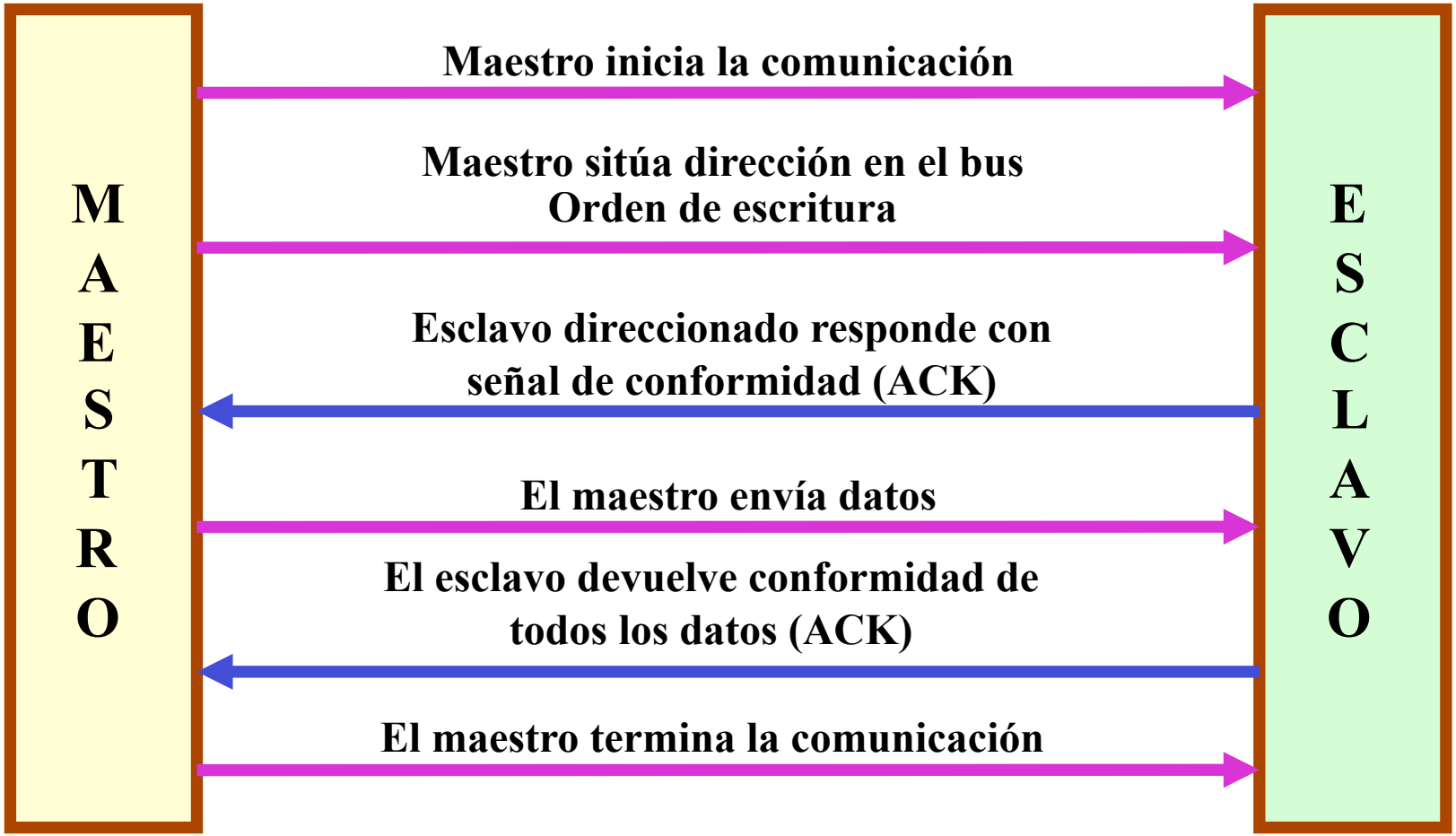
Direccionamiento

– **Tras la condición de inicio el maestro envía:**

- Dirección del esclavo (7 bits)
- Comando de lectura o escritura (R=1 – W=0)



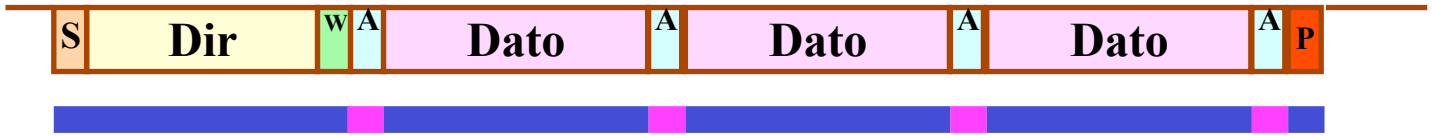
Maestro envía datos a un esclavo



Intercambio de datos

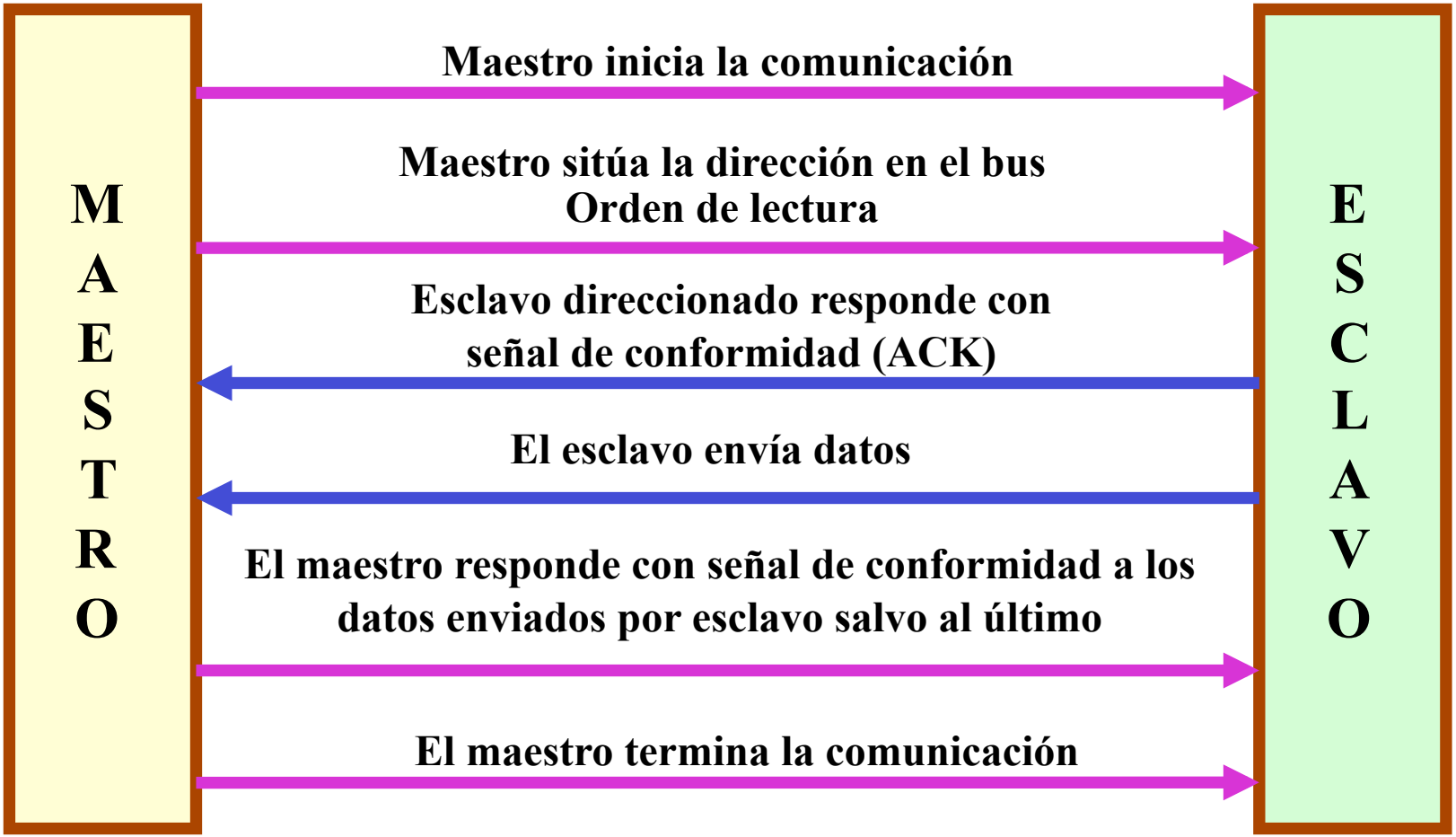
Maestro envía un dato a un esclavo

Maestro envía varios datos a un esclavo



BUS I2C

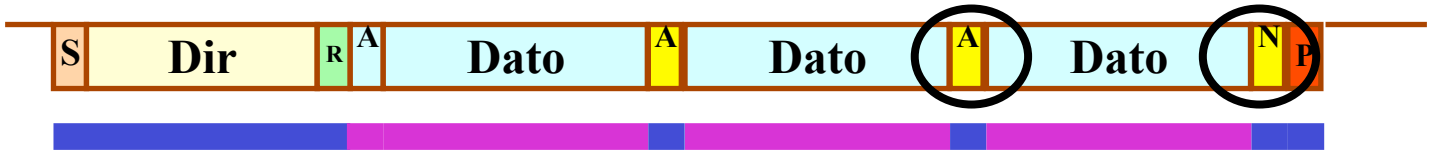
Maestro lee datos de un esclavo



Intercambio de datos

Maestro lee un dato de un esclavo

Maestro lee varios datos de un esclavo



Intercambio de datos

Maestro escribe y lee cambiando de dirección

Casos particulares de respuesta

- Esclavo ocupado y no responde al ACK de dirección
  - Maestro genera condición de *STOP*
- El esclavo quiere interrumpir una recepción periódica de datos
  - El esclavo no responde a un dato con *ACK*
  - El maestro genera la condición de *STOP*
- El maestro quiere interrumpir una recepción de datos de un esclavo
  - El maestro no responde con *ACK*
  - El esclavo deja de transmitir
  - El maestro genera condición de *STOP*

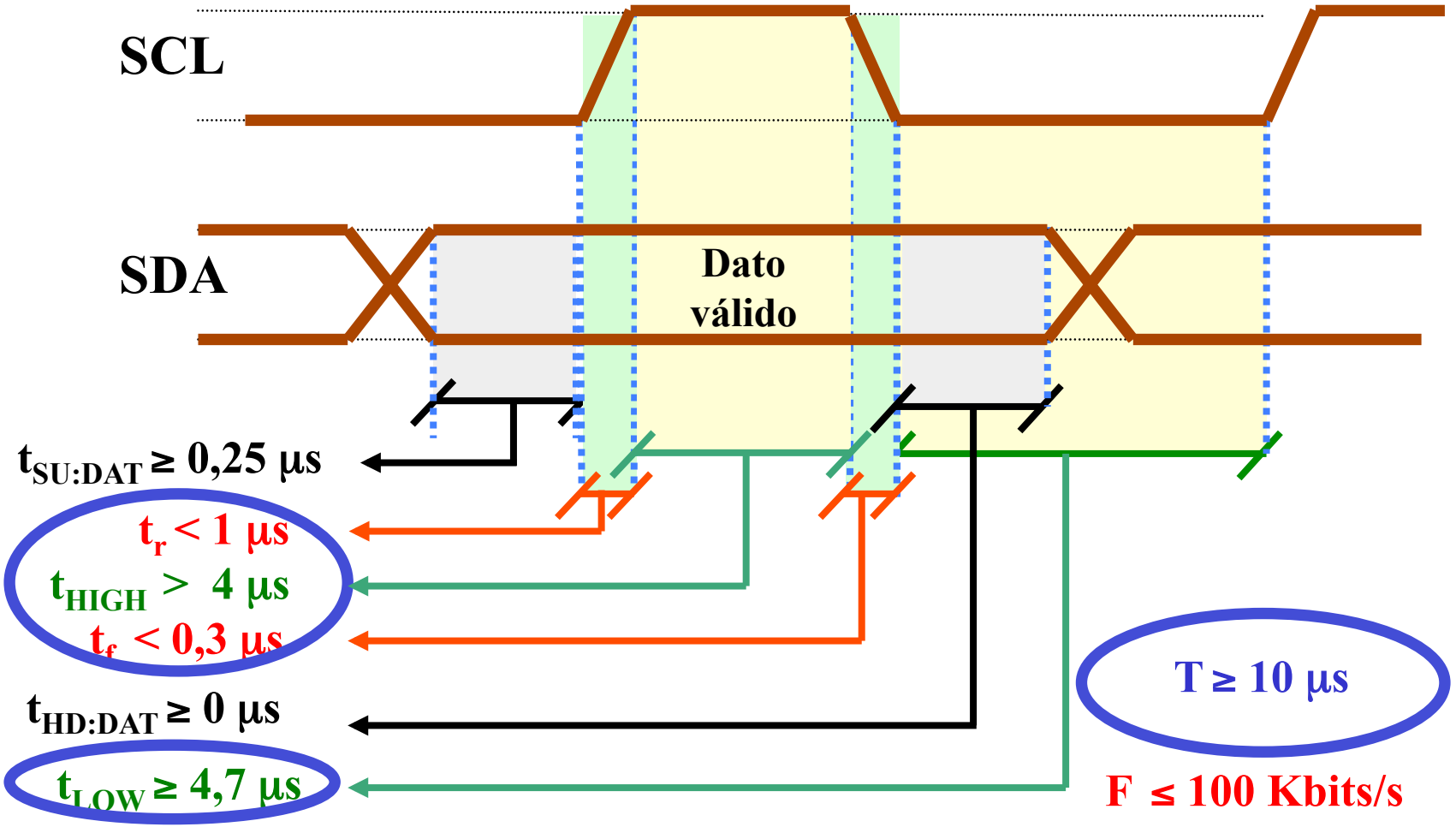
Protocolo multimaestro

- **Con dos maestros en el bus existe posibilidad de conflicto**

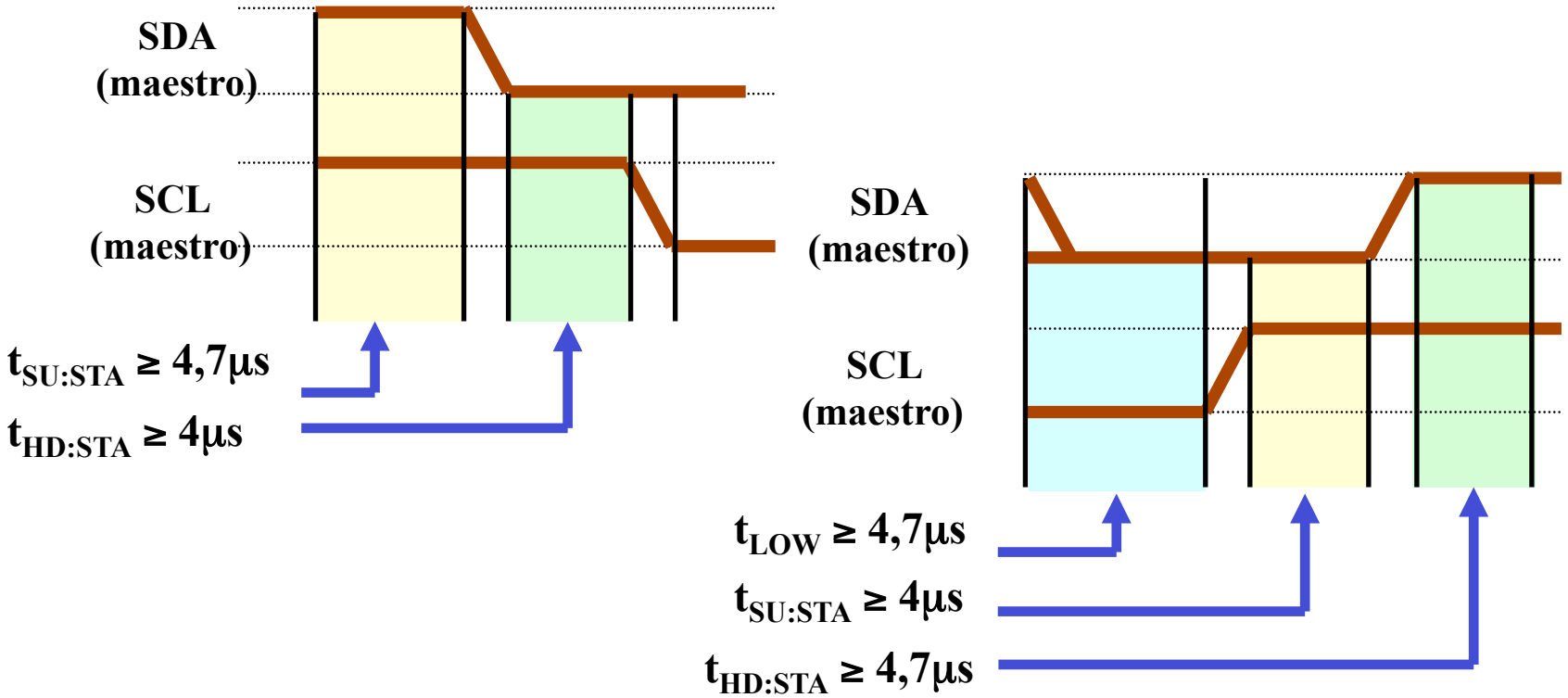
Arbitración: procedimiento para asegurar que sólo un maestro tiene el control del bus en un instante

- **Si un maestro está utilizando el bus no puede ser interrumpido por otro.**
  - Desde *START* hasta *STOP*
- **Si dos maestros intentan comenzar a utilizar el bus a la vez:**
  - Conexión del bus en colector abierto → prevalecen los ceros
  - A la vez que ponen datos en el bus, escuchan la línea.
  - Si un maestro está intentando enviar un nivel alto y lee un nivel bajo
    - Existe otro maestro utilizando el bus
    - Deja de transmitir esperando que la línea quede libre (condición de *STOP*)

Transmisión de bits: Temporización



Temporización en *START* y en *STOP*



Protocolo de Comunicación I²C

Índice de la Lección

- Introducción a la comunicación entre dispositivos mediante un bus serie
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

Control del bus I2C desde un microcontrolador

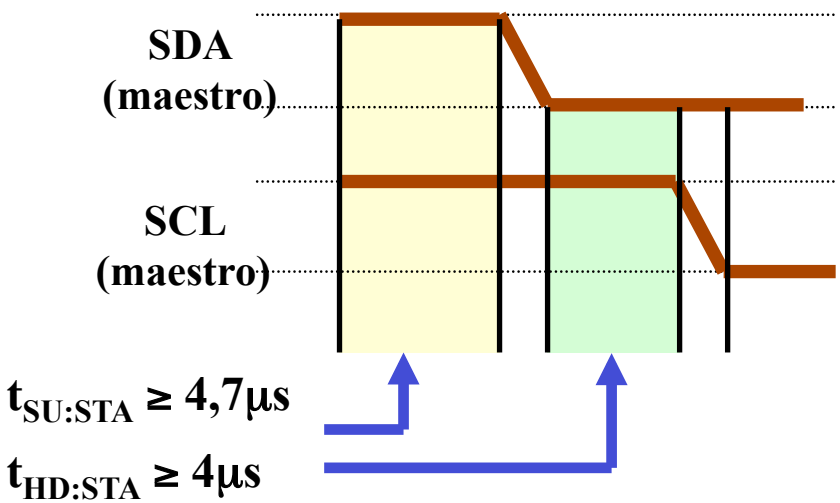
- **Sin una unidad hardware específica**
  - El protocolo I2C se puede controlar utilizando puertos de entrada y salida
  - El maestro controla la velocidad del bus
    - No debe generar señales demasiado rápidas
    - Puede ralentizar el bus cuando quiera
  - Complicado utilizarlo como esclavo.
    - Debería cumplir la norma de tiempos
    - Sólo procesadores muy rápidos o dedicados.
  - No se permite multimaestro
- **Con una unidad hardware específica**
  - El procesador no debe preocuparse de gestionar el protocolo
  - Se puede aprovechar toda la funcionalidad del bus

Control del bus I2C utilizando puertos de E/S

- **Eventos a programar**
  - Generación de *START*
  - Generación de *STOP*
  - Enviar dirección o dato y recibir *ACK*
  - Recibir dato y enviar *ACK* o *NACK*
- ***Consideraciones temporales***
  - Velocidad del procesador
    - El reloj del 8051 típico se puede considerar 12 MHz
    - Ciclo máquina en modo X1: 1 us.
    - Duración media de una instrucción: 2 ciclos (2 µs)

Control del bus I2C utilizando puertos de E/S

I2C_START

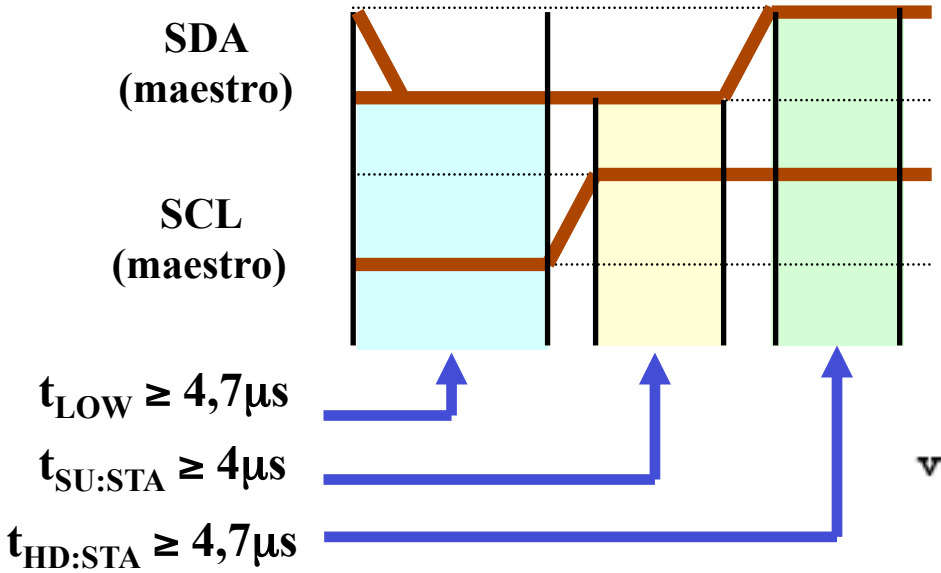


Pseudocódigo

SDA = 1
SCL = 1
delay(4,7µs)
SDA = 0
delay(4µs)
SCL = 0

Control del bus I2C utilizando puertos de E/S

I2C_STOP



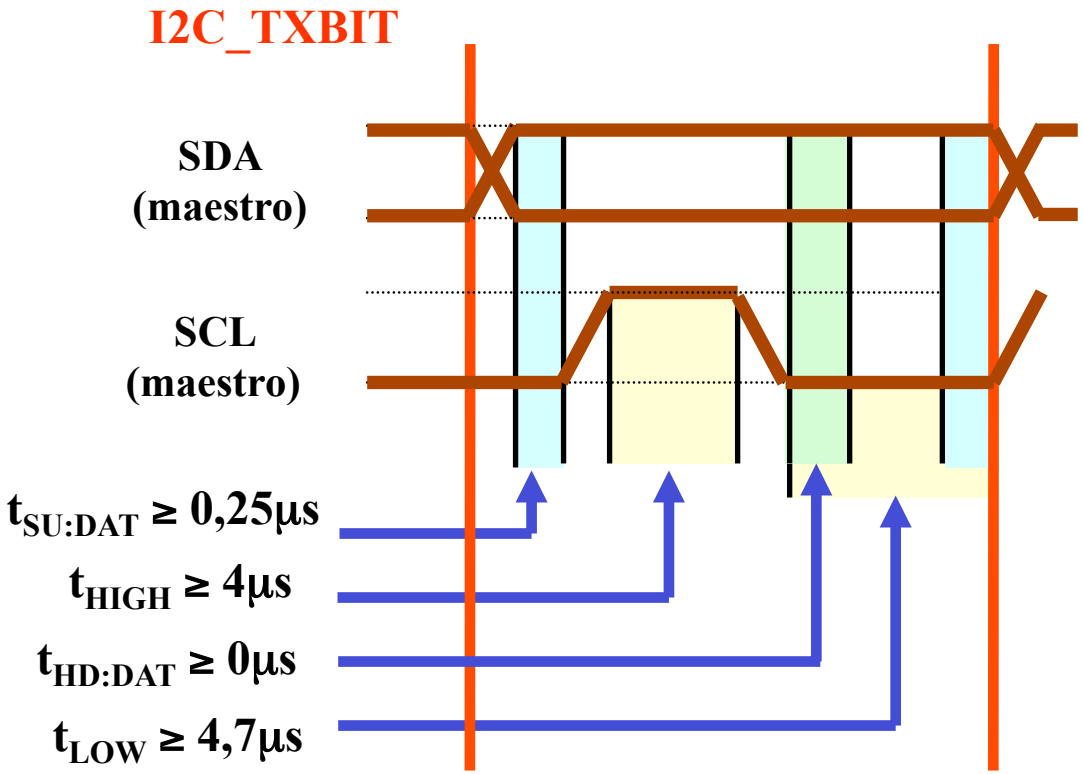
Pseudocódigo

SDA = 0
SCL = 0
delay(4,7µs)
SCL = 1
delay(4µs)
SDA = 1
delay(4,7µs)

```
void I2CSendStop(void)
        {
        SDA = 0;
        I2Cdelay();
        SCL=1;
        I2Cdelay();
        SDA = 1;
        I2Cdelay();
        }
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

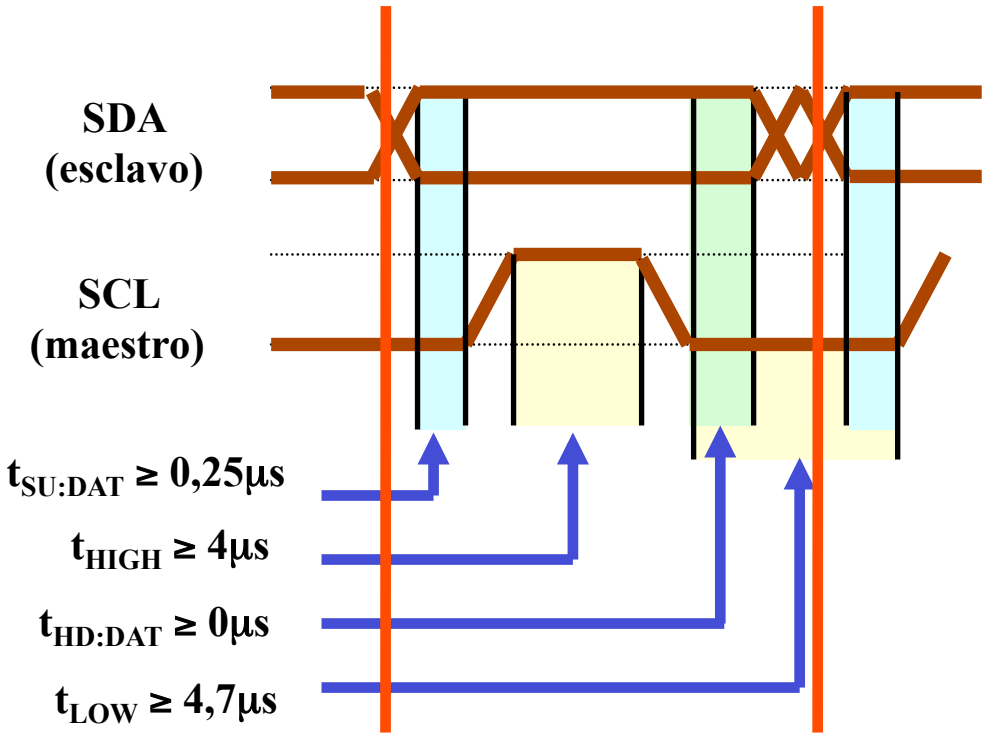


Pseudocódigo

```
SDA = BIT
SCL = 1
delay(4µs )
SCL = 0
delay(4,7µs )
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

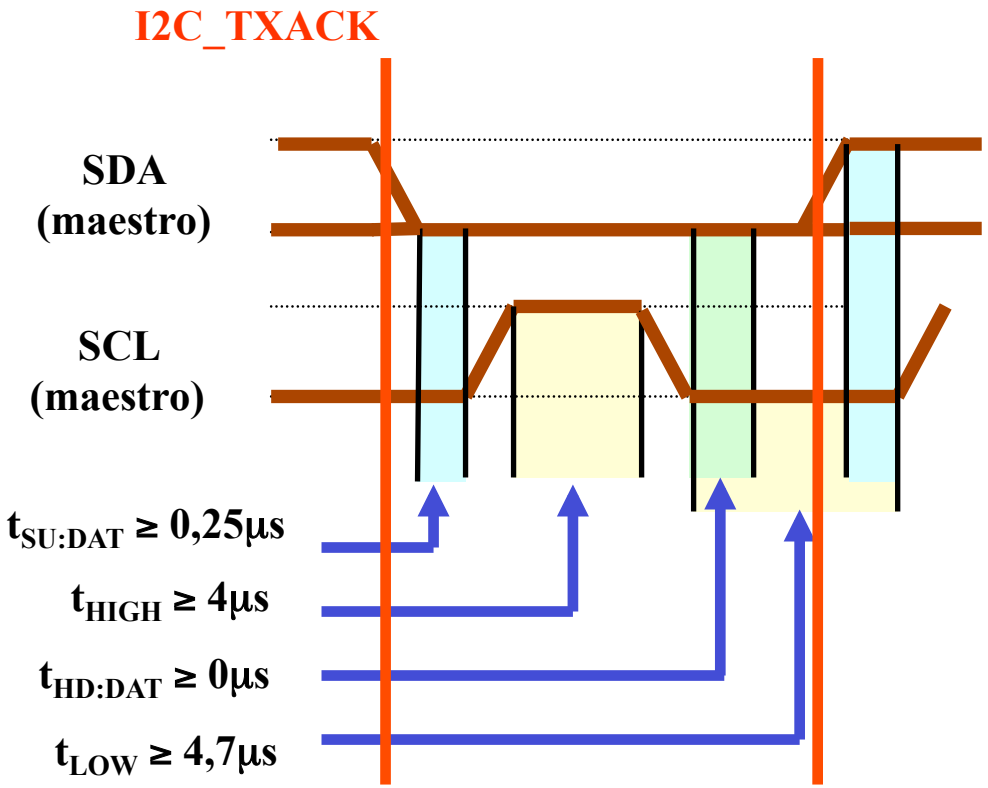


Pseudocódigo

```
SCL = 1
delay(4µs )
BIT = SDA
SCL = 0
delay(4,7µs )
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S



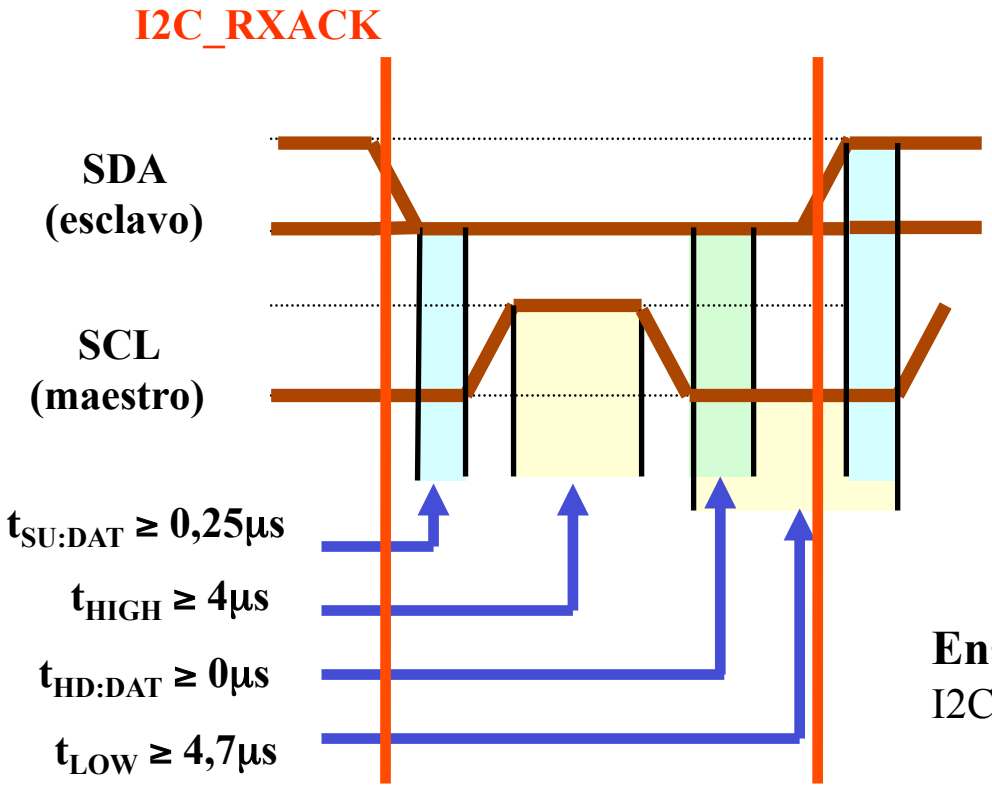
Pseudocódigo

SDA = 0
SCL = 1
delay(4µs)
SCL = 0
delay(4,7µs)

Generación del protocolo desde un µC

BUS I2C

Control del bus I2C utilizando puertos de E/S



Pseudocódigo

SCL = 1
delay(4µs)
BIT = SDA
SCL = 0
delay(4,7µs)

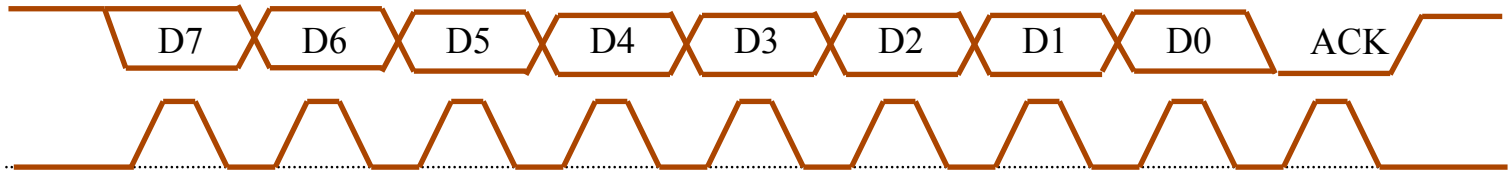
Ensamblador (ACK = carry)

I2C_RXACK BSR I2C_RXBIT
RTS

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_TXBYTE



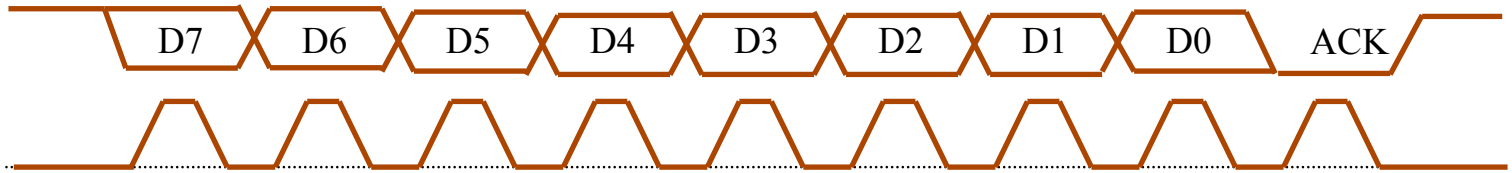
Pseudocódigo

Repite 8
    DESPLAZA DATO ←
    I2C_TXBIT
I2C_RXACK
Devuelve ACK

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_TXBYTE

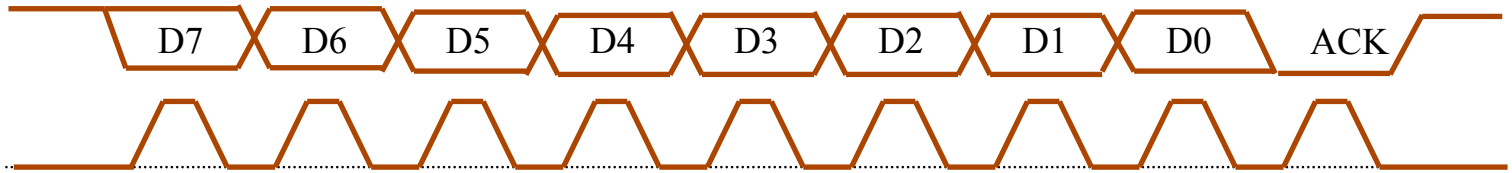



```
void I2CSendByte(unsigned char byte)
      {
      unsigned char i;
      for (i=0; i<8; i++)
            {
            if (byte & 0x80) SDA = 1;//envia cada bit, comenzando por el MSB
            else SDA = 0;
            SCL=1;
            I2Cdelay();
            SCL = 0;
            I2Cdelay();
            byte = byte << 1;
            }

      SDA = 1;                        // espera ACK (config. pin como lectura)
      SCL=1;
      I2Cdelay();
      SCL = 0;
      I2Cdelay();
      }
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_RXBYTE



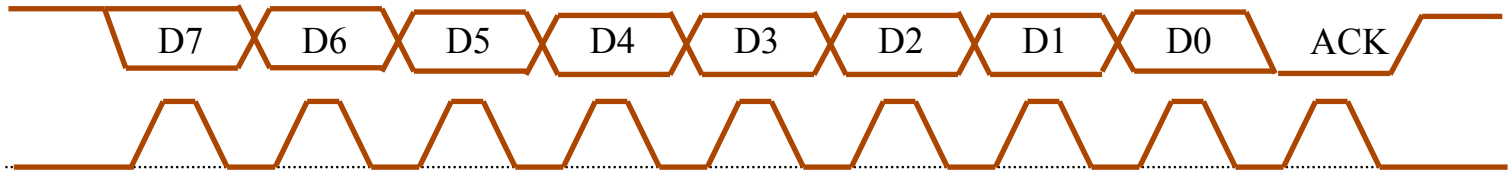
Pseudocódigo

Repite 8
    I2C_RXBIT
    DESPLAZA DATO ←
I2C_TXACK
Devuelve DATO

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_RXBYTE

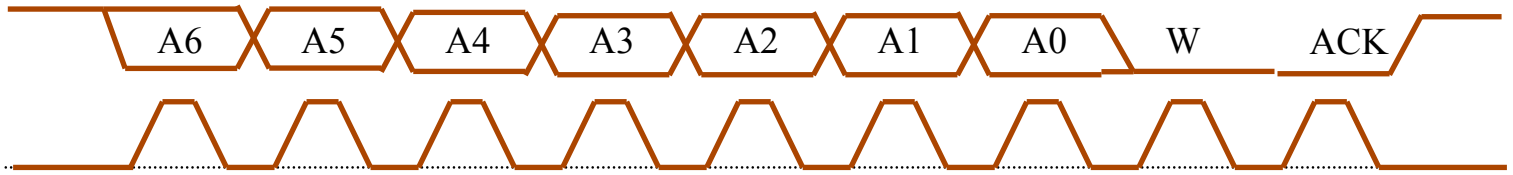



```
unsigned char I2CGetByte(unsigned char ultimo)
// ultimo = NACK = 1 para el ultimo byte
// ultimo = ACK = 0 para cualquier byte que no sea el último.
      {
      unsigned char i, byte;
       SDA=1;
       for (i=0;i<8;i++)            // lee un bit comenzando por el MSB
             {
             SCL=1;
             I2Cdelay();
             byte=byte<<1;
             if (SDA) byte++;
             SCL = 0;
             I2Cdelay();
             }
       SDA = ultimo;               //  ACK es funcion de ultimo
       SCL=1;
       I2Cdelay();
       SCL = 0;
       I2Cdelay();
       return(byte);
 }
```

Generación del protocolo desde un µC

BUS I2C

I2C_DIR_W



Pseudocódigo

A=DIR <<1 + W(0)
I2C_START
I2C_TXBYTE
Devuelve ACK

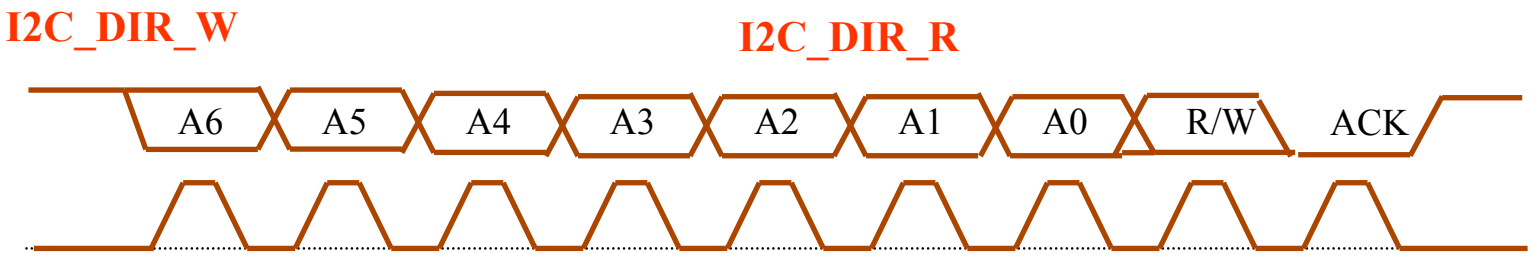
I2C_DIR_R

Pseudocódigo

A=DIR<<1 + R(1)
I2C_START
I2C_TXBYTE
Devuelve ACK

Generación del protocolo desde un µC

BUS I2C




```
void I2CSendAddr(unsigned char addr, unsigned char rw)
      {
            SDA = 1;
            SCL = 1;
            I2Cdelay();
            SDA = 0;                    // condicion de START
            I2Cdelay();
            SCL = 0;
            I2Cdelay();
            I2CSendByte((addr=addr<<1) + rw);   // envia byte de direccion
                                                 // addr, dirección (7 bits)
                                                 // rw=1, lectura
                                                 // rw=0, escritura
      }
```

Protocolo de Comunicación I²C

Índice de la Lección

- Introducción a la comunicación entre dispositivos mediante un bus serie
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

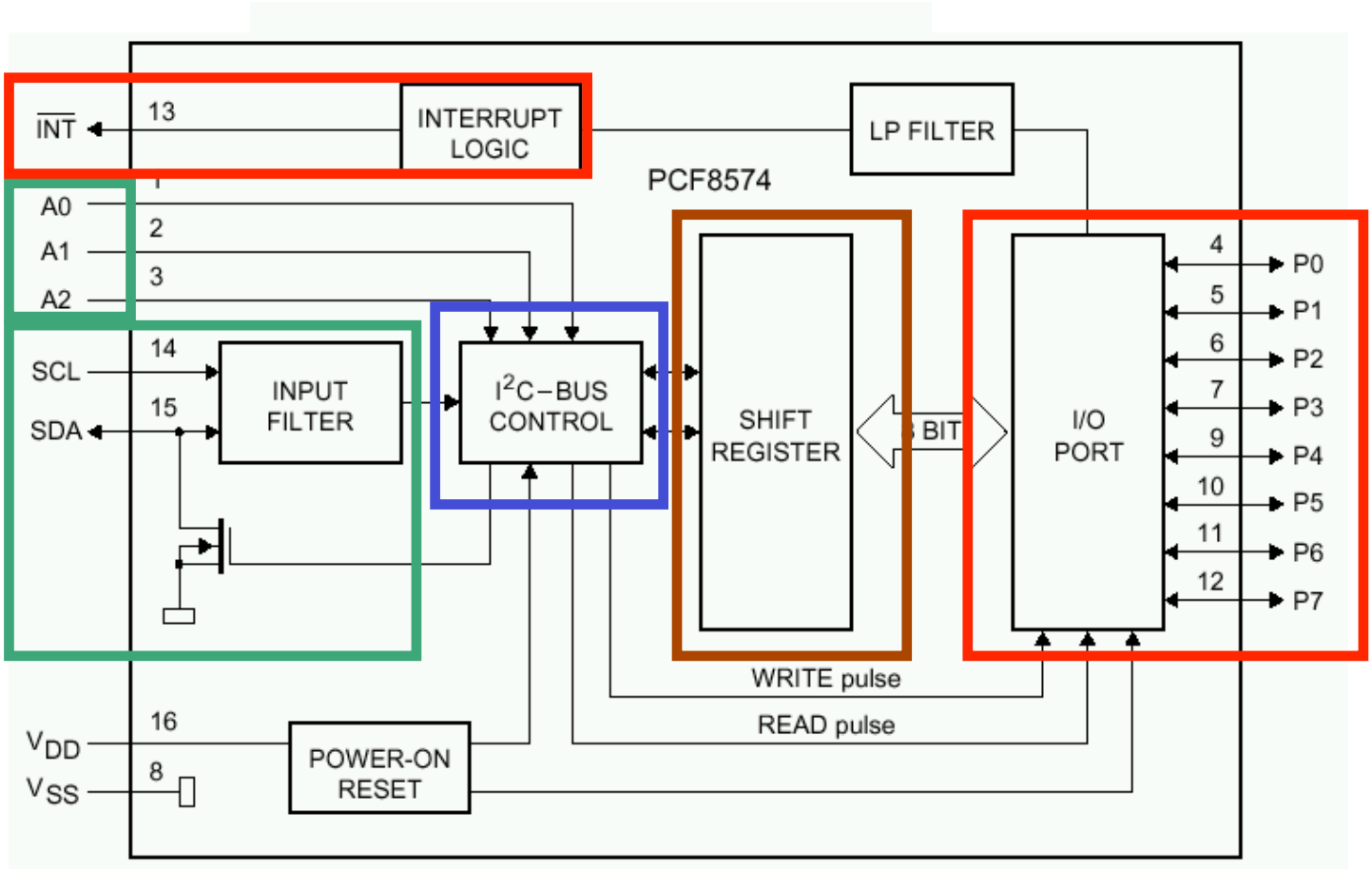
Ejemplos de dispositivos que utilizan I²C

Factores a tener en cuenta

- Los dispositivos tendrán dos partes.
  - Bloque funcional del dispositivo
  - Interfaz I²C
- Interfaz I²C
  - Direccionamiento
  - Acceso a la funcionalidad del dispositivo
    - En dispositivos sencillos – lectura y escritura de datos.
    - En dispositivos complejos
      - » Configuración de modo de funcionamiento
      - » Comando + dato
- Ejemplos de dispositivo
  - puerto de E/S de 8 bits
  - Memoria serie
  - Sensor de temperatura - termostato

Ejemplos de dispositivos que utilizan I²C

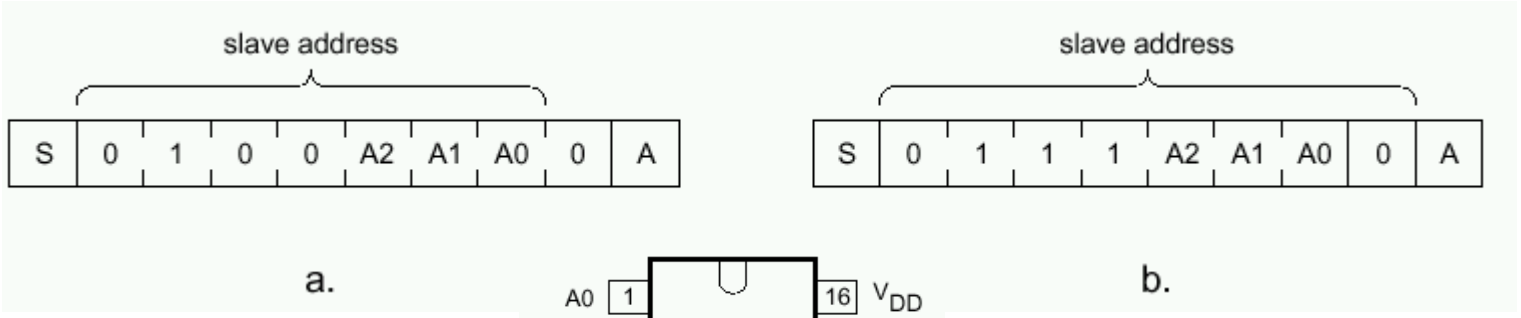
Puerto de 8 bits de entrada/salida – PCF8574

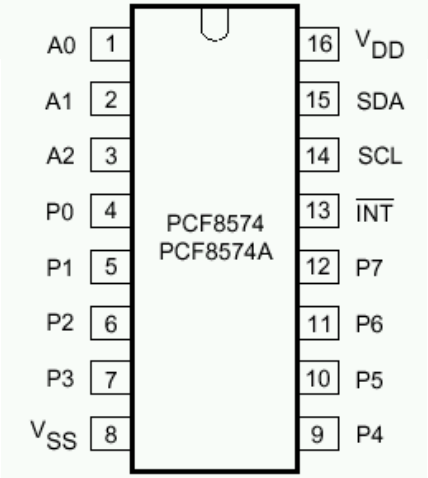


Puerto de 8 bits de entrada/salida – PCF8574

– **Direccionamiento**

- Parte de la dirección es fija. Depende del dispositivo
- Los tres bits de menor peso se configuran con pines del dispositivo

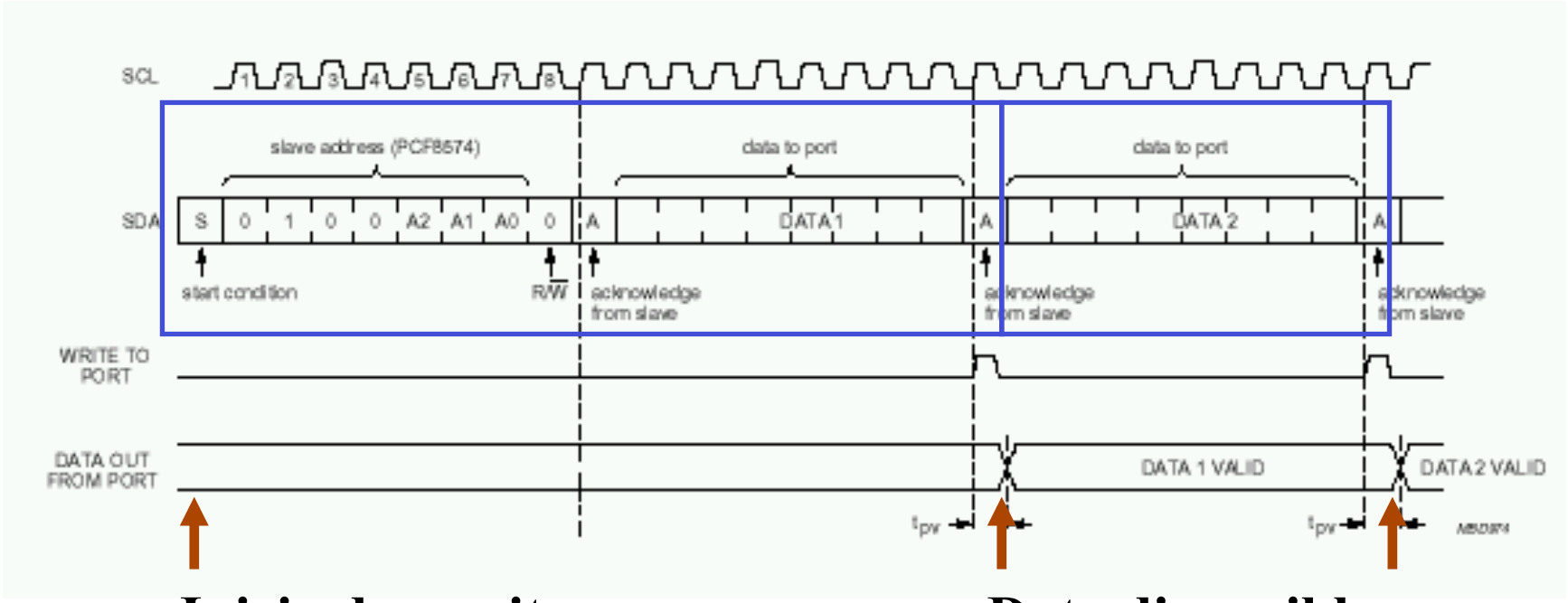




Ejemplos de dispositivos que utilizan I²C

Puerto de 8 bits de entrada/salida – PCF8574

– **Escritura en el puerto**

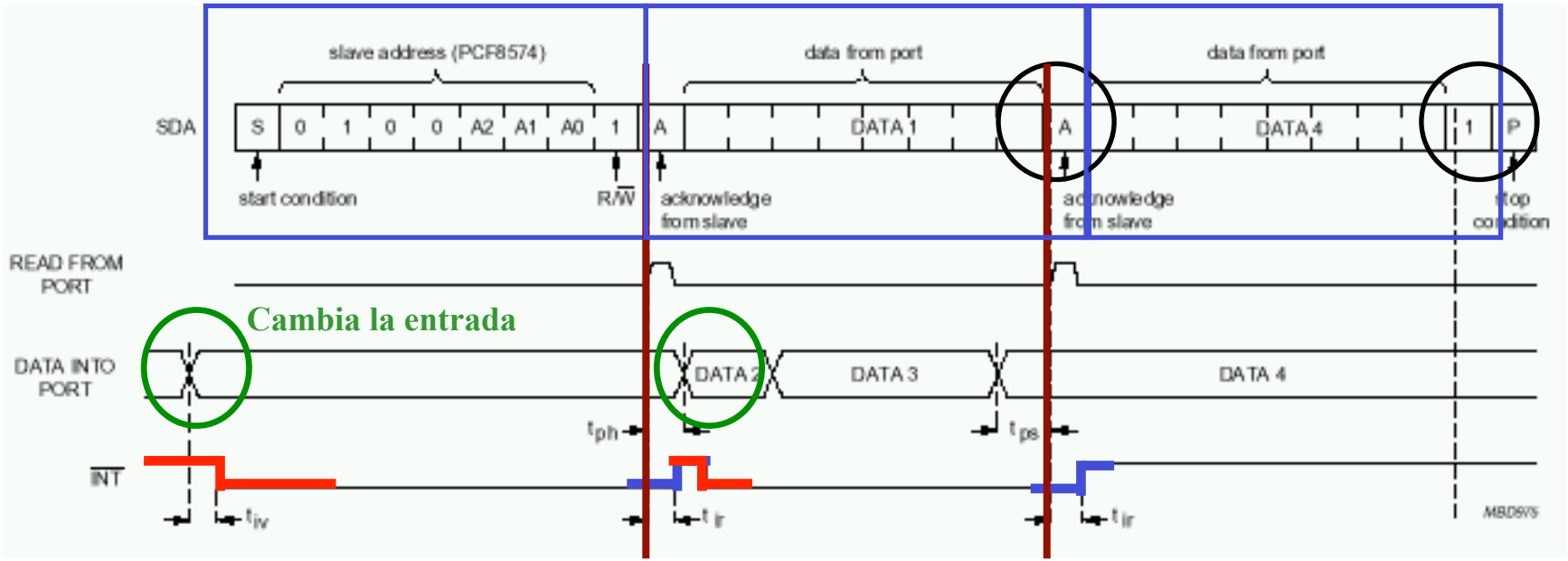


Inicio de escritura **Dato disponible**

Ejemplos de dispositivos que utilizan I²C

Puerto de 8 bits de entrada/salida – PCF8574

- **Lectura del puerto**
  - Si se modifica una entrada se activa una interrupción
  - La línea de interrupción tiene salida en colector abierto.



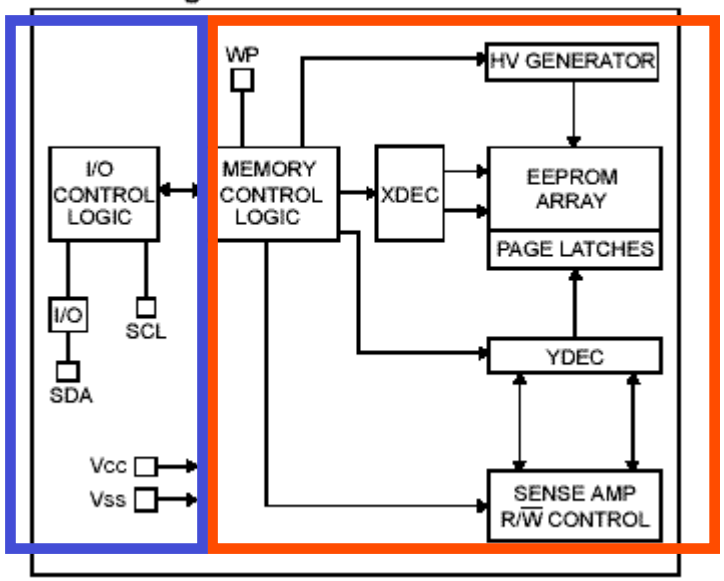
Ejemplos de dispositivos que utilizan I²C

Memoria serie

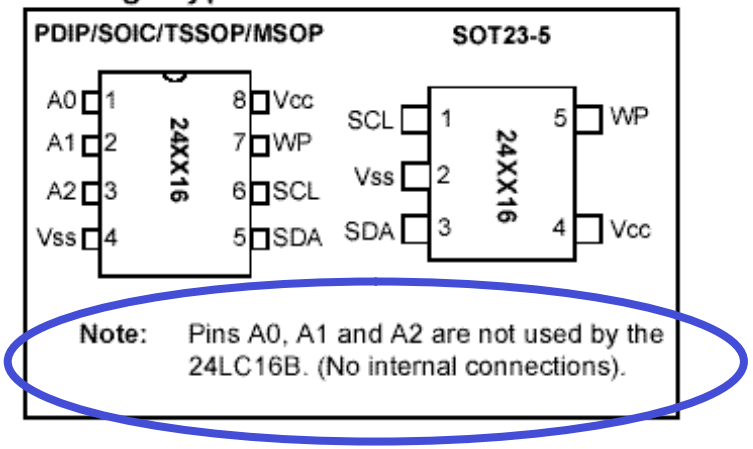
MICROCHIP **24AA16/24LC16B**

16K I²C™ Serial EEPROM

Block Diagram



Package Types



Memoria serie - Comando

Memoria de 16K bits (2KBytes)

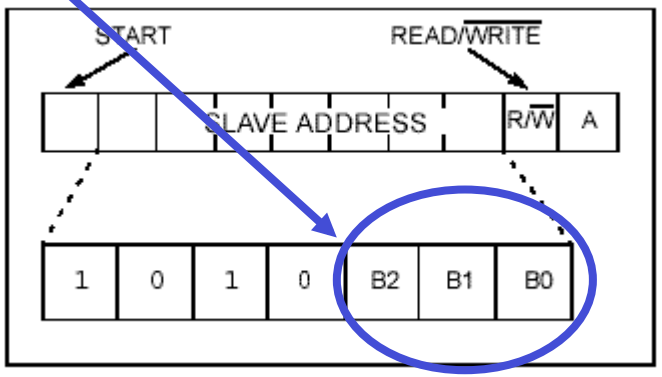
Organizada en 8 bloques de 256 bytes

Dirección de bloque (3 bits)

Dirección de byte (8 bits)

Lectura / escritura

Operation	Control Code	Block Select	R/$\overline{W}$
Read	1010	Block Address	1
Write	1010	Block Address	0



Memoria serie - Escritura

FIGURE 4-1: BYTE WRITE

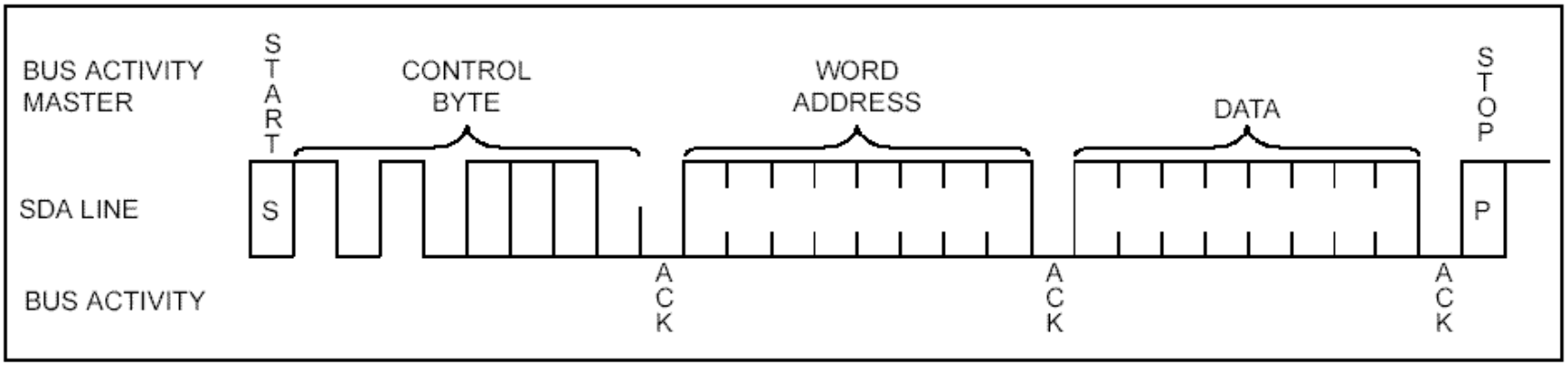
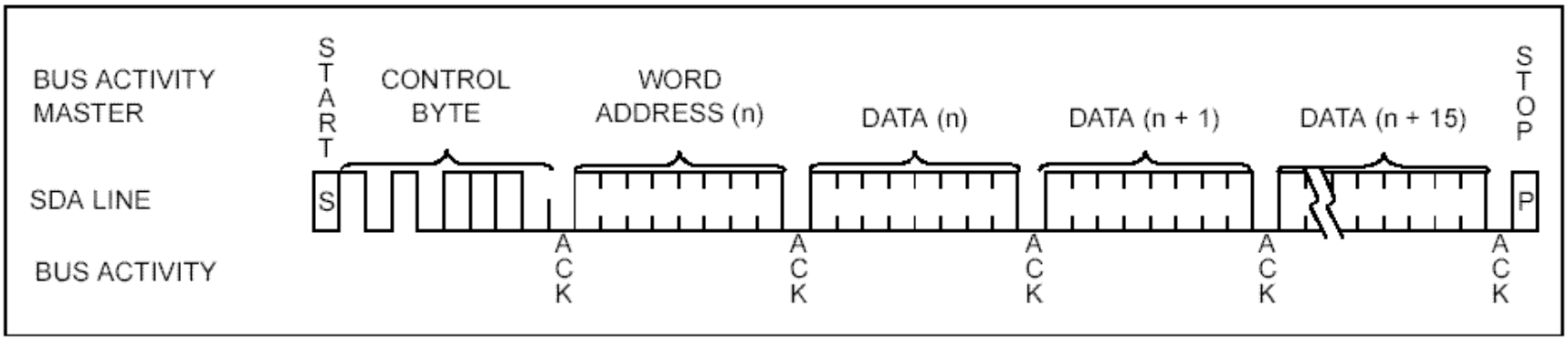


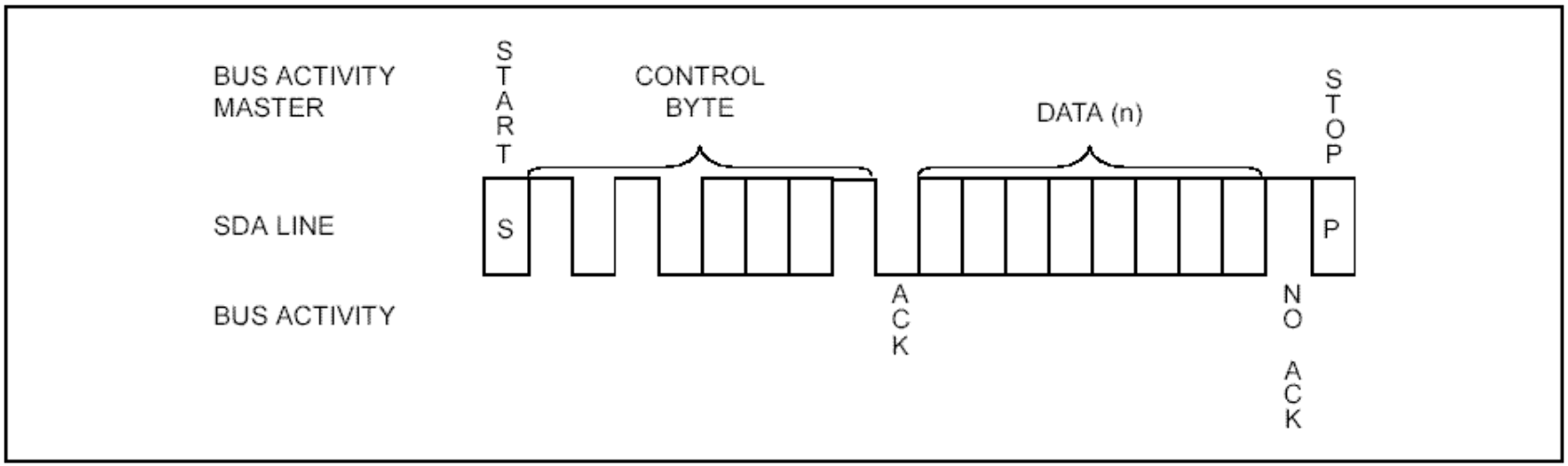
FIGURE 4-2: PAGE WRITE



Ejemplos de dispositivos que utilizan I²C

Memoria serie - lectura

FIGURE 7-1: CURRENT ADDRESS READ



Con cada lectura se autoincrementa un puntero interno

Ejemplos de dispositivos que utilizan I²C

BUS I2C

Memoria serie - lectura

FIGURE 7-2: RANDOM READ

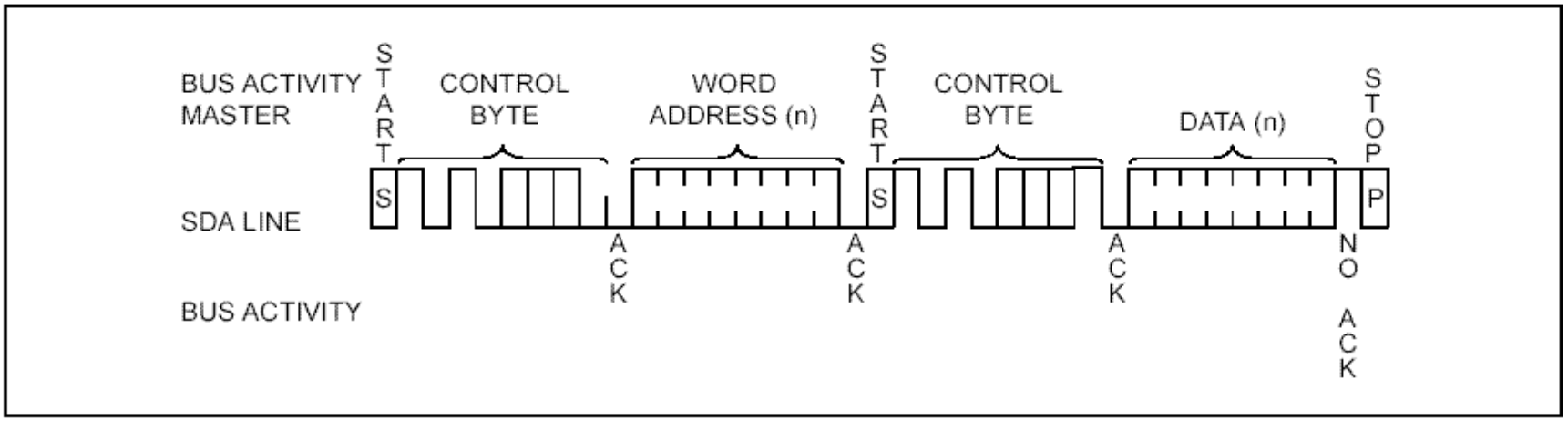
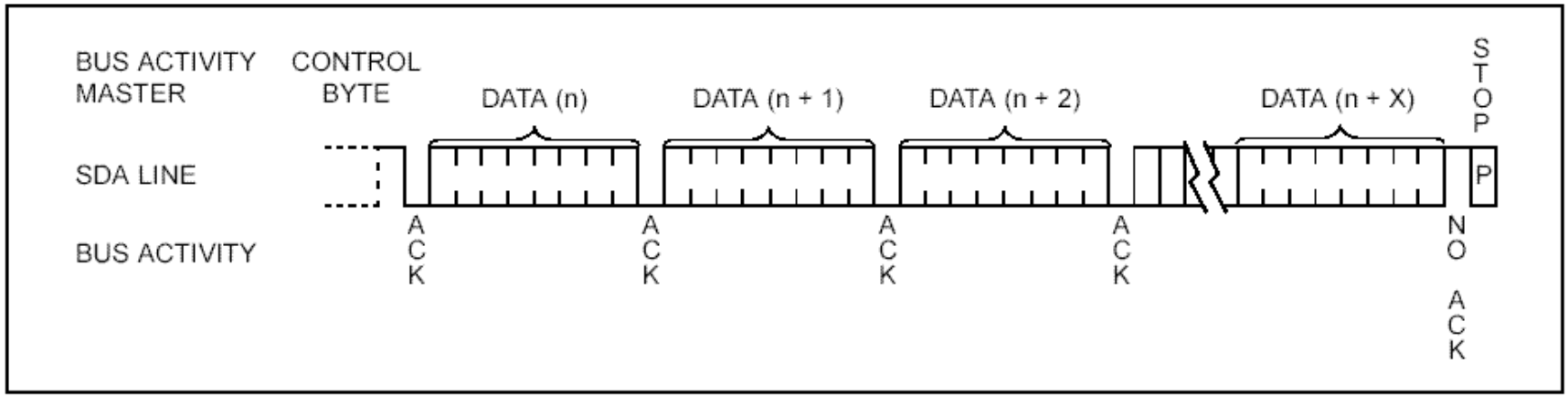


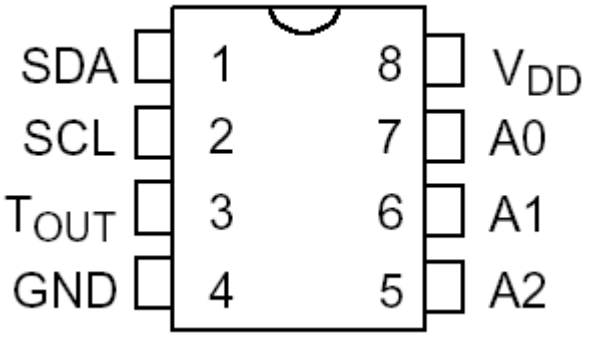
FIGURE 7-3: SEQUENTIAL READ



Ejemplos de dispositivos que utilizan I^2C

Termómetro / Termostato
DS1621

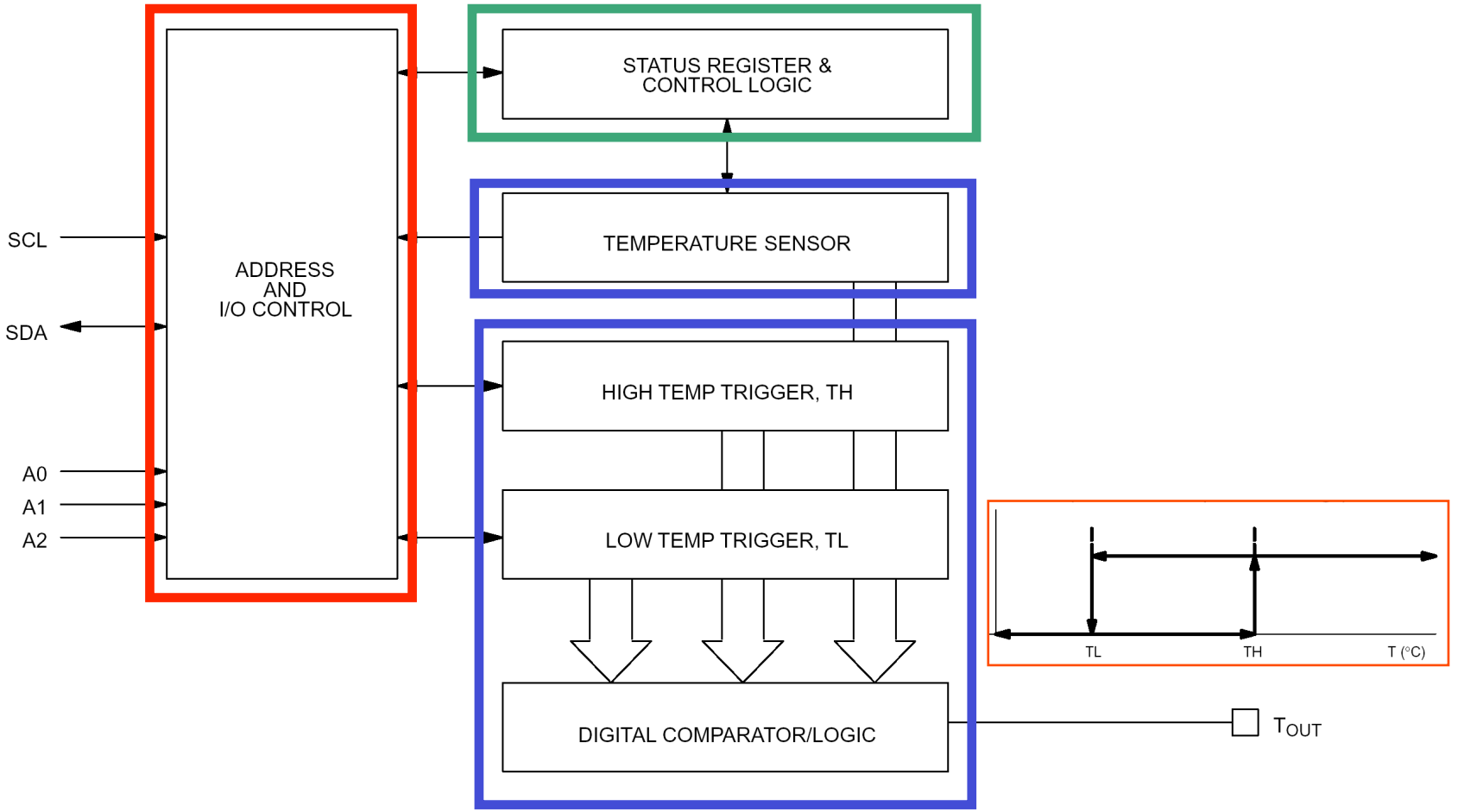
- **Margen de temperatura: -55 a +125ºC**
- **Resolución: 0.5ºC**
- **Tª almacenada en 9 bits (2 bytes)**
- **Rango de alimentación: 2.7-5.5V**
- **Tiempo de conversión: 1 seg.**
- **Parámetros del termostato (TH y TL) almacenados en memoria no volátil (eeprom)**
- **No requiere circuitería externa.**



Ejemplos de dispositivos que utilizan I²C

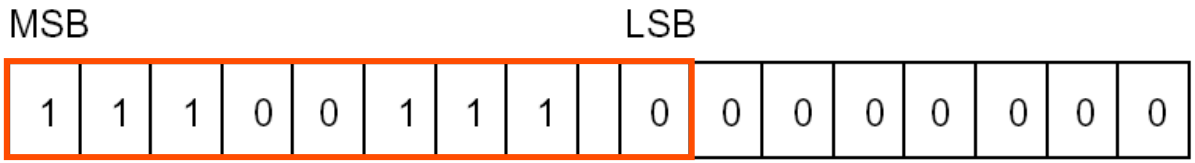
BUS I2C

Termómetro / Termostato



Ejemplos de dispositivos que utilizan I²C

Termómetro / Termostato



T = –25°C

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	01111101 00000000	7B00h
+25°C	00011001 00000000	1900h
+1/2°C	00000000 10000000	0080h
+0°C	00000000 00000000	0000h
–1/2°C	11111111 10000000	FF80h
–25°C	11100111 00000000	E700h
–55°C	11001001 00000000	C900h

Termómetro / Termostato

Registro de Configuración

DONE	THF	TLF	NVB	1	0	POL	1SHOT

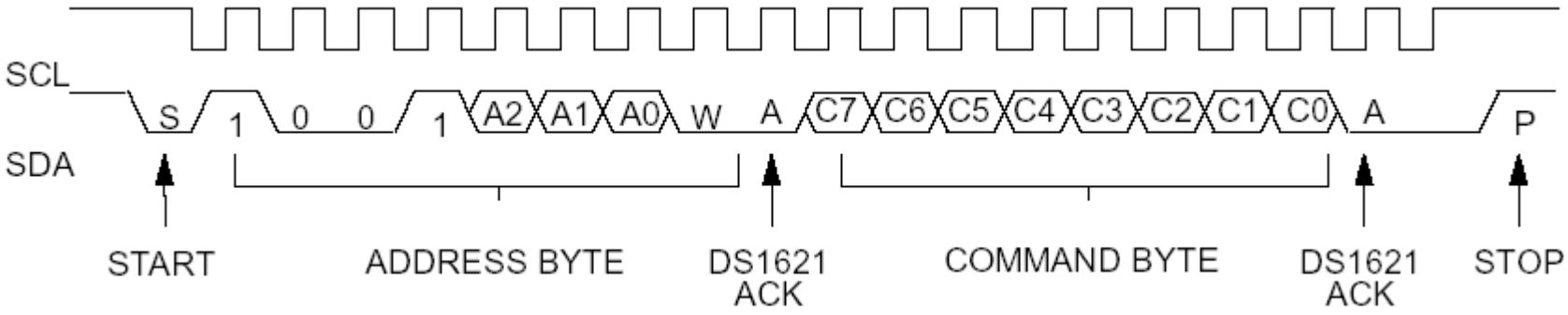
- **DONE. Flag de fin de conversión.**
- **THF. Flag (T^a > TH).**
- **TLF. Flag (T^a < TL).**
- **NVB. Flag indicador de que la grabación en eeprom está en proceso.**
- **POL. Polaridad de la salida Tout.**
- **1SHOT. Selección del modo de conversión única o continua. En el modo de conversión continua, el termostato puede funcionar de forma autónoma, sin necesidad de uC, una vez configurados TH y TL.**

Comandos (1 byte):

- Lectura de la temperatura [AAh]
- Acceso a TH [A1h]
- Acceso a TL [A2h]
- Acceso al Reg. de Config. [Ach]
- Inicio de conversión [EEh]
- Parar la conversión [22h]

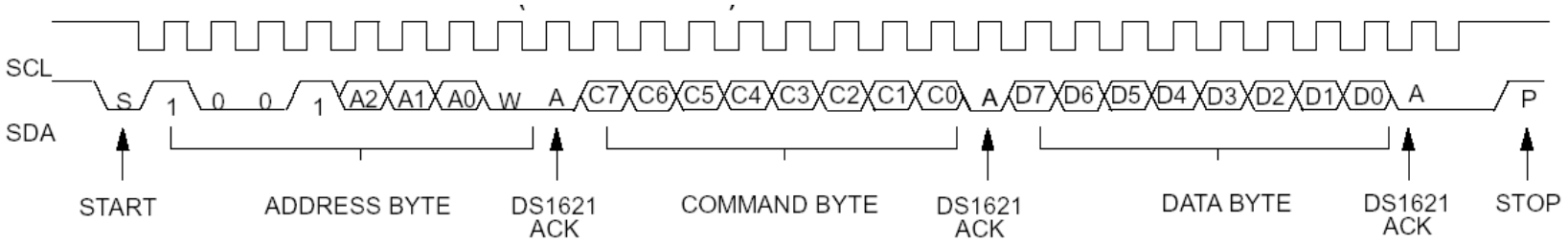
DS1621. Protocolo de acceso

Comando Inicio/Stop de conversión



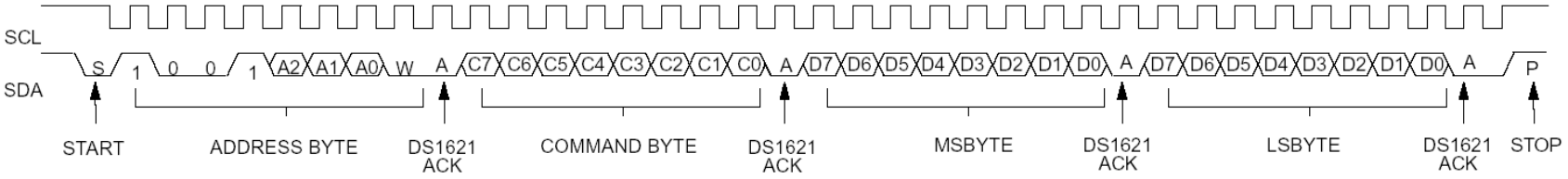
DS1621. Protocolo de acceso

Escritura de un dato en un registro (1byte) (Reg. De Config.)



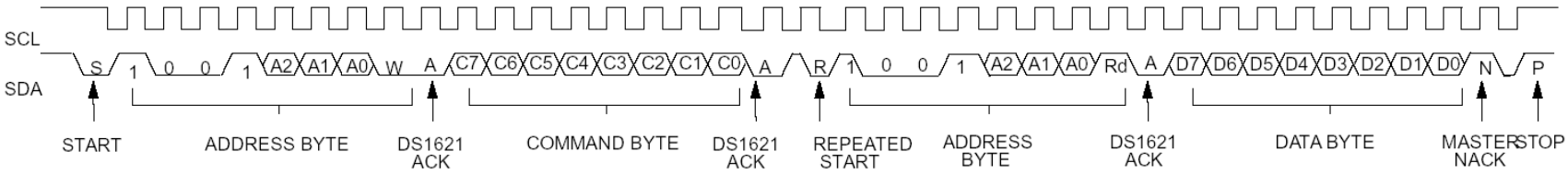
DS1621. Protocolo de acceso

Escritura de un dato en un registro (2bytes) (TH, TL)



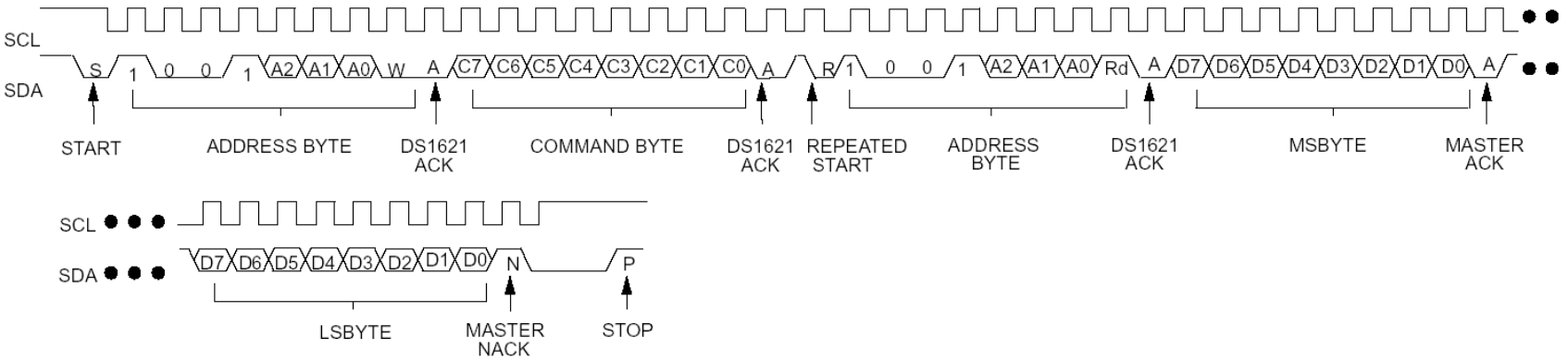
DS1621. Protocolo de acceso

Lectura de un dato (1byte) de un registro (Reg. De Config.)



DS1621. Protocolo de acceso

Lectura de un dato (2 bytes) de un registro (Tª, TH, TL)



DS1621. Protocolo de acceso

Configuración del dispositivo (modo de conversión continua).

Configurar valores de TH y TL (si opción termostato).

Comando de Inicio de conversión (1ª vez)

Llamada a rutina de lectura de la Tª:

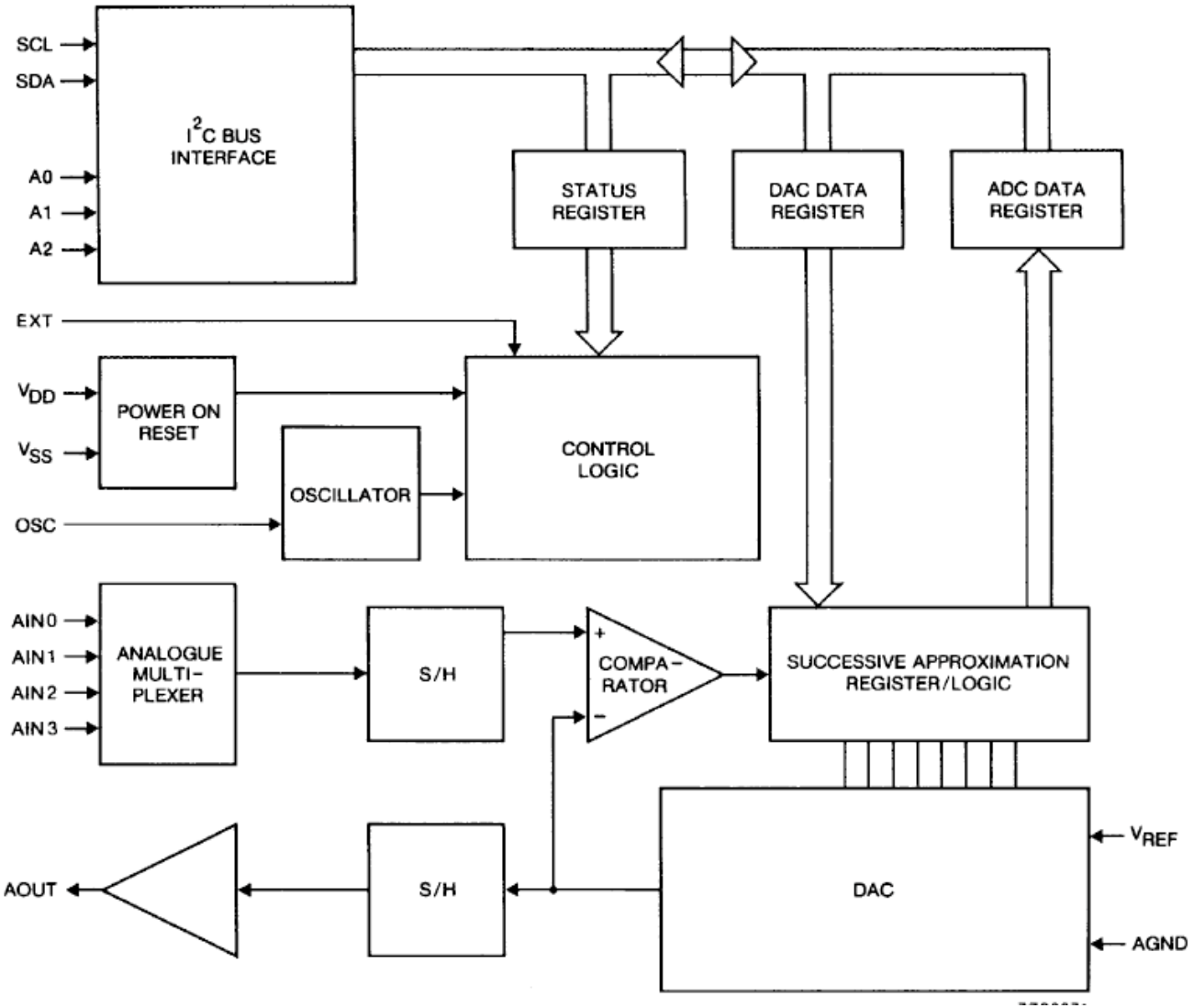
PCF8591. Conversor A/D y D/A

1 FEATURES

- Single power supply
- Operating supply voltage 2.5 V to 6 V
- Low standby current
- Serial input/output via I²C-bus
- Address by 3 hardware address pins
- Sampling rate given by I²C-bus speed
- 4 analog inputs programmable as single-ended or differential inputs
- Auto-incremented channel selection
- Analog voltage range from V_{SS} to V_{DD}
- On-chip track and hold circuit
- 8-bit successive approximation A/D conversion
- Multiplying DAC with one analog output.

Ejemplos de dispositivos que utilizan I²C

BUS I2C



Ejemplos de dispositivos que utilizan I²C

PCF8591. Conversor A/D y D/A

SYMBOL	PIN	DESCRIPTION
AINO	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I²C-bus data input/output
SCL	10	I²C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

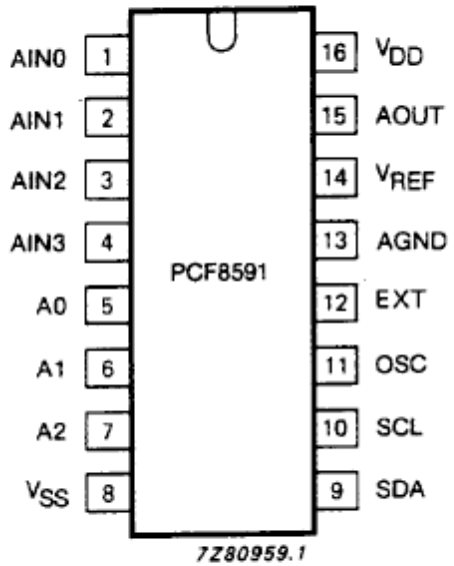


Fig.2 Pinning diagram.

PCF8591. Conversor A/D y D/A

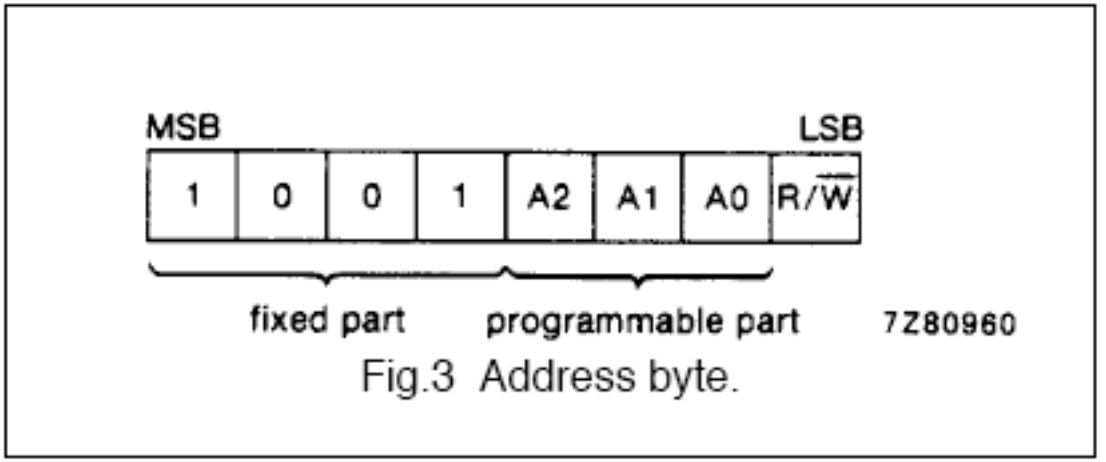
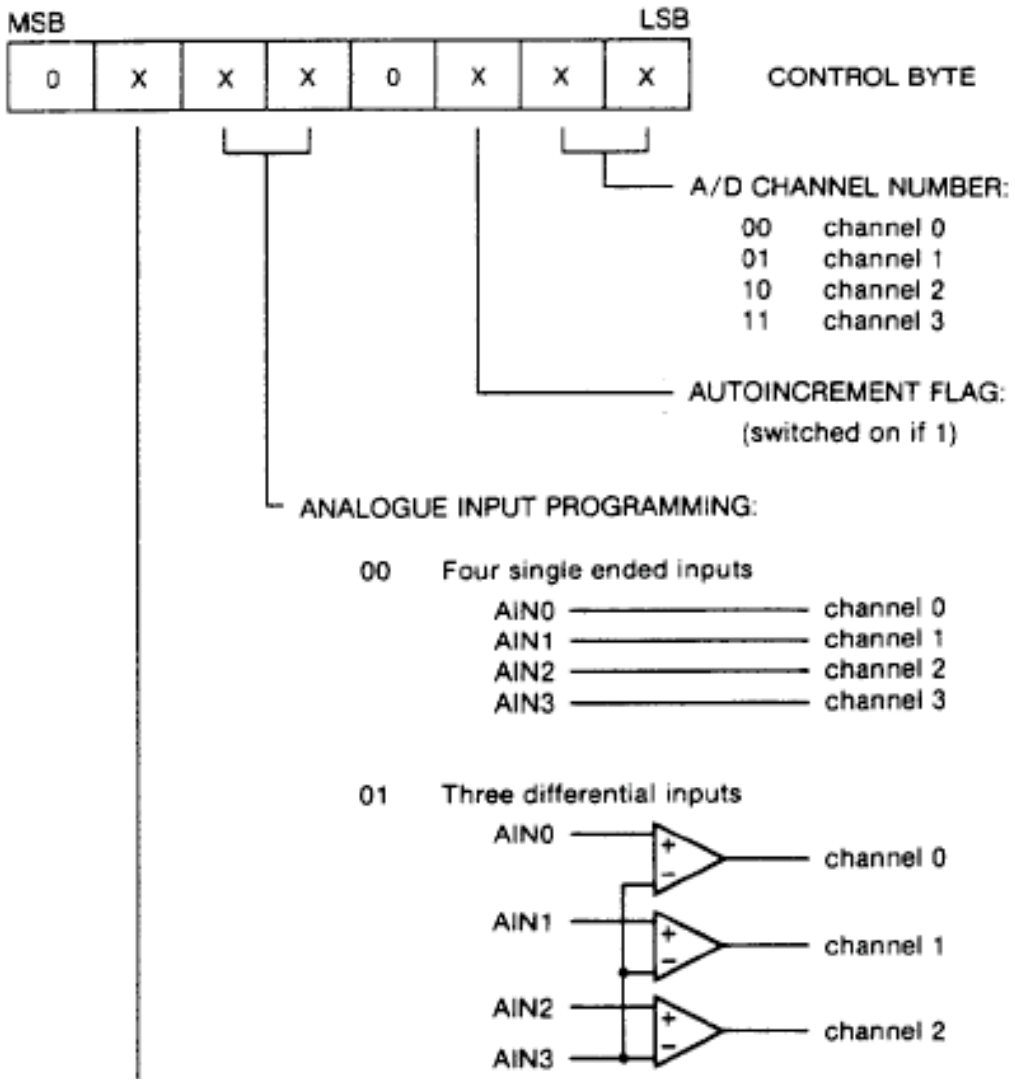


Fig.3 Address byte.

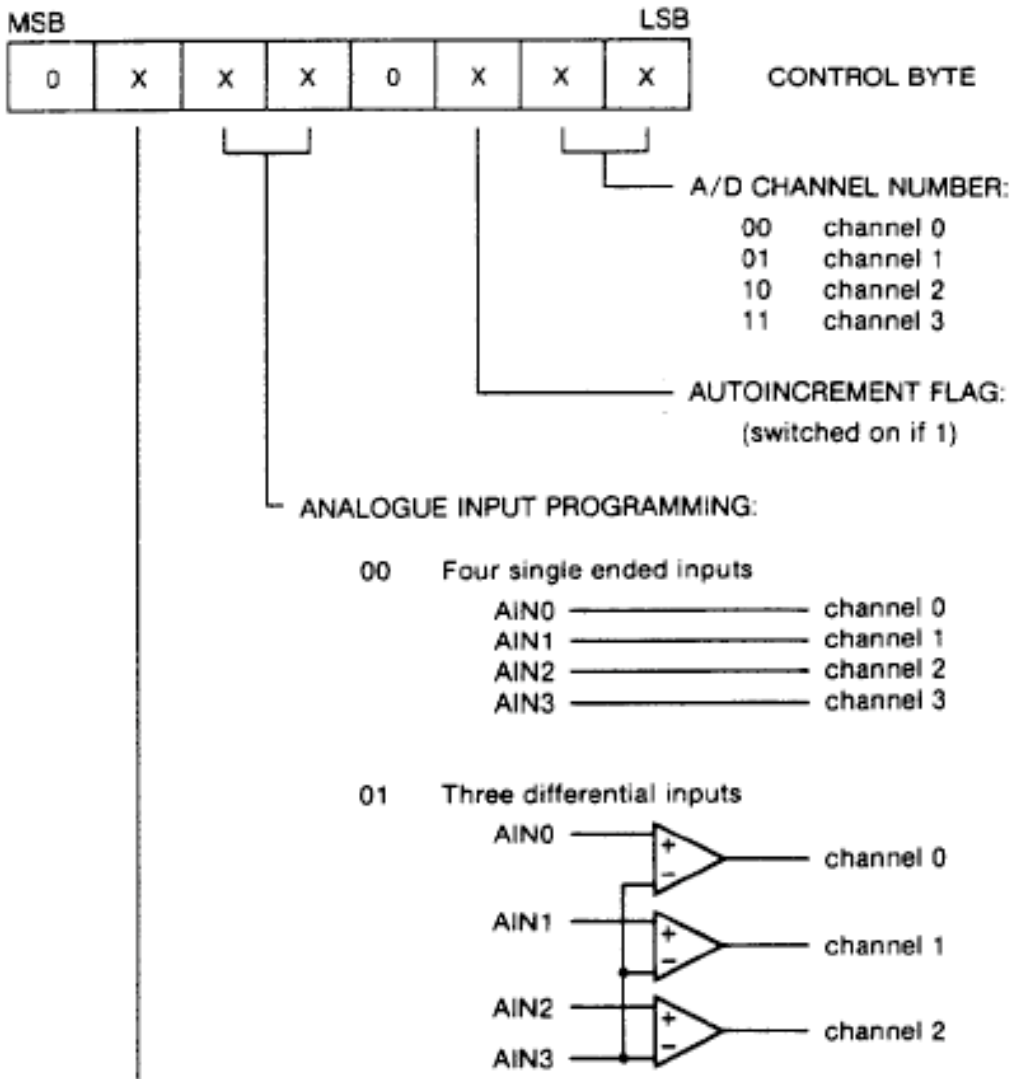
Ejemplos de dispositivos que utilizan I²C

BUS I2C



Ejemplos de dispositivos que utilizan I²C

BUS I2C



Ejemplos de dispositivos que utilizan I²C

PCF8591. Conversor A/D y D/A

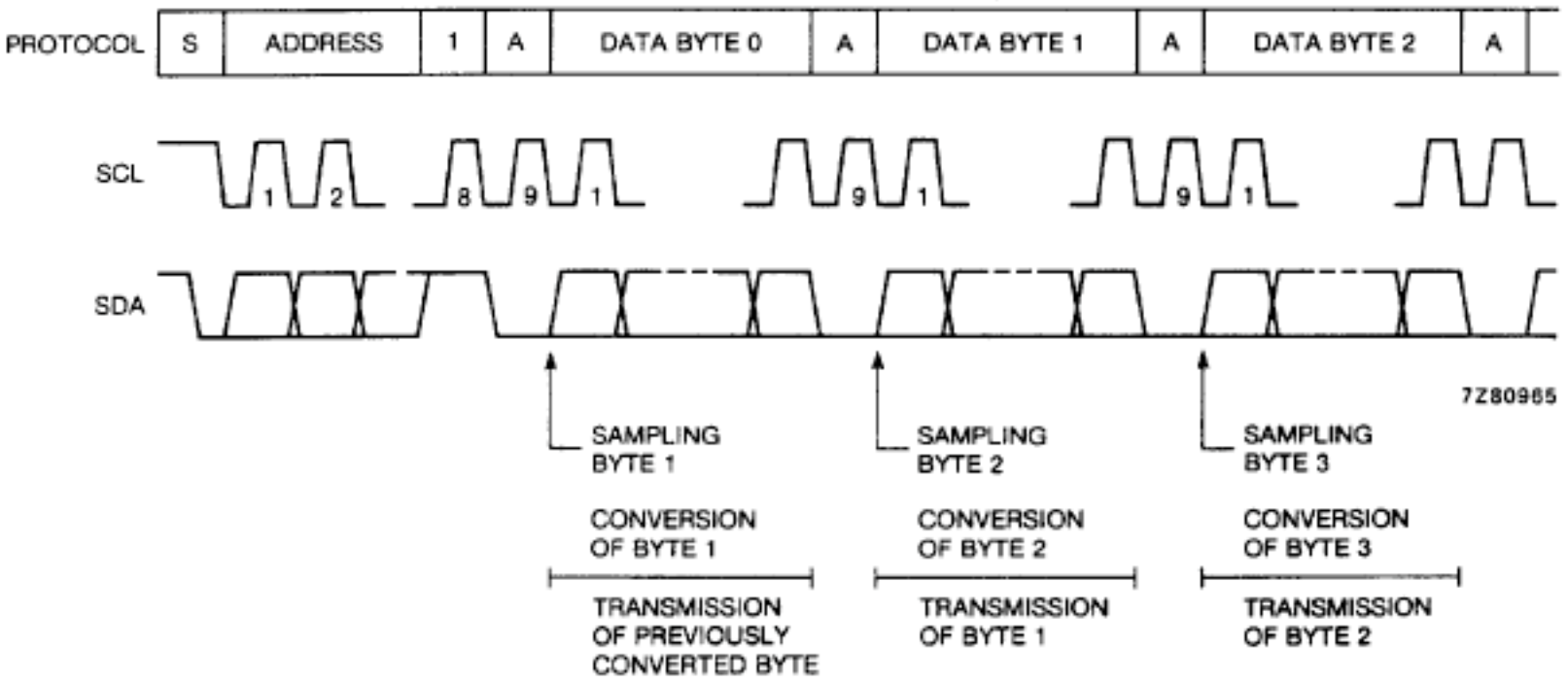


Fig.8 A/D conversion sequence.

Índice de la Lección

- Introducción a la comunicación entre dispositivos mediante un bus serie
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

Conclusiones

- Un bus serie permite reducir el tamaño general del circuito
- El protocolo del bus I^2C contempla
  - Conexión de nivel físico
  - Intercambio de datos
- El protocolo se puede implementar
  - Utilizando puertos de entrada/salida → maestro
  - Unidad específica
- Gran cantidad de dispositivos compatibles con I^2C

Índice de la Lección

- Introducción a la comunicación entre dispositivos mediante un bus serie
- Características del bus I²C
- Conexión de dispositivos al bus: nivel físico
- Intercambio de información: nivel de enlace
- Generación del protocolo desde un microcontrolador
- Ejemplos de dispositivos que utiliza el bus I²C
- Conclusiones
- Bibliografía

Referencias

Bibliografía

"El bus I2C. De la teoría a la práctica"

Dominique Paret Ed. Paraninfo. 1995. (ISBN: 84-283-2167-1)

"The I2C-BUS Specification" Versión 2.1 Enero 2000

Hojas de características de los componentes explicados

Referencias

Enlaces de Internet

The I2C FAQ

http://www.ping.be/~ping0751/i2cfaq/i2cindex.htm

Single Master I2C driver routines

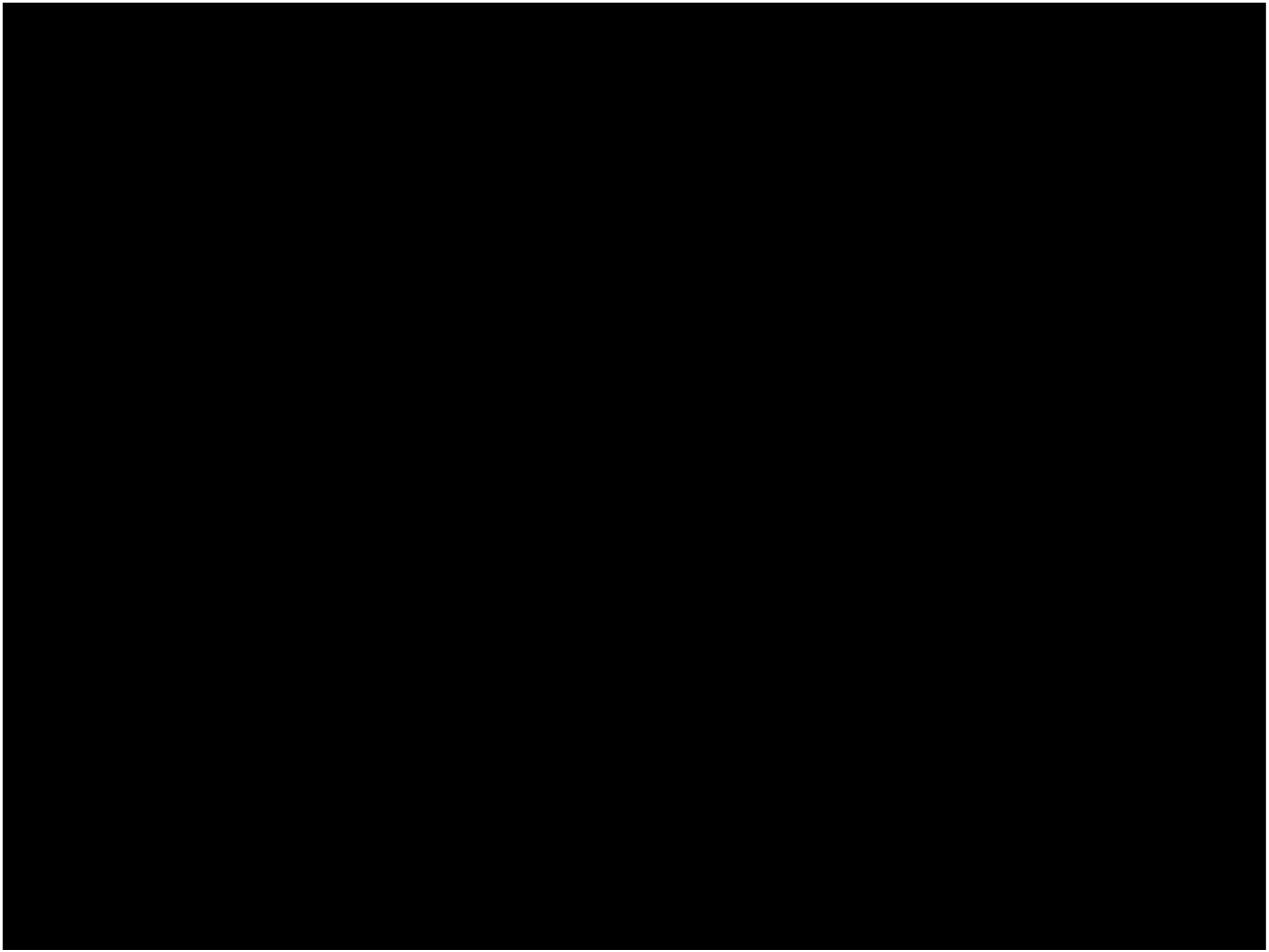
http://www.specs.de/users/danni/appl/hard/i2c/

I2C Bus Technical Overview and Frequently Asked Questions (FAQ)

by Axel Wolf, ESAcademy based on the I2C FAQ by Vince Himpe

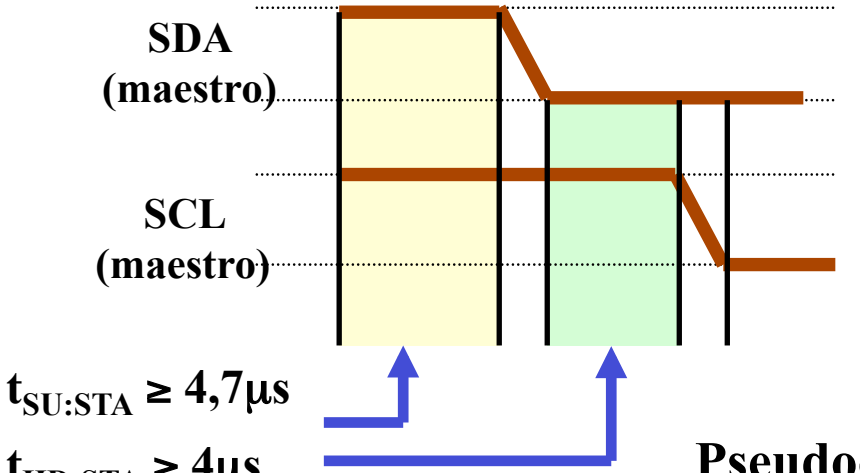
Embedded Systems Academy:

http://www.esacademy.com/faq/i2c/index.htm



Control del bus I2C utilizando puertos de E/S

I2C_START



Pseudocódigo

SDA = 1
SCL = 1
delay(4,7µs)
SDA = 0
delay(4µs)
SCL = 0

Ensamblador

```
I2C     EQU PORTS
DI2C    EQU DDRS
SDA     EQU %00000100
SCL     EQU %00001000
```

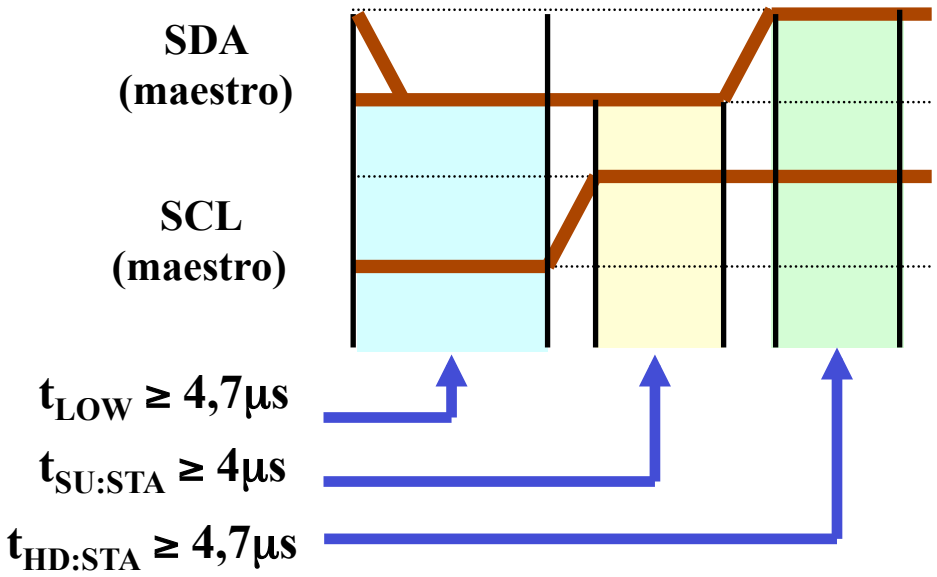
Ensamblador

```
I2C_START   BSET   DI2C,SDA
            BSET I2C,SDA
            BSET I2C,SCL
            BSR delay47
            BCLR I2C,SDA
            BSR delay4
            BCLR I2C,SCL
            RTS
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_STOP



Pseudocódigo

```
SDA = 0
SCL = 0
delay(4,7µs )
SCL = 1
delay(4µs )
SDA = 1
delay(4,7µs )
```

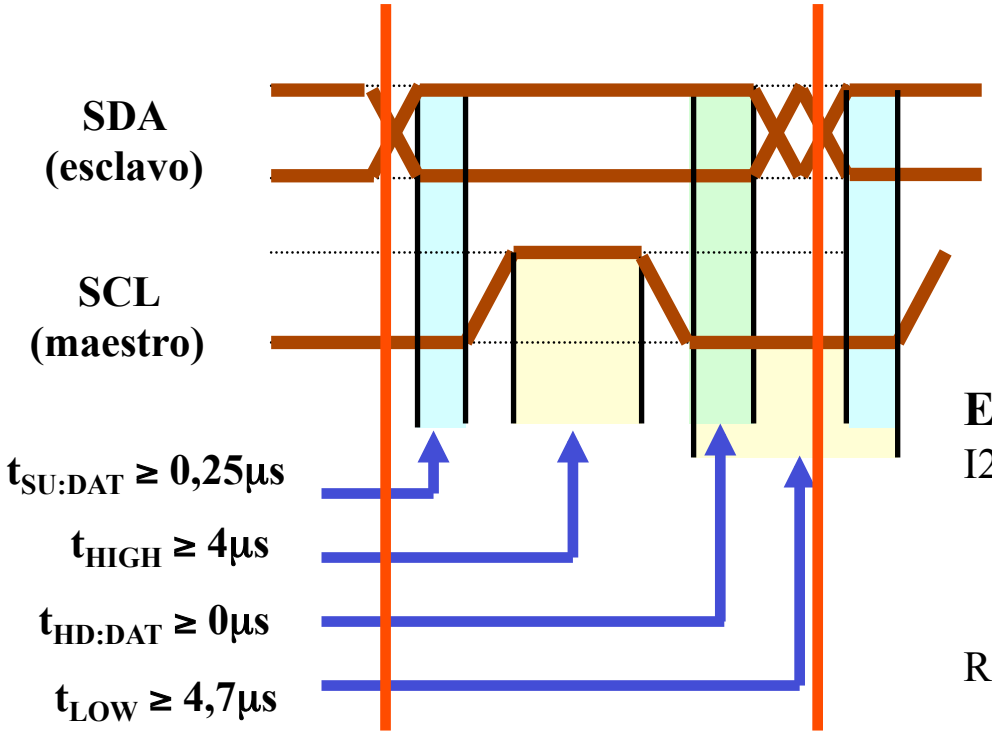
Ensamblador

```
I2C_STOP    BSET   DI2C,SDA
            BCLR   I2C,SDA
            BCLR   I2C,SCL
            BSR    delay47
            BSET   I2C,SCL
            BSR    delay4
            BSET   I2C,SDA
            BSR    delay47
            RTS
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_RXBIT



Pseudocódigo

```
SCL = 1
delay(4µs )
SDA = BIT
SCL = 0
delay(4,7µs )
```

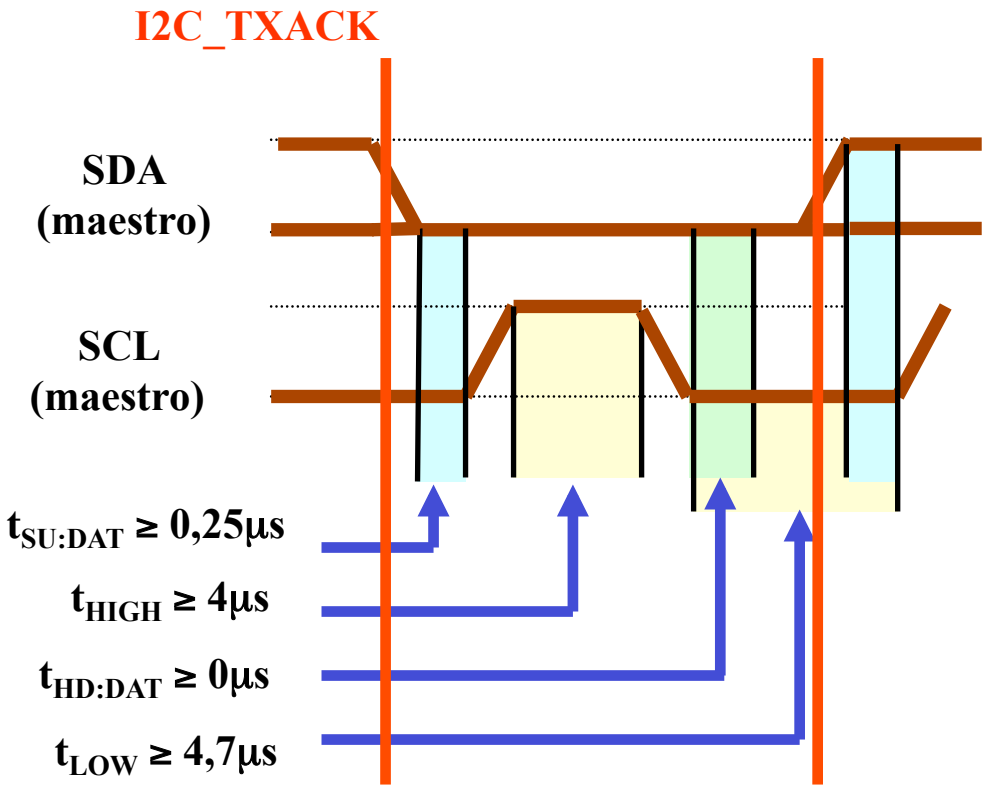
Ensamblador (BIT = carry)

```
I2C_RXBIT   BCLR   D I2C,SDA
            BSET   I2C,SCL
            BSR    delay4
            BRSET I2C,SDA,RX_UNO
RX_CERO     CLC
            BRA    RX_SUGUE
RX_UNO      SEC
TX_SUGUE    BCLR   I2C,SCL
            BSR    delay4,7
            RTS
```

Generación del protocolo desde un µC

BUS I2C

Control del bus I2C utilizando puertos de E/S



Pseudocódigo

```
SDA = 0
SCL = 1
delay(4µs )
SCL = 0
delay(4,7µs )
```

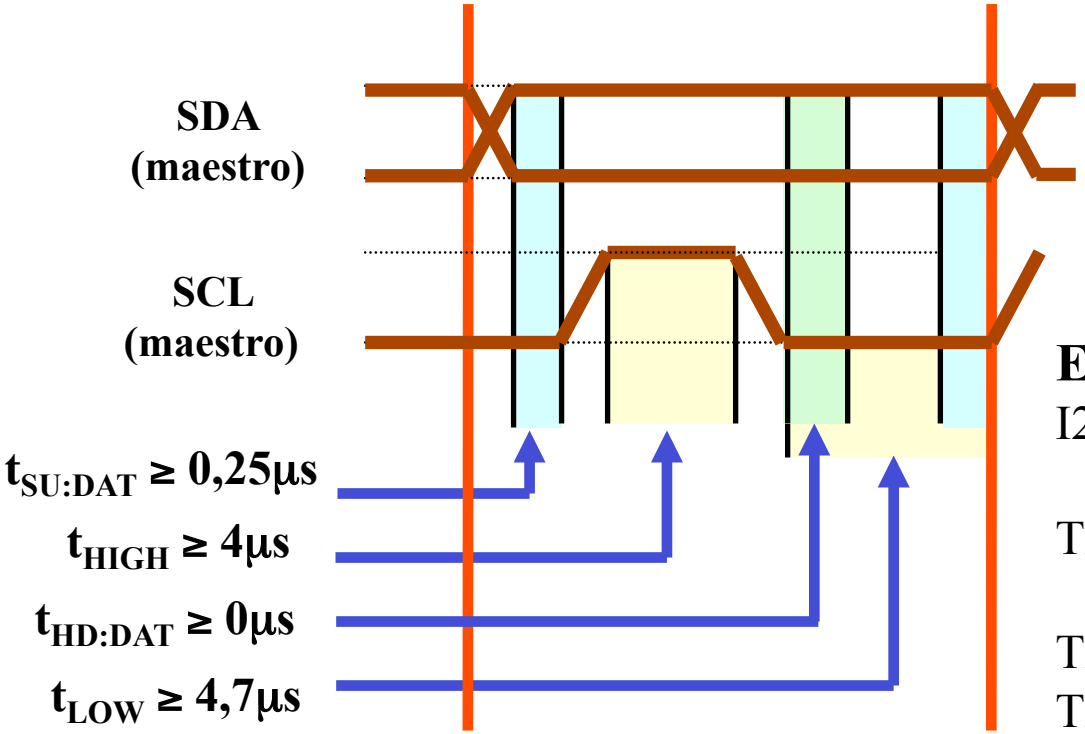
Ensamblador (ACK = carry)

```
I2C_TXACK   CLC
            BSR I2C_TXBIT
            RTS
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_TXBIT



Ensamblador

```
I2C     EQU PORTS
DI2C    EQU DDRS
SDA     EQU %00000100
SCL     EQU %00001000
```

Pseudocódigo

```
SDA = BIT
SCL = 1
delay(4µs )
SCL = 0
delay(4,7µs )
```

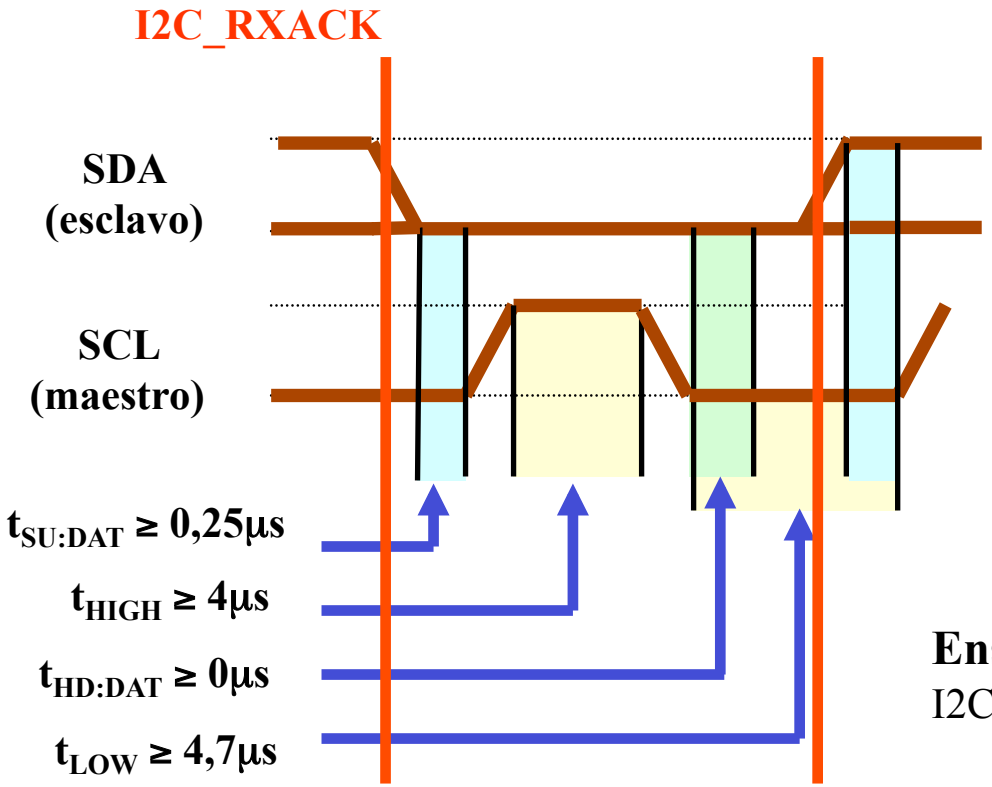
Ensamblador (carry=BIT)

```
I2C_TXBIT   BSET   D I2C,SDA
            BCS    TX_UNO
TX_CERO     BCLR   I2C,SDA
            BRA    TX_SIGUE
TX_UNO      BSET   I2C,SDA
TX_SIGUE    BSR    delay4
            BCLR   I2C,SCL
            BSR    delay4,7
            RTS
```

Generación del protocolo desde un µC

BUS I2C

Control del bus I2C utilizando puertos de E/S



Pseudocódigo

SCL = 1
delay(4µs)
BIT = SDA
SCL = 0
delay(4,7µs)

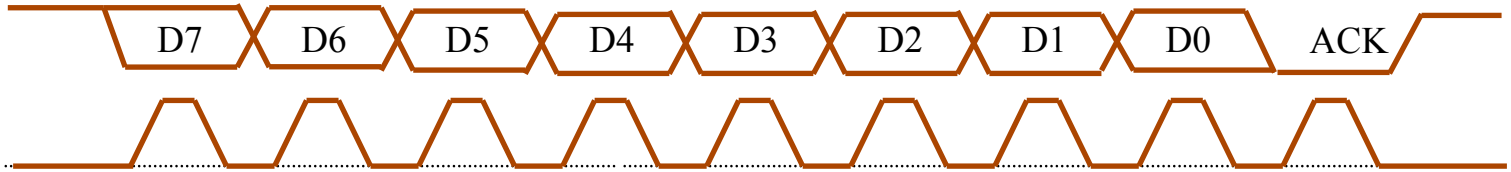
Ensamblador (ACK = carry)

I2C_RXACK BSR I2C_RXBIT
RTS

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_TXBYTE



Pseudocódigo

Repite 8
    DESPLAZA DATO ←
    I2C_TXBIT
I2C_RXACK
Devuelve ACK

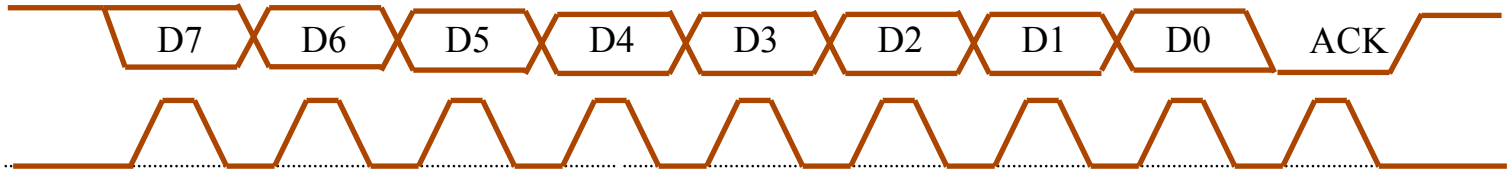
Ensamblador (ACK = carry)

```
I2C_TXBYTE   PSHB
             LDAB #8
OTRO_TX      LSLA
             BSR I2C_TXBIT
             DBNE  B,OTRO_TX
             POPB
             BSR I2C_RXACK
             RTS
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_RXBYTE



Pseudocódigo

```
Repite 8
        I2C_RXBIT
        DESPLAZA DATO ←
I2C_TXACK
Devuelve DATO
```

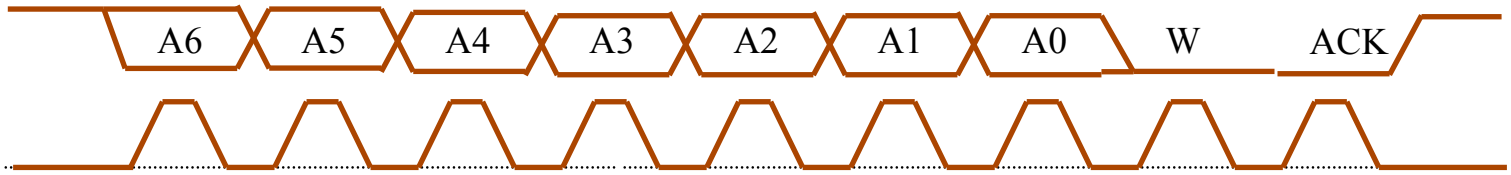
Ensamblador (A = DATO)

```
I2C_RXBYTE   PSHB
             LDAB #8
OTRO_RX      BSR I2C_RXBIT
             ROLA
             DBNE  B,OTRO_RX
             POPB
             BSR I2C_TXACK
             RTS
```

Generación del protocolo desde un µC

BUS I2C

I2C_DIR_W



Pseudocódigo

A=DIR <<1 + W(0)
I2C_START
I2C_TXBYTE
Devuelve ACK

Ensamblador (ACK = carry)

```
I2C_DIRW    CLC
            ROLA
            BSR  I2C_START
            BSR  I2C_TXBYTE
            RTS
```

I2C_DIR_R

Pseudocódigo

A=DIR<<1 + R(1)
I2C_START
I2C_TXBYTE
Devuelve ACK

Ensamblador (ACK = carry)

```
I2C_DIRR    SEC
            ROLA
            BSR  I2C_START
            BSR  I2C_TXBYTE
            RTS
```

Control del bus I2C utilizando puertos de E/S

LLAMADAS DESDE LENGUJE C

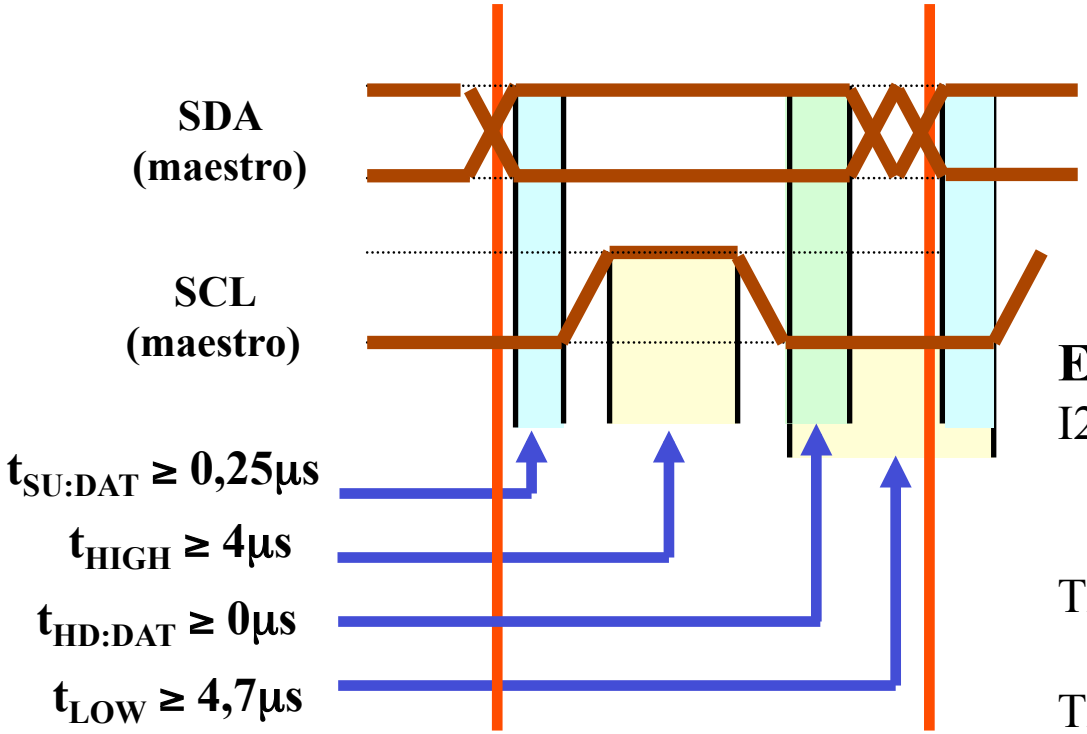
Leer y escribir datos del puerto

```
unsigned  char in_port (unsigned char dir){

}

unsigned  char out_port (unsigned char dir; unsigned char dato){

}
```

Generación del protocolo desde un µC

Control del bus I2C utilizando puertos de E/S

I2C_TXBIT



Pseudocódigo

```
SDA = BIT
SCL = 1
delay(4µs )
SCL = 0
delay(4,7µs )
```

Ensamblador (carry=BIT)

```
I2C_TXBIT   BSET   D I2C,SDA
            BCS    TX_UNO
            BCLR   I2C,SDA
TX_CERO     BCLR   I2C,SDA
            BRA    TX_SUGUE
TX_UNO      BSET    I2C,SDA
TX_SUGUE    BSR     delay4
            BCLR    I2C,SCL
            BSR     delay4,7
            RTS
```

Protocolo de Comunicación I²C

Introducción

En las lecciones anteriores se ha visto:

- Interacción con el entorno
  - Puertos de entrada/salida
  - Expansión de memoria externa
- Comunicación serie
  - Asíncrona
  - Sencillo interfaz síncrono

En esta lección vamos a ver:

- Expansión de recursos utilizando un bus serie
- Protocolo de comunicación
- Generación del protocolo
- Ejemplos de dispositivos

Características del bus I²C

Origen del bus I²C (Inter Integrated Circuits Bus)

- Desarrollado por Philips a principios de los 80 como medio de interconexión entre una CPU y dispositivos periféricos dentro de la electrónica de consumo.
  - Simplificar las conexiones entre los periféricos (pistas, decodificadores, ..)
  - Aumento de la inmunidad al ruido
  - Control de sistemas de audio y vídeo (baja velocidad)
- Actualmente diseñan dispositivos basados en I²C muchos fabricantes
  - Xicor, SGS-Thomson, Siemens, Intel, TI, Maxim, Atmel, Analog Devices
- **Aplicaciones**
  - Bus de interconexión entre dispositivos en una tarjeta o equipo
  - Sistema de configuración y supervisión en ordenadores servidores
  - Sistemas de gestión de alimentación
  - Conexión en serie de dispositivos externos a un ordenador
  - Tarjetas chip

Puerto de 8 bits de entrada/salida – PCF8574

- Puerto de entrada y salida
  - Para configurarlo como entrada se debe escribir un nivel alto en el registro
    - Al leer se consulta el nivel del pin del puerto
  - Si en un puerto se escribe un cero, la salida pasa a nivel bajo
  - Se comporta como una salida en colector abierto con pull – up interno
  - Después de la inicialización
    - Configurado como entrada
  - Si se modifica una entrada se activa una salida de interrupción que se desactiva cuando se lee el dispositivo (a través del I2C)