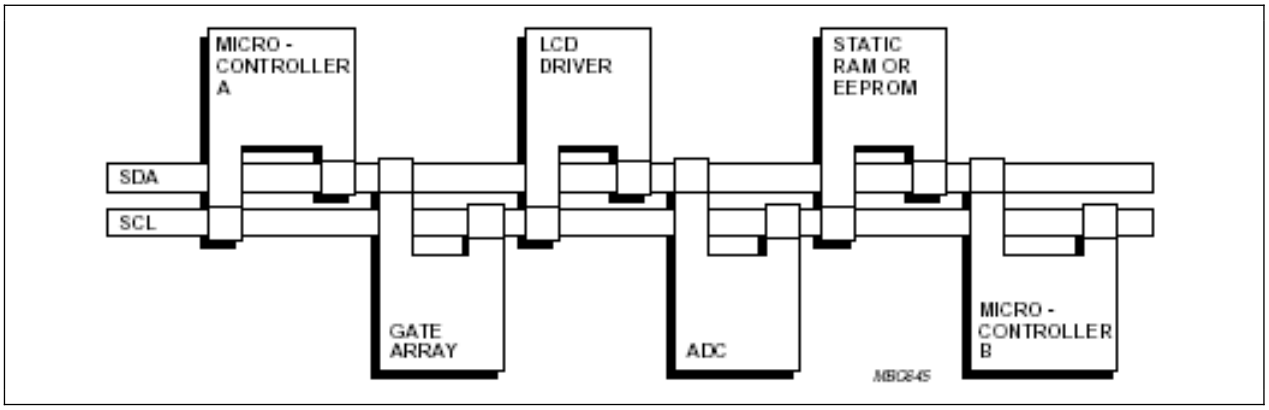
Bus I2C™ (Inter-Integrated Circuit) *(section 19)*

Es un bus bidireccional síncrono, de la empresa Philips, muy utilizado industrialmente para comunicar periféricos con microcontroladores.

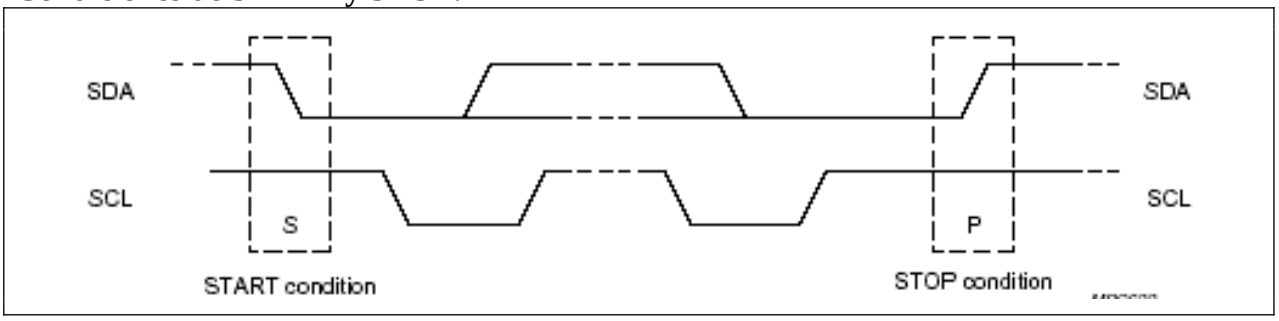
La figura a continuación ilustra la comunicación entre dos microcontroladores y varios periféricos mediante un bus I2C. Tal y como se puede apreciar para realizar la comunicación no hacen falta más que dos señales o líneas: la señal de reloj (SCL) ya que se trata de un bus síncrono, y la señal de datos (SDA), mediante la que se trasmiten tanto datos como direcciones de los componentes que se comunican.



Terminología realacionada con el bus I2C:

Emisor:	componente que envía datos al bus.
Receptor:	componente que recibe los datos presentes en el bus.
Maestro:	componente que controla la transferencia: la inicia, genera la señal de reloj y termina la transferencia. Puede ser emisor o receptor.
Esclavo:	componente al que se dirige un maestro. Un esclavo puede ser emisor o receptor.
Multimaestro:	puede ocurrir que más de un maestro intente controlar el bus al mismo tiempo; gracias al arbitraje no hay problema.
Arbitraje:	el hardware asegura que si más de un maestro intenta acceder simultáneamente al bus, sólo uno de ellos será autorizado a hacerlo.
Sincronización:	procedimiento que sincroniza las señales de reloj suministradas por dos o más componentes.
SDA:	señal de datos.
SCL:	señal de reloj.
F (*FREE*) LIBRE:	el bus está libre; la línea de datos SDA y la de reloj SCL se encuentran ambas en nivel alto de tensión (H).

Condiciones de START y STOP:

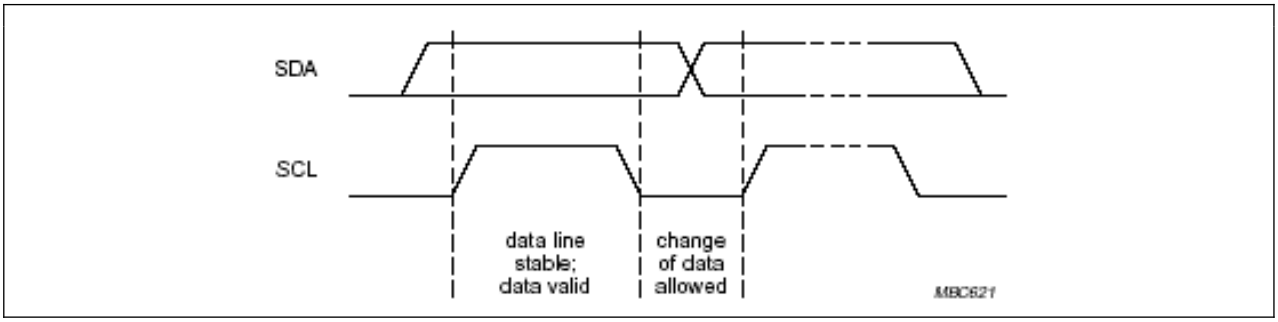


S (START) INICIO: La condición de Start da comienzo a la transferencia de datos y se identifica porque el nivel de la línea de datos SDA cambia de estado alto a bajo mientras la línea de reloj SCL permanece en estado alto. A partir de ese momento el bus estará ocupado hasta que finalice la transferencia.

P (STOP) FIN: la transferencia de datos se termina con una condición de Stop. Esta condición se produce cuando el nivel en la línea de datos SDA pasa del estado bajo al estado alto, mientras la línea de reloj SCL permanece en estado alto. A partir de ese momento el bus estará libre hasta que otro maestro comience una transferencia.

Nota: No se debe confundir la condición de Start o Stop del bus I2C con los bits de Start y Stop que hemos visto en el protocolo de la UART. Los bits de Start y Stop tienen una duración acorde a la velocidad de transmisión (a 9600 baudios, alrededor de 105 microsegundos).

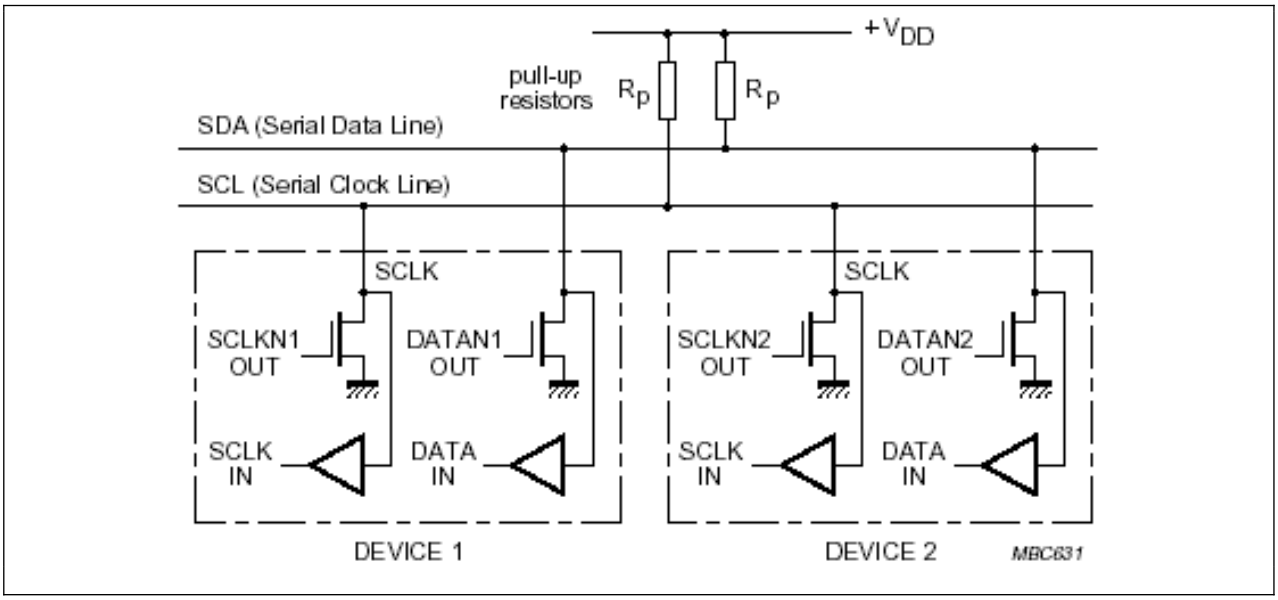
Condiciones de CAMBIO y DATO



C (CHANGE) CAMBIO: mientras la línea de reloj SCL se halla en bajo, puede cambiar el nivel de la línea de datos de SDA.

D (DATA) DATO: si la línea de datos SDA permanece constante (en alto o en bajo) mientras la señal de reloj SCL permanece en alto, se considera válido el bit trasmitido; de lo contrario, si el nivel de SDA cambia en ese intervalo, se considerará como condición de Start o Stop, dependiendo del cambio.

Esquema de dos dispositivos conectados al I2C



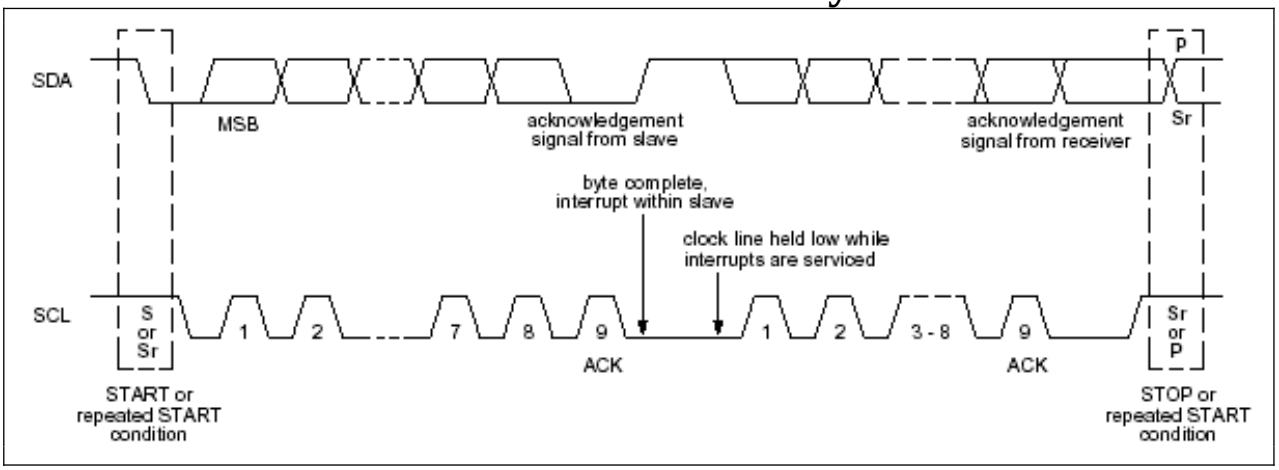
Las dos líneas del bus, SDA y SCL, son bidireccionales y están conectadas a la línea positiva de alimentación a través de dos resistencias de polarización (*pull-up*) cuya misión es asegurar que en las dos líneas la tensión eléctrica es adecuada. Se dice que es una conexión en colector o drenador abierto (*open-collector*, *open-drain*).

En reposo, cuando los transistores no conducen, se dice que el bus está libre y que las líneas están descansando en el nivel alto gracias a las resistencias de polarización.

La velocidad de transmisión de los datos en el bus puede llegar a 100 kbits/s en el modo estándar, a 400 kbits/s en el modo rápido (*Fast*) y a 3,4 Mbit/s en el modo de alta velocidad (*High-speed*).

El número de dispositivos conectados al bus depende solo de la capacitancia del bus, que está limitada a 400pF.

Transmisión de la dirección del esclavo y de un dato

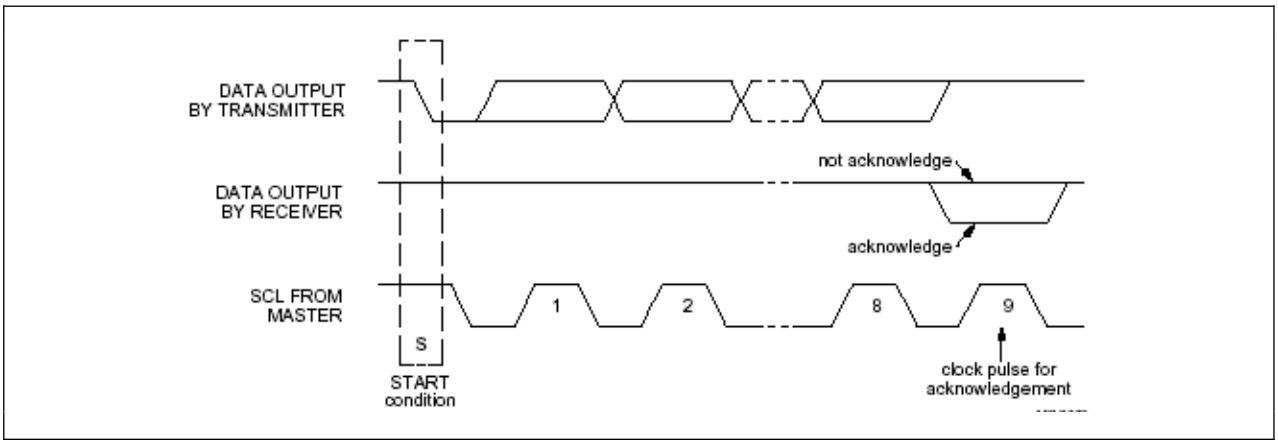


La figura muestra en detalle la secuencia de una transmisión de un dato de 1 byte:

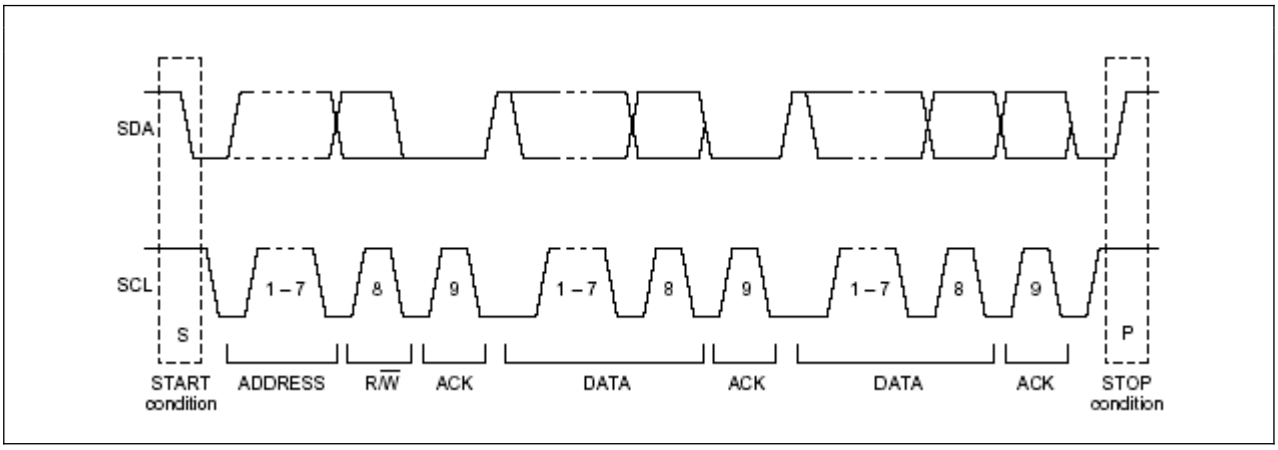
1. El dispositivo que quiere realizar la transmisión (el maestro) transmite la condición de *Start*.
2. El maestro trasnmite la dirección del esclavo así como el sentido de la operación: lectura o escritura .
3. Si el esclavo activa el bit de reconocimiento (ACK), el maestro continuará con la transmisión. De lo contrario, si el esclavo no contesta (por ejemplo porque está ocupado en otra tarea), el maestro puede tomar diferentes decisiones: intentar otra transferencia, volver a intentar la misma, abandonar la transferencia...
4. Si el esclavo ha accedido a la transferencia el dato es transmitido por el bus. Hay que tener en cuenta que el sentido puede ser cualquiera: del maestro (emisor) al esclavo (receptor) o del esclavo (emisor) al maestro (receptor). Al finalizar el dato el receptor debe enviar el ACK para indicar que ha recibido el dato. Como excepción, si el receptor es el maestro y no quiere recibir más datos del esclavo, no envía el ACK.
5. Finalmente, el maestro genera la condición de *Stop* o se repite la condición de *Start* si quiere realizar una nueva transferencia.

Un dispositivo receptor puede reducir la velocidad de transmisión del bus alargando los periodos bajos del reloj. De esta manera, la velocidad de cualquier maestro puede adaptarse a la del esclavo.

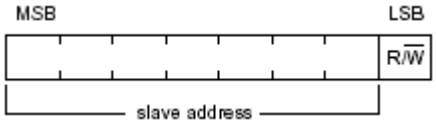
Detalle del acuse de recepción (*Acknowledge*):



Transferencia completa:



Bits del primer byte:



Los primeros 7 bits del primer byte indican la dirección del esclavo. El octavo bit (el de menor peso = LSB, *least significant bit*) determina la dirección del mensaje: un '0' en este bit indica que el maestro va a escribir información en el esclavo seleccionado; un '1' indica que va a leer información del esclavo.

Cuando un maestro envía una dirección por el bus, cada dispositivo compara los 7 bits siguientes al *Start* con su dirección. Si coinciden, el esclavo se considera direccionado por el maestro como esclavo receptor o como esclavo transmisor, dependiendo del octavo bit (R/W).

Durante todo el proceso, la dirección de transferencia no cambia, y el número de datos es ilimitado. Según se ha comentado ya, las direcciones de los dispositivos son de 7 bits, pero hay algunas direcciones reservadas que no se pueden utilizar:

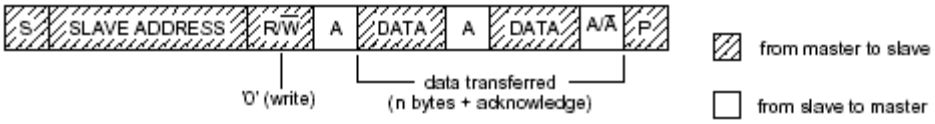
Dir-Esclavo	**R/W**	**DESCRIPCION**
0000 000	0	Dirección de llamada general
0000 000	1	START byte
0000 001	x	Dirección CBUS
0000 010	x	Reservado para formatos de bus diferentes
0000 011	x	Reservado para propósitos futuros
0000 1xx	x	Modo *High-speed*
1111 1xx	x	Reservado para propósitos futuros
1111 0xx	x	Dirección de esclavo de10-bits

Detalle de los bits de dirección reservados.

Algunos formatos de transferencia posibles

Maestro emisor y esclavo receptor

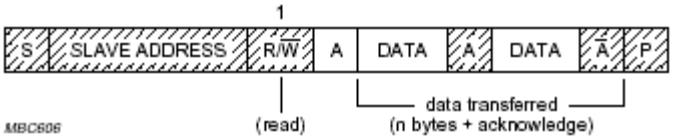
Al inicio, el maestro transmite un primer byte con los 7 bits de la dirección asociada al esclavo y un bit indicativo de la operación, escritura, y a continuación envía los datos.



A = Ack (SDA LOW) */A = no se ha recibido Ack (SDA HIGH)*
S = condición de START *P = condición de STOP*
R/W = 0= escritura, el maestro va a enviar datos al esclavo

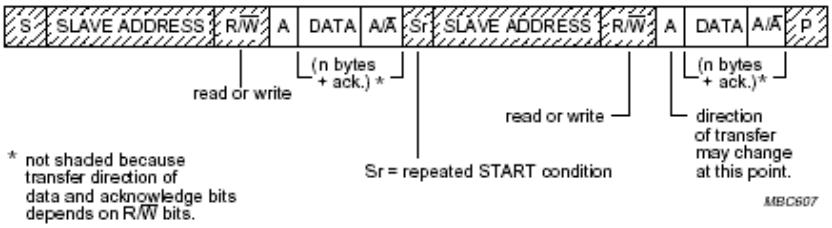
Maestro receptor y esclavo emisor

En el primer byte, el maestro es emisor ya que es él el que indica la dirección del esclavo con el que quiere realizar la transferencia. Cuando el esclavo devuelva el ACK, el maestro pasará a ser receptor y recibirá los datos enviados por el esclavo emisor.
El primer ACK lo genera siempre el esclavo como respuesta a la dirección, y el resto de ACKs el maestro. La condición de *Stop* la genera el maestro, que previamente envía un No Ack (/A).



Formato combinado

En este tipo de transferencia, el sentido de la información puede cambiar. Para ello es suficiente repetir la condición de *Start (Sr en el esquema)* y proporcionar la dirección del esclavo, conmutando el bit que indica el sentido de la transferencia. De ese modo se realizan dos transferencias consecutivas sin que entre ellas se envíe la condición de *Stop*.



Sr = condición de Start repetida

Parámetros de tiempo:

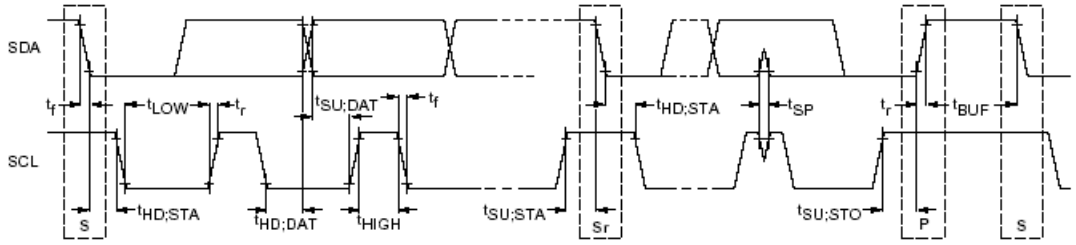
PARAMETER	SYMBOL	STANDARD-MODE		FAST- MODE		UNIT
		MIN.	MAX.	MIN.	MAX.	
SCL clock frequency	fSCL kHz	0	100	0	400	
Hold time (repeated) START condition. After this period, the first clock pulse is generated	tHD;STA μs	4.0	–	0.6	}	
LOW period of the SCL clock	tLOW μs	4.7	–	1.3	–	
HIGH period of the SCL clock	thigh μs	4.0	–	0.6	–	
Set-up time for a repeated START condition	tSU;STA μs	4.7	–	0.6	–	
Data hold time:	tHD;DAT					
for CBUS compatible masters	μs	5.0	–	–	–	
for I2C-bus devices	μs	0(2)	3.45(3)	0(2)	0.9(3)	
Data set-up time	tSU;DAT	250	}	100(4)	–	ns
Rise time of both SDA and SCL signals	tr	–	1000	20+0.1Cb(5)	300	ns
Fall time of both SDA and SCL signals	tf	–	300	20+0.1Cb(5)	300	ns
Set-up time for STOP condition	tSU;STO μs	4.0	–	0.6	–	
Bus free time between a STOP and START condition	tBUF μs	4.7	–	1.3	–	
Capacitive load for each bus line	Cb pF	–	400	–	400	
Noise margin at the LOW level for each connected device (including hysteresis)	VnL	0.1VDD	–	0.1VDD	–	V
Noise margin at the HIGH level for each connected device (including hysteresis)	VnH	0.2VDD	–	0.2VDD	–	V

Notes

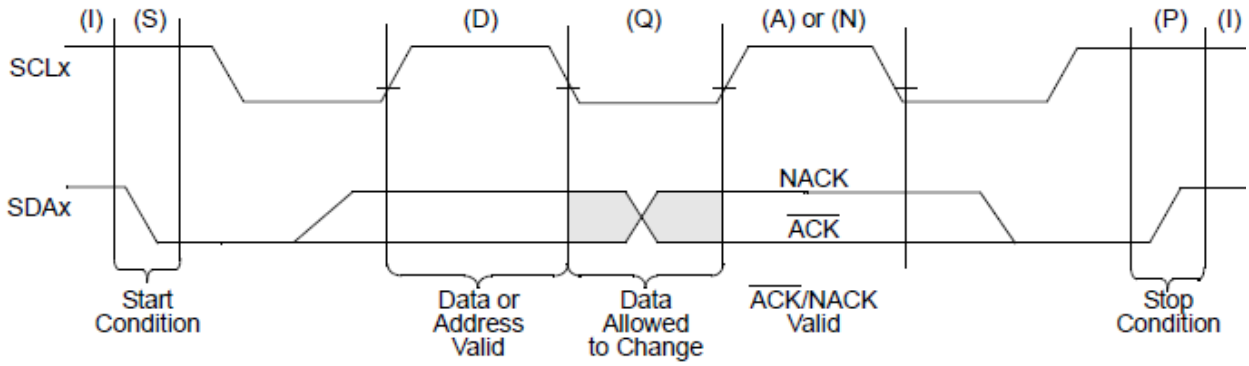
2. A device must internally provide a hold time of at least 300 ns for the SDA signal (referred to the VIHmin of the SCL signal) to bridge the undefined region of the falling edge of SCL.
3. The maximum tHD;DAT has only to be met if the device does not stretch the LOW period (tLOW) of the SCL signal.
4. A Fast-mode I2C-bus device can be used in a Standard-mode I2C-bus system, but the requirement tSU;DAT 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal.

If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line tr max + tSU;DAT = 1000 + 250 = 1250 ns (according to the Standard-mode I2C-bus specification) before the SCL line is released.

5. Cb = total capacitance of one bus line in pF. If mixed with Hs-mode devices, faster fall-times according to Table 6 are allowed.



RESUMEN DE ESTADOS DEL PROTOCOLO DEL BUS I2C



START DATA TRANSFER (S): Partiendo de un estado de bus inactivo (SCL y SDA en alta), una transición de alta a baja de la línea de SDAx mientras el reloj (SCLx) está en alta determina el comienzo de una transferencia (condición de *Start*). Todas las transferencias de datos deben ser precedidas por una condición de *Start*.

STOP DATA TRANSFER (P): Una transición de baja a alta de la línea de SDAx mientras el reloj (SCLx) está en alta determina el final de una transferencia (condición de *Stop*). Todas las transferencias de datos se deben finalizar con una condición de *Stop*.

REPEATED START (Sr): Después de un estado de espera, una transición de alta a baja de la línea de SDAx mientras la línea de reloj (SCLx) está en alta determina una condición de Reinicio (*Re-Start*). Re-Inicios repetidos permiten a un maestro cambiar el sentido de la transferencia (cambiando el bit RW) o cambiar el esclavo (proporcionando la nueva dirección) sin renunciar al control del bus.

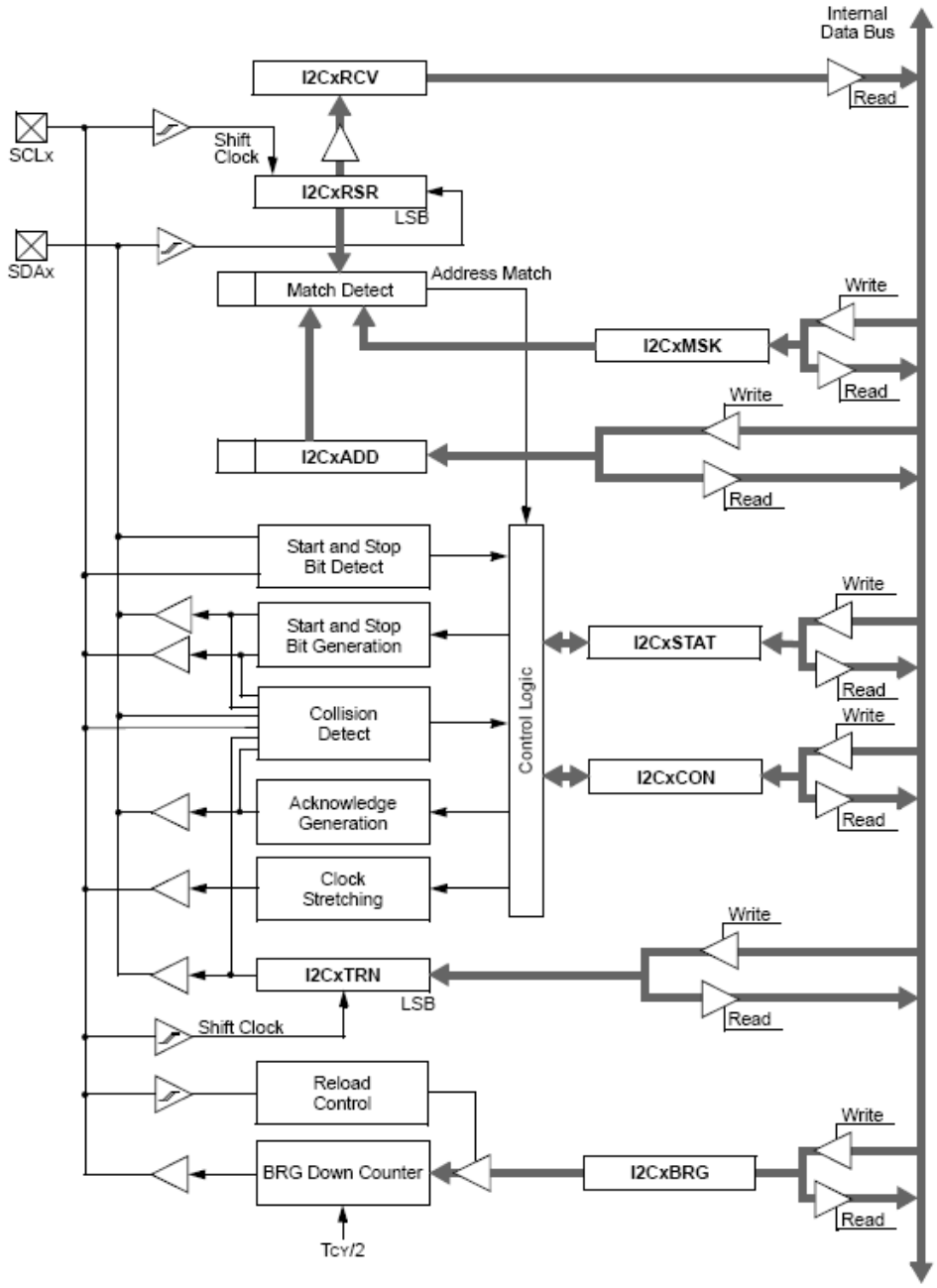
DATA VALID (D): Después de una condición de *Start*, la información en la línea SDAx (dirección del esclavo o dato transmitido) se puede dar por válida siempre que el nivel de la linea SDAx sea estable mientras la señal de reloj esté en alta. De ese modo, en cada ciclo de la señal SCLx se tranmite un bit de datos.

ACKNOWLEDGE (A) OR NOT-ACKNOWLEDGE (N): Todas las transmisiones de cada byte deben ser reconocidos (ACK) o No-Reconocidos (NACK) por el receptor. Para ello, el receptor pone la línea SDAx en baja para indicar el reconocimiento (ACK) o la mantiene en alta para indicar un NACK. La transmisión de ese indicación dura lo mismo que la transmisión de un bit, es decir, un ciclo de reloj de SCLx.

WAIT/DATA INVALID (Q): La información en la línea SDAx (la tensión) sólo puede cambiar durante el intervalo en baja de la señal de reloj SCLx. Los dispositivos también pueden alargar el tiempo del reloj forzando que la línea SCLx se mantenga en baja; generan así un estado de espera.

BUS *IDLE* (I): SCLx y SDAx se mantienen en alta después de una condición de *Stop* y antes de una condición de *Start*.

MÓDULO I2C™ del PIC24H: DIAGRAMA DE BLOQUES (x = 1 o 2)



Registros del I2C

I2CxCON: registro de control, configura el modo de operación.
I2CxSTAT: registro de estado, contiene los *flags* de estado (son solo de lectura).
I2CxBRG: contiene el valor con el que se recarga el generador de baudios.
I2CxADD: contiene la dirección del esclavo asociada al PIC24 en modo esclavo.
I2CxMSK: contiene la mascaras de bits del registro I2CxADD, de forma que el PIC pueda responder, en modo esclavo, a múltiples direcciones.
I2CxRCV: registro donde se reciben los datos (modo receptor).
I2CxTRN: registro donde se escriben los datos a enviar (modo emisor).

El módulo I2C, solo tiene un buffer de un dato para la transmisión y otro para la recepción. Además, el control del bus debe hacerse secuencialmente, en tiempo real, es decir, el hardware no guarda las operaciones a realizar: Start, dirección, ACK, etc. Cada operación hay que activarla una vez terminada la anterior.

I2CxCON: Registro de Control de I2Cx

R/W-0	U-0	R/W-0	R/W-1, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC
I2CEN	**-**	**I2CSIDL**	**SCLREL**	**IPMIEN**	**A10M**	**DISSLW**	**SMEN**	**GCEN**	**STREN**	**ACKDT**	**ACKEN**	**RCEN**	**PEN**	**RSEN**	**SEN**
Bit15							Bit8	Bit7							bit0

Legend: R= Readable bit W= writable bit HC= Hardware Cleared

I2CEN: Habilita el módulo I2C y configura las patas SDA y SCL Cuando se habilita el módulo, el hardware sobrescribe el estado y dirección de los pines:
1 = Módulo I2Cx habilitado y patas SDA y SCL configuradas / 0 = I2C Deshabilitado

I2CSIDL: Configura en modo *Idle:*
1 = Módulo parado en modo *Idle* / 0 = Módulo continua en modo *Idle*

SCLREL: Alarga la señal de SCLx (funcionando como I2C esclavo)
1 = Libera la señal de reloj SCLx / 0 = Mantiene la señal de reloj SCLx en baja
Si STREN = 1, el bit es R/W (ej.: el software puede escribir '0' para mantener SCL y '1' para liberarla. El hardware lo borra al comienzo de la transmisión del esclavo y al final de la recepción del esclavo
Si STREN = 0, el bit es R/S (ej.: el software solo puede escribir '1' para liberar el reloj). El hardware lo borra al inicio de la transmisión del esclavo

IPMIEN: Habilita modo IPMI, configura al dispositivo como un esclavo que recoge toda la información del bus y la replica en una estructura de red distribuida
1 = IPMI habilitado /*0 = IPMI deshabilitado*

A10M: Dirección esclavo de 10 bits
1 = Reg. I2CxADD es una dirección de esclavo de 10 bits
0 = Reg. I2CxADD es una dirección de esclavo de 7 bits

DISSLW: Controla el *Slew Rate* requerido a 400kHz
1 = Control de *Slew Rate* deshabilitado / 0 = Control de *Slew Rate* habilitado

SMEN: Selecciona nivel de entrada para modo SMBus
1 = Permiten umbrales de I/O de la pata compatible con la especificación de SMBus
0 = Deshabilita umbrales de entrada para SMBus

GCEN: Habilitación de llamada general (solo en modo ESCLAVO)
1 = Habilita interrupción ante la recepción de una llamada general (con módulo en recepción)
0 = Deshabilita interrupción

STREN: Habilita atrapar la señal de reloj SCL en modo ESCLAVO; usada junto a SCLREL
1 = Habilita / 0 = Deshabilita

ACKDT: Fija valor del Ack en modo MAESTRO y solo en la recepción
1 = Envía NACK durante la secuencia de ACK / 0 = Envía ACK

ACKEN: Habilita la secuencia de ACK en una operación de recepción en modo MAESTRO
1 = Inicia la secuencia de ACK, se pone a '0' al final de la secuencia
0 = Secuencia de Ack no en proceso

RCEN: Habilita Recepción, en modo MAESTRO
1 = Iniciada, el hardware lo borra al recibir los 8 bits de datos
0 = Secuencia de recepción no en proceso

PEN: Habilita la condición de Stop, en modo MAESTRO
1 = Iniciada, el hardware lo borra al finalizar la secuencia
0 = Condición de Stop no en proceso

RSEN: Habilita condición de *Re-Start*, en modo MAESTRO
1 = Iniciada, el hardware lo borra al finalizar la secuencia
0 = Condición de *Re-Start* no en proceso

SEN: Habilita condición de *Start*, en modo MAESTRO
1 = Iniciada, el hardware lo borra al finalizar la secuencia
0 = Condición de *Start* no en proceso

I2CxSTAT: Registro de Estado de I2Cx

R-0, HSC	R-0, HSC	U-0	U-0	U-0	R/C-0, HS	R-0, HSC	R-0, HSC	R/C-0, HS	R/C-0, HS	R-0, HSC	R/W-0, HC	R/W-0, HC	R-0, HSC	R-0, HSC	R-0, HSC
ACKSTAT	**TRSTAT**	**-**	**-**	**-**	**BCL**	**GCSTAT**	**ADD10**	**IWCOL**	**I2COV**	**D/A**	**P**	**S**	**R/W**	**RBF**	**TBF**
Bit15							Bit8	Bit7							bit0

Legend: R= Readable bit W= writable bit HS= Hardware Set HSC= Hardware Set/Cleared

ACKSTAT: Estado del bit ACK, el hardware lo activa/desactiva al final de la secuencia ACK
1 = recibido NACK / 0 = recibido ACK

TRSTAT: Estado de la transmisión, en modo MAESTRO. El hardware lo activa al inicio de una transmisión del maestro y lo borra al final del ACK del esclavo
1 = Transmisión del Maestro en proceso (8 bits + ACK)
0 = Transmisión no en proceso

BCL: Detecta colisión de arbitraje de bus, se puede producir en la condición de *Start*, *Re-Start*, Dirección, Dato, Ack y Stop.
1 = Se ha detectado una colisión durante una operación del maestro
0 = No colisión

GCSTAT: Indica si ha llegado una Llamada General
1 = Recibida / 0 = No recibida

ADD10: Indica cuando llega una dirección de 10-bits
1 = Recibida / 0 = No recibida

IWCOL: Detecta colisión por escritura, **se debe borrar por software**
1 = Se ha escrito en el reg. I2CxTRN cuando el módulo está ocupado
0 = No colisión

I2COV: Detecta sobre-escritura (*overflow*) en recepción, **se debe borrar por software**
1 = Se ha recibido un dato sin haber leído de I2CxRCV el dato anterior
0 = No *overflow*

D/A: Indica si se ha recibido Dato/Dirección, en modo ESCLAVO
1 = Ultimo byte recibido es dato / 0 = Ultimo byte recibido es dirección

P: Bit Stop
1 = Detectado bit de Stop / 0 = No detectado

S: Start bit
1 = Detectado *bit* de *Start* (o de *Re-Start*) / 0 = No detectado

R/W: Indica *Read/Write* de dato en modo ESCLAVO
1 = *Read*: dato sale del esclavo / 0 = *Write*: dato entra al esclavo

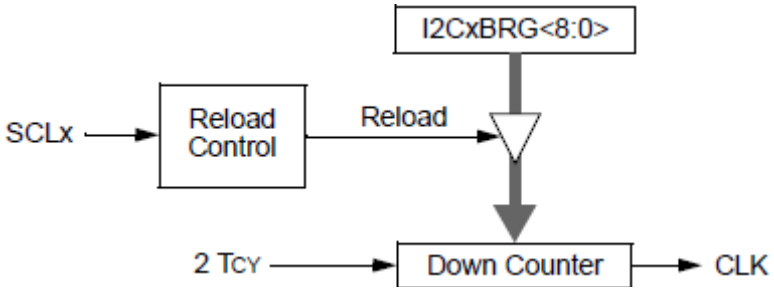
RBF: Estado del buffer receptor, se borra cuando se lee I2CxRCV
1 = Dato recibido, reg, I2CxRCV lleno
0 = Recepción no completada, reg. I2CxRCV vacío

TBF: Estado del buffer transmisor, se activa cuando se escribe en I2CxTRN, el hardware lo borra al terminar la transmisión
1 = Transmisión en proceso, reg. I2CxTRN lleno
0 = Transmisión completa, I2CxTRN vacío

VELOCIDAD DE TRANSFERENCIA (*Baud rate*)

El generador de baudios está compuesto de un contador asociado al reloj de la CPU y un registro I2CBRG que contiene el valor de recarga de dicho contador.

Diagrama de bloques:



Ecuación de BRG:

$I2CxBRG = [(F_{CY}/F_{SCL}) - (F_{CY}/10^7)-1]$

F_{SCL} = Frecuencia de I2C deseada *I2CxBRG es de 9 bits*

Secuencia de UN MAESTRO
1- Espera a que el bus I2C está libre
2- Comprueba y/o limpia los posibles bits de error
3- Genera *Start*
4- Envía dirección del esclavo y señal de operación (lectura o escritura)
5- Espera y verifica un ACK del esclavo
6- Envía o recibe el dato o datos, comprobando la recepción del Ack o enviándolo si es receptor
7- Genera Stop
En todo momento se deben comprobar los bits de error y responder en consecuencia.

INTERRUPCIONES

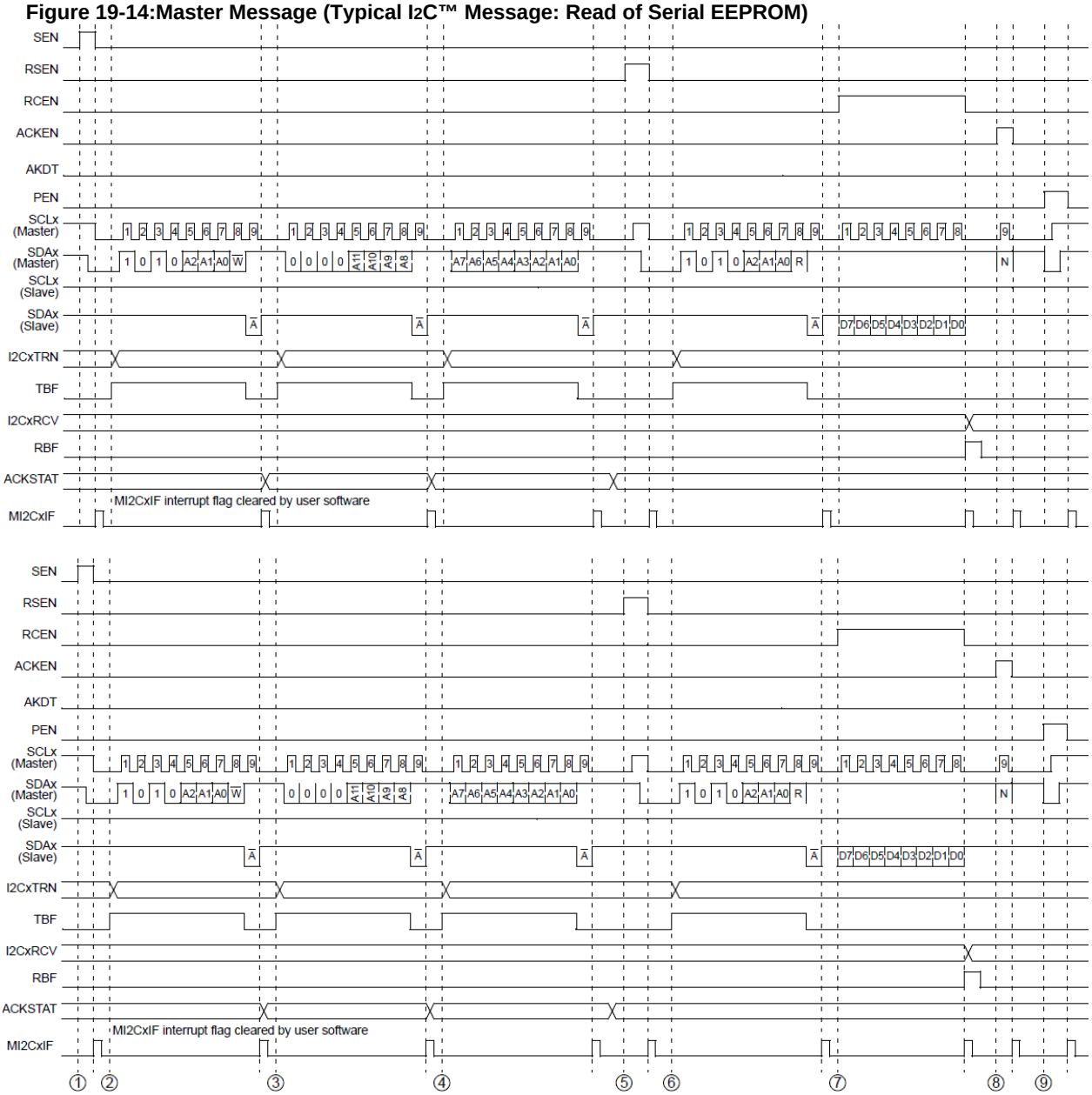
El módulo genera dos interrupciones. Una se asocia a eventos del maestro (MI2CxIF) y otra a eventos esclavo (SI2CxIF).

MI2CxIF se activa al completarse cada uno de los siguientes eventos:
- Condición *Start*
- Condición Stop
- Transferencia de byte transmitido ó recibido
- Transmisión de ACK
- *Re-Start*
- Detección de colisión en el bus → BCL (I2C1STAT)

SI2CxIF se activa al completarse cada uno de los siguientes eventos:
- Detección de una dirección valida
- Petición para transmitir dato (ACK) o para terminar la transmisión (NACK)
- Recepción de dato

MI2C1IF → IFS1	MI2C2IF → IFS3	SI2C1IF → IFS1	SI2C2IF → IFS3
MI2C1IE → IEC1	MI2C2IE → IEC3	SI2C1IE → IEC1	SI2C2IE → IEC3
MI2C1IP → IPC4	MI2C2IP → IPC12	SI2C1IP → IPC4	SI2C2IP → IPC12

Figure 19-14:Master Message (Typical I2C™ Message: Read of Serial EEPROM)



1 Setting the SEN bit starts a Start event.

2 Writing the I2CxTRN register starts a master transmission. The data is the serial EEPROM device address byte, with R/W status bit clear, indicating a write.

3Writing the I2CxTRN register starts a master transmission. The data is the first byte of the EEPROM data address.

4 Writing the I2CxTRN register starts a master transmission. The data is the second byte of the EEPROM data address.

5 Setting the RSEN bit starts a Repeated Start event.

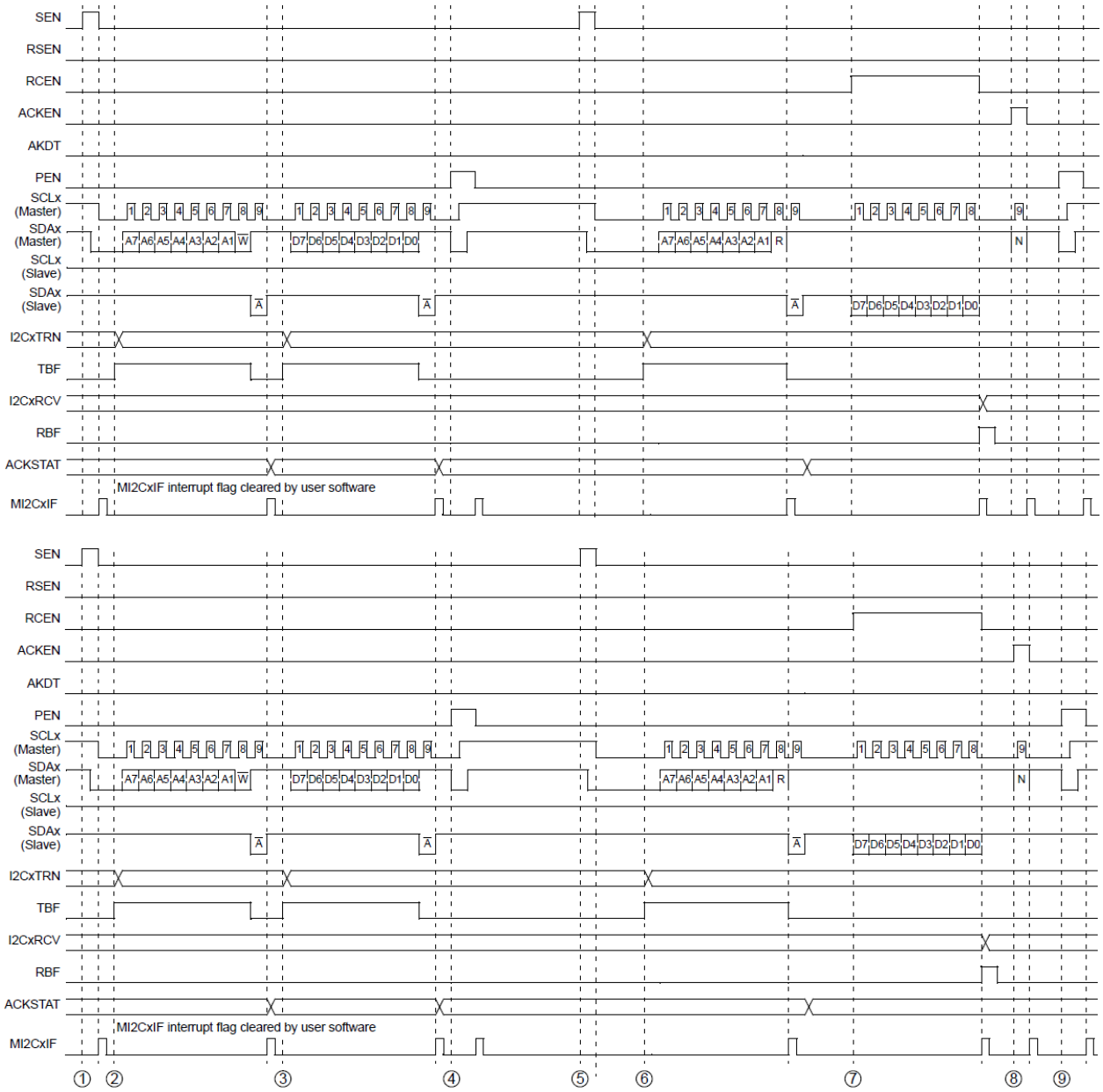
6 Writing the I2CxTRN register starts a master transmission. The data is a re-send of the serial EEPROM device address byte, but with R/W status bit set, indicating a read.

7 Setting the RCEN bit starts a master reception. On interrupt, the user software reads the I2CxRCV register, which clears the RBF status bit.

8 Setting the ACKEN bit starts an Acknowledge event. ACKDT = 1 to send NACK.

9 Setting the PEN bit starts a master Stop event.

Figure 19-15:Master Message (7-bit Address: Transmission and Reception)



1 Setting the SEN bit starts a Start event.
2 Writing the I2CxTRN register starts a master transmission. The data is the address byte with R/W status bit clear.
3Writing the I2CxTRN register starts a master transmission. The data is the message byte.
4 Setting the PEN bit starts a master Stop event.
5 Setting the SEN bit starts a Start event. An interrupt is generated on completion of the Start event.
6 Writing the I2CxTRN register starts a master transmission. The data is the address byte with R/W status bit set.
7 Setting the RCEN bit starts a master reception.
8 Setting the ACKEN bit starts an Acknowledge event. ACKDT = 1 to send NACK.
9 Setting the PEN bit starts a master Stop event.