TEMA 5

EL BUS I2C

5.1 INTRODUCCIÓN

El acrónimo I2C o I^2C significa *Inter Integrated Circuit*; es decir, que cuando se habla del bus I2C se quiere significar un bus cuyo ámbito de aplicación es la *comunicación* entre circuitos integrados. En este momento tal vez el lector pueda extrañarse de tal concepto, puesto que posiblemente piense en la manera *natural* en la que se interconectan diversos circuitos integrados. Por ejemplo, si se considera un microprocesador y el tipo de circuitos que habitualmente forman parte de su sistema (memorias RAM, EEPROM, y periféricos como conversores A/D y D/A, puertos paralelos, pantallas alfanuméricas de cristal líquido, etcétera) entonces automáticamente se pensará en que el modo normal de conectar dichos circuitos es haciendo uso del bus paralelo al nivel de componente o de sistema. Efectivamente, esto permite conseguir el máximo rendimiento en términos de velocidad de la unidad central de procesos. Pero piénsese ahora no tanto en un microprocesador y circuitería asociada sino en un sistema de control basado en microcontrolador; en este caso muy frecuentemente el µC ya contiene toda la circuitería periférica que requiere el sistema, de manera que no se necesitará circuitería externa. No obstante, también es frecuente que el sistema que se pretenda diseñar requiera circuitería periférica con características de las que no dispone ningún miembro de la familia de controladores que se utiliza. Esto implica la necesidad de generar externamente al µC los buses de dirección, de datos y de control para así poder conectarle externamente los circuitos oportunos. Esta cuestión, que conceptualmente está trillada y no presenta ningún tipo de dificultad, resulta un inconveniente en ciertos casos. ¿En cuáles? Pues en aquellos en los que es de primordial importancia minimizar el tamaño del diseño. No se pierda de vista que un circuito integrado cuantas más señales contemple mayor será el número de patillas que necesite y por tanto mayor será el tamaño del encapsulado utilizado y mayor espacio de placa de circuito impreso requerirá el trazado de las pistas de interconexión; por no hablar del coste. Por otro lado, cuanto mayor número de señales circulan por una placa de circuito impreso o interconectan sistemas tanto mayor es la posibilidad de emisión o captación de EMI.

La clave del bus I2C está en la manera de ver los circuitos integrados. Aquí el concepto de *comunicación entre sistemas* se aplica a su nivel más bajo; es decir, cualquier tipo de circuito integrado, sea de la naturaleza que sea, se considera como *un sistema* cuyo modo de funcionamiento es *controlable* y que además requiere un canal de comunicación bidireccional con el sistema que lo controla. Este canal, que con ser semibidireccional es suficiente, se utilizará por un lado para que el sistema *central* defina cómo se comportará tal circuito –es decir, para transmitir *órdenes*– y por el otro lado para transmitir los datos que sean precisos en uno u otro sentido. Si se piensa en un módulo conversor dual A/D y D/A podrá captarse la idea: el conversor A/D podrá tener varios modos de funcionamiento (conversión única o continua, y en este último caso la frecuencia de la conversión); en lugar de usar señales específicas se utilizaría el canal de comunicación para transmitir una orden de configuración operativa. Por

otra parte, una vez realizada una conversión es necesario que el CAD envíe el dato convertido al procesador; y de igual manera éste deberá enviar al CDA el dato a convertir a analógico.
Aunque se haya captado la idea, tal vez no se acabe de ver la ventaja de esta aproximación al problema. En cualquier caso, podrá pensarse, se necesitarán las señales que implementen el canal de comunicación así como las demás necesarias para sincronizar de alguna manera las transferencias por él de órdenes y de datos, con lo que la economía de señales no parece evidente en comparación con las que poseen los circuitos *normales*. Craso error, puesto que el número de señales que necesita un determinado bus de comunicación depende del modo de transporte de la información así como de la funcionalidad deseada. En primer lugar, si se opta por una comunicación serie en lugar de paralela entonces se puede ahorrar un gran número de señales en el bus. Quedan ahora por determinar las características funcionales de éste para poder así definir el número imprescindible de señales adicionales de protocolo. En este punto tal vez el lector piense, a semejanza del bus EIA-232, en la posibilidad de utilizar una técnica serie asíncrona totalmente bidireccional. De esta manera existiría una línea de transmisión, otra de recepción y el común de las señales; es decir, se tendría un bus de tres hilos que permitiría el intercambio de información entre sistemas; y en el plano eléctrico podrían utilizarse, por simplicidad, señales con niveles TTL si la distancia máxima del bus no pudiese ser nunca excesiva. En el plano lógico quedaría por definir un protocolo de estructura de mensajes para los distintos tipos de circuitos que se pudiesen conectar. Pues bien, esta solución no es aceptable en este caso por múltiples motivos que más adelante se irán analizando, aunque uno de ellos –que no el principal– sería el pobre rendimiento que ofrecería un enlace serie asíncrono en lo que se refiere a la información neta transmitida (no se olvide que cada carácter ha de ir enmarcado por un bit de inicio y otro de parada); por no hablar de la imposibilidad de las topologías multipunto.
Por tanto, ¿qué características debería tener un bus pensado para el enlace entre los circuitos integrados de un sistema microcomputador o microcontrolador? Si se analiza la cuestión, teniendo en mente que el número de señales ha de ser el menor posible, podrá comprenderse que son las siguientes:

- **Será un bus serie síncrono**. El término *síncrono* deberá entenderse como que existirá, además de la línea de datos, una señal de sincronía explícita para validar los datos en el canal serie, y no en el sentido de que en la línea de datos siempre deberá existir un flujo de datos, bien propiamente como tales o bien como medio para mantener la sincronía.

- **El sentido del enlace será semibidireccional**. Es decir, existirá una única línea de datos que podrá utilizarse para el flujo en ambos sentidos, pero no simultáneamente. De esta manera se ahorra el número de señales del bus. Esta limitación es un hecho menor debido al ámbito de aplicación de este tipo de bus, en el que no se necesita la bidireccionalidad.

- **Deberá admitir topologías multipunto**. Es decir, podrán conectarse al bus varios dispositivos, pudiendo actuar varios de ellos como emisores, aunque no simultáneamente.

- **Un mismo dispositivo podrá actuar o como emisor o como receptor**, en distintos momentos.

- **Deberá admitir topologías multimaestro**. Es decir, más de un dispositivo puede intentar gobernar el bus. Un maestro podrá actuar tanto como emisor como receptor; y lo mismo puede decirse de un esclavo. Lo que diferenciará a un maestro de un esclavo es la capacidad de gobierno del bus (suministro de la señal de sincronía).

- **Deberá establecer un mecanismo de arbitraje** para el caso en que simultáneamente más de un maestro intente gobernar el bus.

- **Un maestro podrá funcionar también como un esclavo**. El motivo es que en las topologías multimaestro un maestro que haya ganado el gobierno del bus pueda dirigirse a cualquier otro maestro, que entonces deberá comportarse como un esclavo.

- **Deberá establecer un mecanismo de adaptación de velocidad**, para el caso en que un esclavo no pueda seguir la velocidad impuesta por el maestro.

- **La velocidad de las transferencias, marcada por el maestro, podrá ser variable dentro de unos límites**. Además, por su ámbito de aplicación, la tasa de transferencia máxima podrá ser relativamente pequeña, entre 100 y 400 kilobits por segundo normalmente.

- **Deberá establecer un criterio de *direccionamiento* abierto** por medio del cual se sepa a qué esclavo se dirige un maestro. Además, la *dirección* asignada a un circuito podrá ser modificada dinámicamente (no confundir el concepto de *direccionamiento de sistema* con el de direccionamiento de una memoria). Por otra parte, el mecanismo de *direccionamiento* deberá admitir futuras ampliaciones del número máximo de elementos direccionables.

- **El formato de las tramas transmitidas habrá de ser suficientemente flexible** como para poder adaptarse a la funcionalidad de cualquier tipo de circuito pasado, presente o futuro.

- **Desde el punto de vista de la capa física de la interfaz, deberá admitir la conexión de circuitos de diferentes tecnologías** (diversas bipolares, MOS, etcétera).

El bus I2C fue desarrollado por Philips al inicio de la década de 1980, teniendo en mente todos estos aspectos, y ha encontrado una gran aceptación en el mercado. Actualmente los principales fabricantes de dispositivos semiconductores ofrecen circuitos que implementan un bus I2C para su control.
El *Comité I2C* es el encargado, entre otras cuestiones, de asignar las *direcciones* I2C a cada tipo de circuito. De esta manera cada función implementada, independientemente del fabricante, posee la misma dirección; es decir, circuitos que realicen funciones equivalentes deberán poseer la misma *dirección oficial* I2C independientemente del fabricante.
Antes de continuar es conveniente aclarar una cuestión; se ha dicho que cualquier circuito integrado puede poseer un bus I2C, y por tanto existen circuitos de memoria RAM y ROM (EEPROM) pero su finalidad no estriba en contener datos normales y programa sino parámetros funcionales que han de accederse de tarde en tarde. En el caso de la EEPROM lo común es utilizarla para contener parámetros que definen la configuración del sistema y que han de retenerse en ausencia de alimentación (por ejemplo, las sintonías en un aparato de TV). Los circuitos de RAM y ROM habituales serán, por supuesto, de tipo paralelo tradicional; dicho de otra manera, las memorias I2C **nunca** se encuentran mapeadas en la memoria del sistema.

5.2. EL PROTOCOLO DEL BUS I2C

El bus I2C sólo define dos señales, además del común:

- **SDA**. Es la línea de datos serie (***S**erial **DA**ta*, en inglés), semibidireccional. Eléctricamente se trata de una señal a colector o drenador abierto. Es gobernada por el emisor, sea éste un maestro o un esclavo.

- **SCL**. Es la señal de sincronía (reloj serie, o ***S**erial **CL**ock* en inglés). Eléctricamente se trata de una señal a colector o drenador abierto. En un esclavo se trata de una entrada, mientras que en un maestro es una salida. Un maestro, además de generar la señal de sincronía suele tener la capacidad de evaluar su estado; el motivo –como se verá– es poder implementar un mecanismo de adaptación de velocidades. Esta señal es gobernada única y exclusivamente por el maestro; un esclavo sólo puede *retenerla* o *pisarla* para forzar al maestro a ralentizar su funcionamiento, cosa que se explicará más adelante.

Esta particularidad física de que las salidas de los excitadores I2C hayan de ser a colector o drenador abierto no es casual sino que resulta de vital importancia para que a este bus con tan sólo una señal de datos y otra de sincronía se le pueda dotar de todas las características funcionales apuntadas en el apartado anterior. Recuérdese que un dispositivo lógico con este tipo de salida permite realizar la función producto lógico con una simple conexión eléctrica (montaje *Y por conexión*). Evidentemente, un enlace I2C necesitará sendas resistencias de elevación en las respectivas líneas SDA y SCL. De esta manera un excitador I2C realmente sólo gobierna el estado 0 lógico en las líneas I2C, mientras que el estado 1 lógico no es suministrado por el excitador directamente sino por medio de la oportuna resistencia de elevación. Así, el concepto de *aislamiento del bus* se consigue con tal sólo hacer que el

excitador del nodo que se desea aislar ofrezca a su salida el estado alto (1 lógico), al que le corresponde una elevadísima impedancia de salida, equivalente a un aislamiento eléctrico con respecto al bus.

5.2.1 Los estados del bus I2C

El bus I2C puede encontrarse en distintos estados, que la norma define como los siguientes:

- **Libre**. Este estado se caracteriza por encontrase las líneas SDA y SCL a 1, sin que se esté realizando ningún tipo de transferencia.

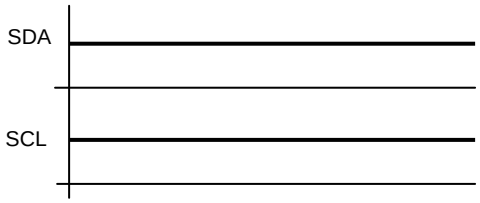


Fig. 5.1: Condición de bus libre

- **Inicio**. Se produce una condición de *inicio* cuando un **maestro** inicia una transacción. En concreto, consiste en un cambio de alta a baja en la línea SDA mientras SCL permanece a alta. Esto puede apreciarse en el cronograma mostrado en la figura 5.2. A partir de que se dé una condición de inicio se considerará que el bus está ocupado y ningún otro maestro deberá intentar generar su condición de inicio.

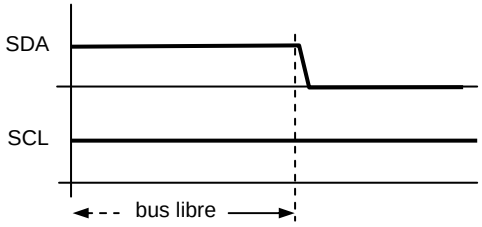


Fig. 5.2: Condición de inicio

- **Cambio**. Se produce una condición de *cambio* cuando, estando a baja la línea SCL, la línea SDA puede cambiar de estado. En la transferencia de datos por un bus I2C éste es el único instante en el que el sistema emisor (que podrá ser tanto un maestro como un esclavo) podrá poner en la línea SDA cada bit del carácter a transmitir.

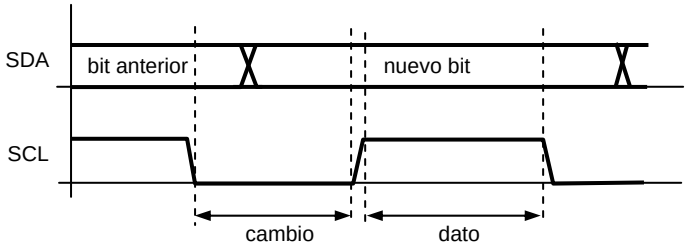


Fig. 5.3: Condiciones o estados de *cambio* y de *dato*

- **Dato**. Este estado es el que, una vez iniciada una transacción, queda definido por la fase alta de la señal de sincronía SCL. En este estado se considera que el dato emitido es válido, y no se admite que pueda cambiar. Recuérdese que, ya iniciada una

transferencia, el único instante en que la línea de datos puede cambiar es en el estado de *cambio*.

- **Parada**. Se produce una condición de *fin* o *parada* cuando, estando la línea SCL a alta, se produce un cambio de baja a alta en la línea SDA. Obsérvese que esto es una violación de la condición de *dato*, y es precisamente por esto por lo que se utiliza para que un maestro pueda indicar al esclavo que se finaliza la transferencia. Tras la condición de parada se entra automáticamente en el estado de *bus libre*.

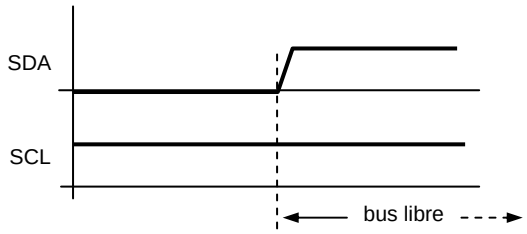


Fig. 5.4: Condición de parada

5.2.2. El formato de una transacción

Entenderemos por *transacción* todos los eventos desarrollados entre una condición de inicio y una de parada. Una vez producido el inicio de una transacción comienza propiamente ésta, y en ella se lleva a cabo la transferencia de uno o varios caracteres, en uno u otro sentido.

- **El carácter de ocho bits es la unidad de transferencia**. El orden de emisión de los bits de un carácter es empezando por el más significativo, siguiendo en orden decreciente de pesos y terminando por el menos significativo (como se ve, es un criterio contrario al que siguen las UART).

- Un carácter puede tener diversos significados en función de quién lo emita y en qué instante lo haga.

- **Tras cada carácter se debe producir un *reconocimiento* por parte del receptor**; el motivo es dotar al protocolo de un mecanismo de seguridad.

- El primer carácter transferido lo emite siempre el maestro; sus siete bits más significativos indican la dirección del esclavo al que se dirige, y el bit de menor peso indica el sentido de la transferencia de los subsiguientes caracteres (0=escritura, 1=lectura, siempre desde el punto de vista del maestro).

- Dentro de una transacción es posible cambiar el sentido del flujo, pero para ello hace falta generar una nueva condición de inicio (*reinicio*) y direccionar de nuevo el mismo esclavo.

Como puede verse, para el direccionamiento de un esclavo se tienen siete bits, lo que daría un máximo de 128 dispositivos I2C distintos. No obstante, en la realidad se dispone de muchas menos direcciones puesto que un cierto número de ellas no se emplean como tales sino como identificadores de función especial, como se verá más adelante; además, ciertos circuitos reservan para sí varias direcciones debido a que es frecuente que en un diseño se tengan que emplear varios del mismo tipo. En estos casos los circuitos tienen una dirección compuesta por dos partes; una fija establecida por el comité I2C y otra variable (bits inferiores) fijada por el diseñador de cada sistema dado. En el encapsulado de estos circuitos existen ciertas patillas (una, dos o tres) mediante las cuales el diseñador puede determinar la parte *programable* de la dirección I2C. Por ejemplo, los circuitos EEPROM tienen asignada una dirección 1010XXX; 1010 es la parte fija de los siete bits, y XXX es la parte programable que corresponde al estado aplicado a tres patillas *ad hoc* de su encapsulado.

Retomando la cuestión de la limitación del número de direcciones, este hecho trajo aparejado el que pocos años después del lanzamiento del bus I2C, y debido a su éxito, se fuesen agotando las direcciones disponibles, lo que implicaba la imposibilidad de ofrecer nuevos circuitos I2C. Afortunadamente, se previó esta contingencia dado que se reservaron para futuras ampliaciones las direcciones que comenzaban por 1111. De esta manera, ahora **existen dos tipos direcciones: las normales de siete bits y las ampliadas de diez bits**. En este último caso el direccionamiento de un esclavo implica la transferencia de dos caracteres; el primero deberá comenzar por 11110XX, siendo XX los dos bits de mayor peso de la dirección de diez bits, y el segundo carácter corresponderá a los ocho bits inferiores de la dirección del esclavo. El sentido de la transacción se sigue indicando por el bit menos significativo del primer carácter de dirección (11110-XX-L/E).
En términos generales, una transacción obedecerá a alguna de las siguientes estructuras:

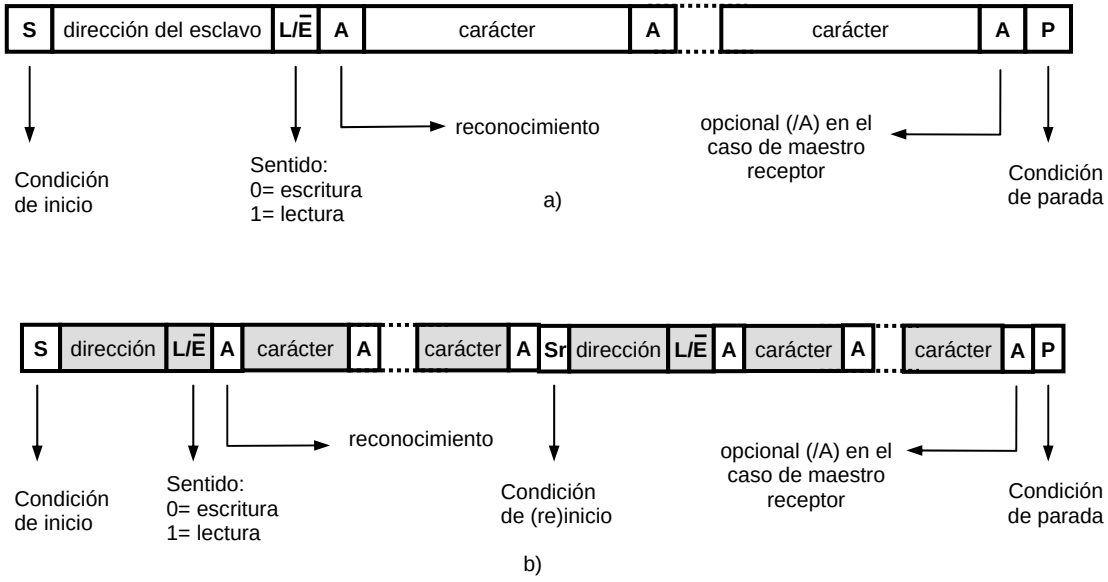


Fig. 5.5: Formatos de una transacción: a) sin cambio de sentido b) con redireccionamiento y posible cambio de sentido

En la figura 5.5 se ha ilustrado el direccionamiento con direcciones de siete bits; si se utilizase un direccionamiento de diez bits entonces la parte del direccionamiento quedaría como se observa en la figura 5.6.

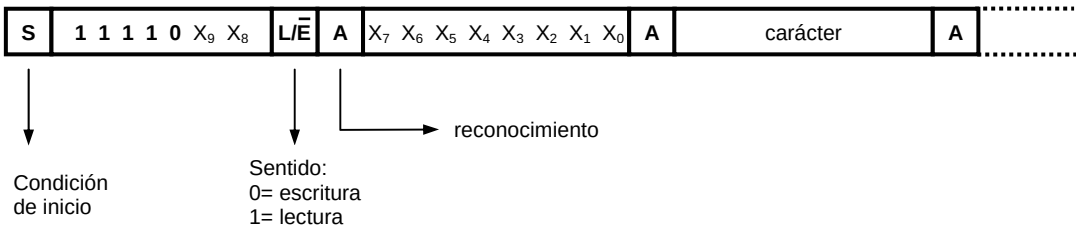


Fig. 5.6: Direccionamiento ampliado, de diez bits

En la figura precedente se indica por $X_9...X_0$ los diez bits de la dirección del esclavo al que se dirige el maestro.

5.2.3. El significado de los caracteres

Como ya se ha anticipado, la norma I2C no indica absolutamente nada sobre el significado de los caracteres que siguen a la fase de direccionamiento. Esto dependerá del circuito I2C de que se trate en cada caso. Por ejemplo, si se tratase de un circuito tipo 24C02, que es una memoria EEPROM, y se desease leer cuatro datos almacenados a partir de la posición relativa 20h desde su origen, entonces la transacción que realizaría el maestro sería la siguiente:

1. Generar la condición de inicio.
2. Direccionar en modo escritura el circuito 24C02 oportuno.
3. Verificar el reconocimiento al direccionamiento.
4. Enviar un carácter de valor 20h, **que sería interpretado por la EEPROM como la dirección a partir de la que se va a hacer un acceso**.
5. Como lo que se va a hacer es leer (recibir), y antes se ha establecido el modo de escritura (transmitir), entonces tras el reconocimiento a 20h se genera una condición de reinicio y se vuelve a direccionar el circuito.
6. Tras verificar el reconocimiento, se solicitan los ocho bits de cada carácter, generando el reconocimiento tras el último (menos significativo) de ellos. Esto se repite cuatro veces. A cada lectura de carácter la EEPROM **irá enviando los contenidos de sus posiciones 20h, 21h, 22h y 23h**.
7. Una vez que el maestro ha leído las cuatro posiciones, no reconocerá el último y generará una condición de parada.

Pero, por supuesto, si se tratase de otro circuito con otro tipo de funcionamiento, entonces el significado de los caracteres no tendría absolutamente nada que ver con lo anteriormente descrito. Los fabricantes de dispositivos I2C ofrecen en sus hojas de características información suficiente como para saber el significado de los caracteres así como qué hacer para conseguir realizar las funciones que tal circuito permita.

5.2.4. El gobierno de la señal de sincronía

Ya se ha indicado que la señal SCL sólo la puede generar el maestro. Si el maestro actúa como emisor entonces la señal de sincronía puede entenderse como una validación de los bits que va volcando en la línea de datos SDA, sean estos bits los de un carácter de dirección o los de un carácter propiamente de datos. Si actúa como receptor entonces SCL sirve no sólo para indicar los instante en los que el maestro toma los bits que le envía el esclavo, sino también para –indirectamente– marcar la velocidad de envío del siguiente bit (el esclavo no debe poner el siguiente bit hasta que se dé una condición de *cambio*, y ésta no se da hasta que finaliza el estado actual de *dato* (esto es, hasta que SCL pasa de alta a baja).
Por otra parte, ya se ha dicho que la unidad de transferencia es el carácter de ocho bits, y que a cada carácter le debe seguir un bit de reconocimiento por parte del receptor. Será siempre el maestro quien validará el bit de reconocimiento, sea éste ofrecido por él mismo (maestro receptor) o por el esclavo (esclavo receptor).
Finalmente, sólo hay una situación en la que un esclavo puede *interferir* la señal SCL gobernada por el maestro. Se trata del mecanismo denominado *sincronización*, por medio del cual un esclavo puede ralentizar la velocidad de transferencia marcada por en maestro. En este caso el esclavo no gobierna el reloj pero interfiere con cierto criterio su estado para que el maestro advierta que se le está pidiendo que vaya más lento. Esto se analizará más adelante.

5.2.5. La filosofía del reconocimiento

Como se ha dicho antes, el establecer la necesidad de que el receptor reconozca cada carácter transmitido obedece al deseo de establecer un mecanismo de seguridad en las transferencias, pero también a algo más.

En primer lugar, el reconocimiento es **obligatorio** para todo esclavo receptor. Si uno no lo hace entonces el maestro emisor considerará que se ha producido un fallo en el bus y deberá establecer los mecanismos oportunos de respuesta a este fallo. Dependiendo de en qué circunstancias se produzca esta ausencia de reconocimiento entonces se podrá obrar de manera distinta. Por ejemplo:

- Si la ausencia de reconocimiento del esclavo receptor se produce en la fase de direccionamiento esto podrá significar varias cosas:

    - Que el circuito se encuentra ausente (removido del bus, o apagado).

    - Que el circuito está ocupado realizando algún tipo de tarea interna que le impide responder al direccionamiento del maestro. Esto podrá tener sentido en algunos dispositivos I2C, pero no en otros. Por ejemplo, en las memorias EEPROM con bus I2C si se les escribe un bloque de datos se tarda un cierto tiempo en llevar a cabo tal proceso, durante el cual no dará el reconocimiento si es vuelta a direccionar para realizar un nuevo acceso.

    - Que el circuito está averiado o ha sucedido alguna anomalía en la línea.

- Pero si esta ausencia se produce en medio de una transferencia de datos entonces esto normalmente será síntoma de una avería funcional (a menos que dé la casualidad de que en ese instante ha sido removido o apagado).

El tipo de medida a tomar por parte del maestro dependerá de cada circunstancia, y podrá ir desde volver a intentar direccionar el esclavo un poco más adelante hasta generar algún tipo de alarma.

En concreto, **el reconocimiento consiste en poner la línea de datos SDA a baja durante el impulso SCL que sigue al del último bit de un carácter; y esto lo debe realizar el receptor**. En el instante que toca el reconocimiento el emisor deberá aislar del bus su línea SDA; en los bits de los caracteres (estados de datos) es él quien la gobierna –no se olvide que actúa como emisor– pero no sucede tal cosa en el reconocimiento. Aquí quien ha de gobernar ahora SDA es el receptor, para dar el reconocimiento. Por ello, si el emisor no aísla del bus su SDA local, y diese la casualidad de que el bit menos significativo, y último del carácter transmitido, fuese un 0, entonces podría suceder que si el receptor no diese su reconocimiento el maestro no advertiría tal eventualidad puesto que al sondear el estado de la línea SDA detectaría que está a baja (por el último bit de dato transmitido) e interpretaría erróneamente que ha habido el reconocimiento. En la siguiente figura puede apreciarse lo dicho.

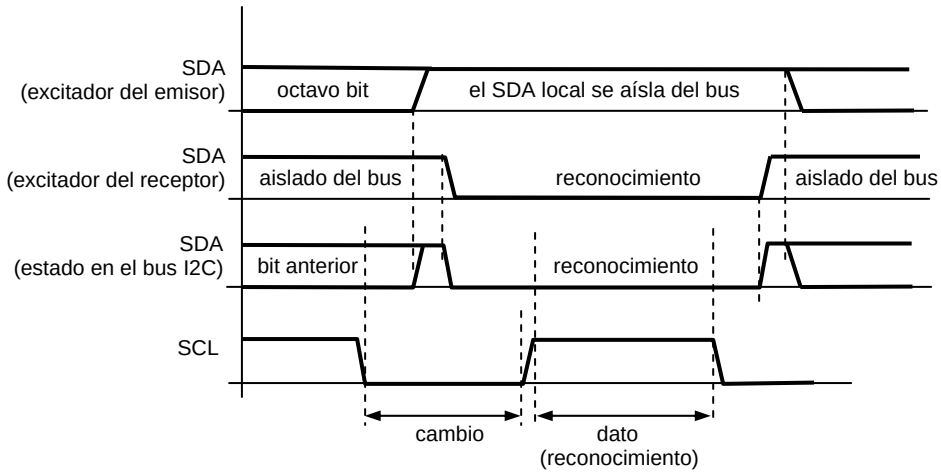


Fig. 5.7: El reconocimiento en el bus I2C

En la figura se ha mostrado el estado de los excitadores SDA del emisor y del receptor I2C. No se pierda de vista que la capa física de la interfaz I2C necesita la existencia de un excitador y de una electrónica receptora tanto para SDA como para SCL. Cuando alguien actúa como receptor, debe aislar del bus su excitador SDA (ofrecer un 1 lógico, como ya se ha dicho), y también SCL si es un esclavo, para que el excitador del emisor sea quien gobierne la línea SDA del bus. Pero cuando llega la hora de dar el reconocimiento entonces es el excitador del emisor quien debe aislarse para que, ahora sí, sea el del receptor quien aplique un nivel a baja en SDA. De esta manera, si se observa la figura 5.7 puede verse que la señal en la línea SDA del bus I2C es el producto lógico de las señales aplicadas por los excitadores de todos y cada uno de los circuitos conectados al bus. Tal y como ya se ha comentado, ésta es una propiedad de los excitadores con salida a colector o drenador abierto.

Aunque se ha dicho que el reconocimiento es obligatorio por parte del receptor, **existe una excepción a esta regla: cuando el receptor es un maestro entonces es posible no dar el reconocimiento al último carácter**, justo antes de generar la condición de parada. Para entender el motivo de esto es necesario recordar que es el maestro quien inicia una transacción, la gobierna y la finaliza. Por tanto, aunque un maestro actúe como receptor sabrá perfectamente cuándo no quedan más datos por recibir. Dicho de otra manera, un esclavo emisor si tiene algo que enviar no lo hará por propia iniciativa sino porque el maestro receptor se lo solicite, y en este caso el maestro receptor sabrá perfectamente cuántos datos desea tomar. Supóngase ahora un caso concreto en el que se desea leer un bloque de siete datos de una memoria EEPROM. Podrá pensarse que esta circunstancia no planteará mayor problema dado que bastará, al recibirse el séptimo carácter, reconocerlo y generar entonces una condición de parada para poder iniciar una nueva transacción con otro circuito I2C. Pues bien, si el maestro receptor reconociese el carácter antes de generar la condición de parada podría suceder que tal condición de parada no se llegase a producir efectivamente en el bus I2C. El motivo es el siguiente: cuando el maestro receptor da su reconocimiento al carácter enviado por el esclavo emisor, éste supondrá que a continuación el maestro le solicitará el primer bit del siguiente carácter, el octavo, dado que no tiene por qué saber cuántos datos le solicitará el maestro. Por ello el esclavo, cuando SCL pase a baja y se produzca un estado de cambio, colocará en la línea SDA el bit más significativo del siguiente dato, y esperará el impulso SCL del maestro. Supongamos que este bit da la casualidad de que es un 0 lógico. ¿Qué sucederá entonces? Pues que el maestro receptor, que desea finalizar la transacción y para ello intenta generar una condición de parada, se encontrará con que no puede hacerlo porque el esclavo emisor no ha liberado la línea SDA del bus. Esto puede apreciarse en la figura 5.8. Es por este motivo por el que es opcional el último reconocimiento de un maestro receptor; como el esclavo emisor **obligatoriamente** tiene que liberar la línea SDA para permitir que el maestro receptor reconozca, pues entonces éste puede, si así lo desea, aprovechar este lapso para tener la garantía de poder generar una condición de parada.

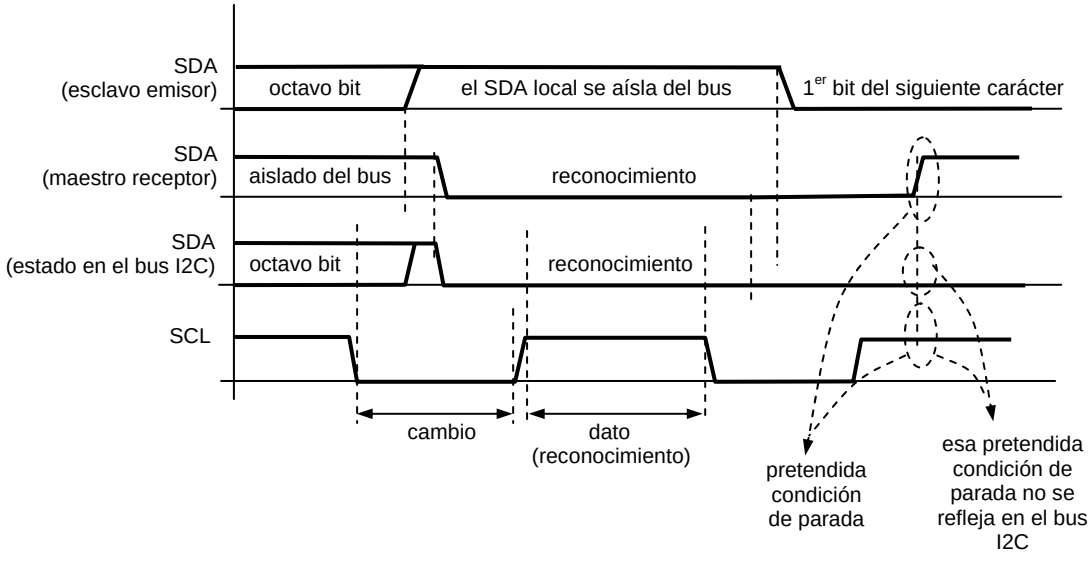


Fig. 5.8: El problema de una parada tras un reconocimiento

Cuando el receptor es un maestro también puede intencionadamente no producirse el reconocimiento en medio de una transacción. Esto se haría para abortarla ante situaciones que hacen aconsejable tal cosa. Supóngase que en medio de una transacción, al leerse un carácter cualquiera de una trama, sucede un evento que hace que el maestro opte por abortar la transacción e iniciar otra con otro dispositivo I2C. Ya se ha visto que si se reconoce ese carácter podría suceder que el maestro no pudiera generar la condición de parada y por tanto no podría iniciar una nueva transacción con el nuevo circuito I2C con el que ahora urge comunicarse. Al no reconocer, se aprovecha este instante para forzar una condición de parada que aborte la transacción en curso, que podrá reanudarse más adelante.

Es necesario advertir que la manera ortodoxa de que un maestro receptor realice con garantía la condición de parada es no aprovechar el instante del reconocimiento sino el siguiente tiempo de bit. En este caso el maestro receptor no reconoce pero aplica el correspondiente impulso de sincronía. Cuando un esclavo emisor detecta que no se ha reconocido el dato enviado, la norma I2C obliga a que justo a continuación aísle del bus su SDA. De esta manera ya el maestro receptor está en condiciones de generar sin problemas su condición de parada. Este segundo método es el que especifica la norma, pero el que se ha comentado en primer lugar resulta igualmente efectivo y más rápido de llevar a cabo (la única precaución que hay que tener es conocer cómo se implementa la interfaz I2C en los esclavos utilizados; si una condición de parada se detecta de manera cableada entonces no habrá problema, pero si se hace por programa entonces se deberá verificar que se comprueba en cualquier instante y no sólo tras cada reconocimiento). Lo comentado puede observarse en la figura 5.9.

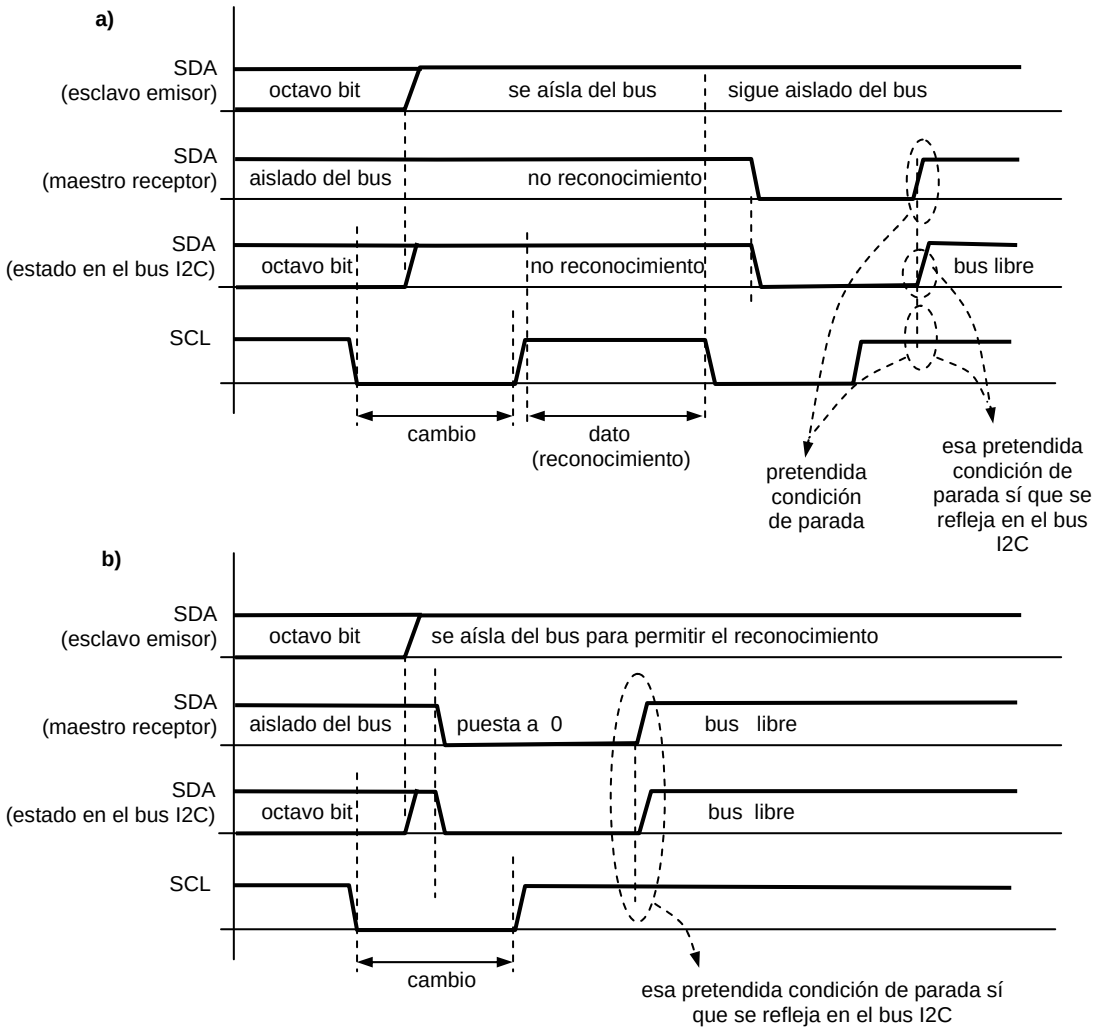


Fig. 5.9: Terminación sin reconocimiento: a) modo normal
b) modo abreviado (puede no ser compatible)

5.2.6. El mecanismo de sincronización

Ya se ha comentado que el bus I2C establece un mecanismo para que un esclavo lento pueda forzar a un maestro rápido a ir más lento; a este mecanismo la norma I2C lo denomina *sincronización*. Se trata de un procedimiento ingenioso que saca provecho de la naturaleza a colector o drenador abierto de la señal de reloj SCL.
Por defecto, un maestro gobernará la señal de reloj del bus, SCL, a la velocidad que considere oportuno. Tan es así que no tiene por qué ser siempre la misma ni siquiera dentro de una misma transacción ni dentro de un mismo carácter. En este sentido tan sólo se han de satisfacer las condiciones mínimas de temporización entre los instantes de cambio de las señales, aspectos éstos que más adelante se detallarán al hablar de las especificaciones eléctricas y temporales de la interfaz I2C.
Por lo dicho, el maestro marcha a su propia velocidad y supondrá que el esclavo con el que se comunique será capaz de seguir el ritmo. En muchas circunstancias esto es así, pero en otras puede resultar que no, bien porque el esclavo de ninguna manera pueda seguir jamás tal velocidad marcada por el maestro, o bien porque en ciertos instantes necesite realizar algunos procesos que le consuman un tiempo que no se pueda dedicar a la comunicación con el maestro. Sea cual sea la causa, el esclavo no estará en condiciones de comunicarse correctamente y se generará un error de comunicación. Para dar mayor flexibilidad al bus, éste admite mecanismos para establecer una especie de diálogo entre las partes para amoldar dinámicamente la velocidad. El mecanismo consiste en lo siguiente: un maestro que admita la *sincronización* deberá comprobar el estado de la línea SCL del bus; si al activarla ésta no se activa efectivamente en el bus entonces querrá significar que el esclavo le está solicitando ralentizar su velocidad. De esta manera, el maestro esperará con su SCL local activada y no considerará que ha comenzado el estado de *dato* hasta que efectivamente la línea SCL del bus esté a alta. Por la otra parte, todo esclavo que admita sincronización, estando la línea SCL del bus a baja podrá *pisarla* (poner a baja su SCL local) durante el tiempo que estime oportuno, y sólo liberará la línea SCL del bus en el instante en que esté preparado para gestionar un nuevo bit o carácter. Todo esto puede observarse en la figura 5.10.

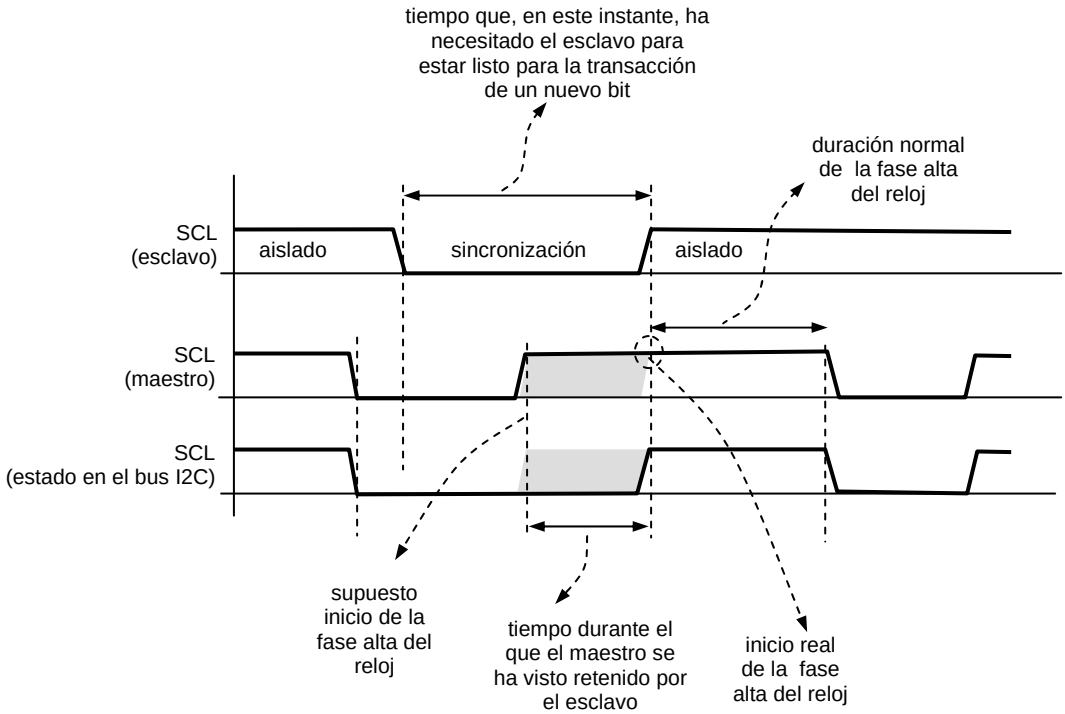


Fig. 5.10: El mecanismo de *sincronización* en el bus I2C

Un esclavo receptor podrá utilizar el mecanismo de sincronización en términos bit a bit, carácter a carácter, o cuando y como lo estime oportuno en función de sus intereses.

5.2.7. Los caracteres de dirección

Ya se ha dicho que el primer carácter que se transmite tras una condición de inicio es la dirección del esclavo. La dirección ya se sabe que es un identificador de siete bits que representa un determinado circuito del bus I2C. El bit menos significativo del carácter de ocho bits indica el sentido de la transferencia de los siguientes caracteres. También se ha dicho que no todas las combinaciones de siete bits del primer carácter están disponibles como direcciones normales I2C. Y, por último, se ha apuntado que existen dos modos de direccionar: con direcciones de siete y con direcciones de diez bits. En el primer caso la dirección se transmite en un único carácter; y en el segundo caso con dos, debiendo ser el primero 11110-XX-L/E.
Este es el momento de comentar el significado de aquellas combinaciones que no son direcciones normales. En la tabla que sigue se resume esta cuestión.

Tabla 5.1: Significado del primer carácter de una transacción

CARÁCTER		FUNCIÓN
DIRECCIÓN	L/E	
0000000	0	Llamada general (*General Call*) que va dirigida a todos los esclavos conectados al bus I2C
0000000	1	Aviso de inicio. Se usa para que los esclavos que implementan la interfaz por programa lento puedan detectar la condición de inicio
0000001	X	Dirección CBUS. En topologías en las que se mezclen circuitos I2C y circuitos CBUS, los I2C deben ignorar este direccionamiento
0000010	X	Reservado para formatos diferentes de bus. Los circuitos I2C que no los admitan deberán ignorar este direccionamiento
0000011	X	Reservado para futuras ampliaciones
00001XX	X	Petición de transacción a alta velocidad (*Hs*)
00010XX (...) 1110XX	X	Direcciones normales asignables a circuitos I2C
11110XX	X	Direccionamiento de diez bits
11111XX	X	Reservado para futuras ampliaciones

5.2.8. La llamada general

La dirección de llamada general se ha implementado para permitir que ciertos *mensajes* que un maestro necesite enviar a todos los elementos conectados al bus se haga de manera colectiva en lugar de individual direccionándolos uno tras otro. Esta cuestión podrá tener sentido en algunos casos pero no en otros. Aquellos circuitos que no tienen nada que hacer ante una llamada general simplemente la ignorarán.
Una llamada general consta de una trama formada por el carácter de la llamada general, el reconocimiento, carácter de tipo de llamada general, y su reconocimiento.
Todos los dispositivos capaces de responder a una llamada general deben reconocer obligatoriamente, pero aquellos que por carecer de sentido no la contemplan deben ignorar toda la transacción y por tanto no reconocer nunca.
El segundo carácter indica qué tipo de llamada general se está realizando. De este octeto tan sólo están definidas algunas combinaciones:

- Valor **00000000**: Está prohibido o reservado.

- Valor **00000010**: Ha sido suprimido.
  La última versión de la norma I2C (2.1) ha suprimido este tipo de llamada general, que antes se utilizaba para indicar a los dispositivos que se les va a fijar por programa la parte programable de su dirección. El motivo de esta eliminación es que esta función es muy

compleja y realmente casi nunca se llegó a utilizar. Ya se ha comentado que hay dispositivos en los que su dirección posee una parte fija y otra variable. Normalmente la parte variable se fija por el diseñador aplicando a ciertas patillas los valores de los bits programables; pero también podían existir dispositivos en los que esta parte programable no se realizaba vía conexión eléctrica sino por medio de comandos de programación a través del propio bus I2C. Por tanto, para realizar esto es necesario hacer una llamada general de este tipo. Cuando se produce, los circuitos implicados entran automáticamente en el modo de programación por programa.

- Valor **00000100**:
  Este tipo de llamada general 4 se utiliza para indicar a los circuitos del bus que recarguen la parte programable vía *hardware*, es decir, vía las patillas del encapsulado dedicadas a indicar la parte programable de su dirección. Esto resulta útil en aquellas topologías en las que un mismo circuito se utiliza en un número que excede el de las direcciones diferentes disponibles. Por ejemplo, supóngase que en un diseño se utilizan seis circuitos del mismo tipo que poseen una dirección del tipo 01000XX; es decir, la parte fija es 01000 y la programable XX cuyo valor se determina mediante dos patillas A_1 y A_0 de su encapsulado. Como puede verse, tan sólo se tiene la posibilidad de asignar a estos circuitos cuatro direcciones diferentes dentro de la misma topología (0100000, 0100001, 0100010 y 0100011) pero se necesitan seis. La solución a este tipo de problema es realizar un diseño con una especie de *conmutación de bancos de circuitos*, de modo que una dirección se reserva para los circuitos de los bancos inactivos, y el resto se asigna a los dispositivos del grupo activo. En el ejemplo que estamos utilizando se podría obrar de la siguiente manera: como sólo existen cuatro direcciones disponibles, una se reserva para no direccionar y el resto, tres, para asignarlas a los circuitos de cada grupo. Por tanto, si en este caso cada grupo está formado por tres circuitos y existen seis del mismo tipo, entonces resulta que hay que formar dos grupos de tres. Si llamamos a cada circuito C0, C1, C2, C3, C4 y C5, y la dirección no utilizada es la 01000**11**, entonces el diseñador tendrá que poder aplicar a las patillas de dirección de cada circuito uno entre dos valores: el que indica no direccionable (**11**) y el asignado para poder direccionarlo (**00**, **01** o **10**, según sea el caso). De este modo, el maestro cada vez que desee acceder a uno de los seis circuitos deberá determinar si se encuentra en el banco activo, y de no ser así aplicar a las patillas programables el valor adecuado y hacer una llamada general de tipo 4. La figura que sigue ilustra cómo se podría hacer este diseño desde el punto de vista físico.

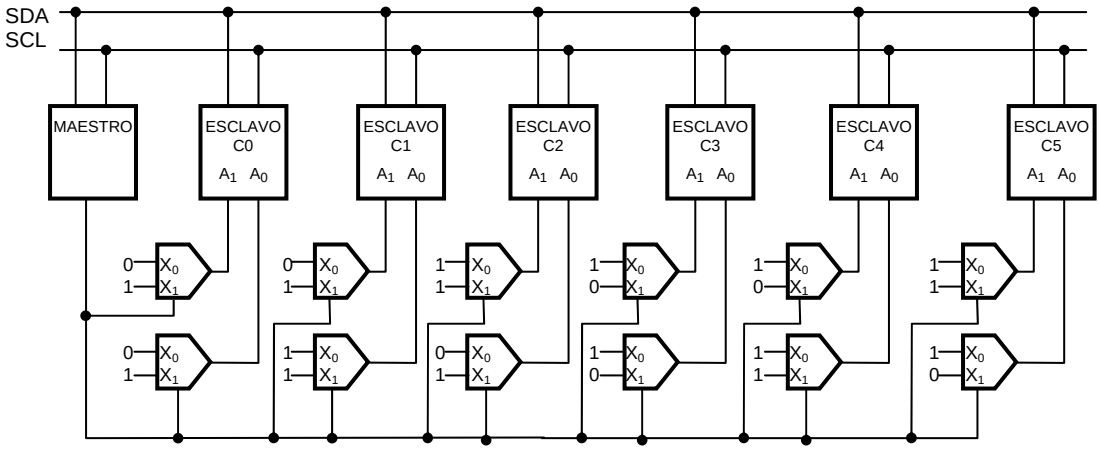


Fig. 5.11: Ejemplo de topología con cambio dinámico de direcciones

El maestro dispone de una señal de control que se aplica a la entrada de selección de unos multiplexores. Si la señal vale 0, se seleccionan los canales cero de todos los multiplexores, con lo que a los tres primeros circuitos, C0, C1 yC2, se les asignan las partes programables de dirección 00, 01 y 10 respectivamente, y a los circuitos C3, C4 y C5 el valor 11; y sucede lo contrario si la señal de control vale 1. Esto implicará que si el

maestro direcciona el dispositivo 0100010 se estará dirigiendo al C2 o al C5 dependiendo del valor de la señal de control. Por supuesto, cuando se conmuta de banco es necesario que los circuitos sean conscientes de que se les ha cambiado su parte de dirección variable, y para ello es por lo que lo que se les hace una llamada general del tipo 4. De esta manera los circuitos recargan la parte programable externa y, de ser el caso, modifican el valor que tomaron a la puesta en alimentación o en una llamada general previa.

- Valor **00000110**:
  Una llamada general de tipo 6 tiene como efecto el reinicio de los circuitos a su configuración por defecto (la de la puesta en alimentación).

- Resto de valores **XXXXXXX0**:
  Todavía no han sido definidos los restantes valores que poseen su bit menos significativo a cero; cualquier circuito I2C actual debe ignorar estos tipos de llamadas generales.

- Valores **XXXXXXX1**:
  Una llamada general en la que el indicador de tipo tiene a 1 el bit menos significativo se denomina *llamada general hardware*. Este tipo de llamada se ha pensado para admitir que en una topología I2C pueda existir un maestro emisor peculiar que se caracterice por su capacidad de emitir al bus información sin dirigirla a ningún receptor en concreto. Es decir, que la información *se hace pública*, en general, y en lugar de indicar **a quién se envía** se indica **quién la envía**, para que un hipotético receptor interesado en ese origen pueda tomarla. Este tipo de circuitos y las topologías resultantes son infrecuentes, pero resulta interesante que sean admisibles.
  Los bits XXXXXXX del segundo carácter de esta llamada general indican la dirección del maestro emisor *hardware*. A este carácter le seguirán una serie de caracteres adicionales (que habrá que reconocer) que son los datos emitidos al público por este tipo de circuito peculiar. Naturalmente, se ha de terminar con una condición de parada.
  Algunos de estos maestros emisores peculiares también pueden actuar como esclavos receptores. En estos casos, a la puesta en alimentación funcionan como esclavos receptores de manera que otro maestro pueda comunicarse con ellos para indicarles a qué dirección concreta deben dirigirse cuando sea el caso; para ello, ese otro maestro lo direccionará en modo escritura y le enviará un segundo carácter que será la dirección concreta a la que debe dirigirse en su momento. Si esto se ha hecho así, entonces estos circuitos peculiares (*maestros hardware*) cada vez que necesiten enviar algo al bus no lo harán mediante una llamada general de tipo impar sino de manera normal, direccionando a quien se les ha dicho en concreto.

5.2.9. El carácter de inicio

El valor **00000001** (dirección 0, en modo lectura) es un carácter especial denominado *de inicio* que se utiliza al comenzarse una transacción en una topología I2C en la que se sabe que existen esclavos cuya interfaz I2C está implementada fundamentalmente por programa. Efectivamente, esto es muy frecuente cuando alguno de los dispositivos es un microcontrolador que no dispone de controlador de bus I2C; en este caso se pueden utilizar líneas de E/S de propósito general para implantar las líneas del bus I2C y gestionar por programa todo el protocolo del bus. En estas circunstancias la carga computacional de gestión del protocolo puede ser notable, con lo que disminuye el tiempo que se puede dedicar a otras tareas ajenas al control del bus I2C. Si se observa bien, un esclavo de este tipo ha de estar sondeando constantemente por programa el estado de las líneas SDA y SCL simplemente para poder detectar una condición de inicio y, en caso afirmativo, ver si se dirigen a él. Si el maestro que inicia una transacción lo hace a una velocidad elevada esto exige que el esclavo al que nos referimos tenga que estar monitorizando casi permanentemente el bus para que no se le pase la condición de inicio, cosa que empobrece el rendimiento de otras tareas que haya que realizar.
Pues bien, para permitir que este tipo de esclavos que implementan el protocolo I2C por programa puedan conseguir un rendimiento superior en otras tareas es por lo que la norma I2C

contempla un modo especial de inicio de una transacción mediante el primer envío de un *carácter de inicio*. Si se observa, este carácter de inicio contiene siete ceros seguidos, que si se añaden al estado 0 en que quedó la línea SDA al producirse la condición de inicio, entonces se tiene que en la línea SDA se produce un estado al nivel 0 durante un tiempo entre siete y ocho veces superior al tiempo de bit de la transmisión. Por este motivo, un esclavo que monitorice por programa la condición de inicio podrá hacerlo de manera mucho más espaciada en el tiempo, y cuando detecte que en la línea SDA existe un 0 lógico entonces conmutará a un modo de gestión muchos más rápida para poder estar pendiente con detalle de la transacción que se acaba a iniciar. Por otra parte, el maestro que ha iniciado la transacción con el carácter de inicio deberá generar tras él el impulso SCL para el reconocimiento, pero no considerará como erróneo el que no se reciba; es más, ningún esclavo debe reconocer un *carácter de inicio*. A continuación generará una condición de reinicio y comenzará la transacción con el formato normal que ya se conoce.

5.3. EL ARBITRAJE EN LAS TOPOLOGÍAS MULTIMAESTRO

Como se sabe, el bus I2C admite configuraciones en las que pueda existir más de un dispositivo capaz de actuar como maestro. Cualquier bus de comunicación que contemple tal circunstancia ha de proveer algún tipo de mecanismo de arbitraje que solucione el problema de quién gana el gobierno del bus si más de un sistema intenta simultáneamente iniciar una transacción. Los buses más evolucionados definen niveles jerárquicos de manera que el dispositivo de mayor nivel tiene prioridad sobre los de menor; de esta manera es posible hacer configuraciones coherentes con el grado de relevancia de los dispositivos conectados al bus. Pero para conseguir esto es necesario el concurso de señales en número suficiente para determinar el nivel de jerarquía. Dado que el bus I2C tan sólo utiliza dos señales, la de datos y la de reloj, no existe la posibilidad de establecer un mecanismo coherente de jerarquía que pueda utilizarse a la hora del arbitraje. No obstante, esto no quiere decir que no se pueda definir tal método de arbitraje. En concreto, se recurre a una técnica ingeniosa que se apoya, una vez más, en el hecho de que las líneas del bus I2C son a colector o drenador abierto.
El mecanismo de arbitraje consiste en los siguientes principios:

- Todo sistema maestro que pueda formar parte de una topología multimaestro ha de admitir la *sincronización*.

- Todo maestro que pretenda gobernar el bus podrá iniciar una transacción si parte de una condición de bus libre.

- Todo maestro que haya iniciado una transacción y compruebe que el nivel aplicado por él a la línea SDA no se refleja efectivamente en tal línea deberá entonces considerar que algún otro maestro contiende con él por el gobierno del bus, habiendo él perdido y el otro ganado.

- Todo maestro que compruebe que ha perdido el arbitraje, debe inmediatamente aislarse del bus y, de ser el caso, actuar como esclavo receptor por si el ganador del arbitraje se dirigiese a él.

Como puede verse, se saca provecho del hecho de que en las líneas SDA y SCL se tiene el producto lógico de las señales locales SDA y SCL de los excitadores de todos los nodos del bus. Este mecanismo de arbitraje es un tanto peculiar pero efectivo, aunque el resultado del arbitraje (quién gana) se deja a cuestiones puramente azarosas. Efectivamente, **gana el arbitraje el primero que genera un bit de dato a 0 mientras todos los restantes generan un 1**.
Aunque múltiples maestros pueden estar transaccionando a velocidades a priori diferentes, no existe posibilidad de que la emisión de datos se corrompa durante el arbitraje puesto que los maestros han de admitir el mecanismo de *sincronización* por el que los más rápidos adaptan sus relojes a la velocidad del más lento.
En la figura 5.12 puede verse un ejemplo en el que dos maestros pugnan por gobernar el bus. Como puede observarse, además de la sincronización en términos de adaptación del más

rápido al más lento, el arbitraje se pierde en el momento en que un maestro intenta poner un 1 lógico en la línea SDA y se encuentra con que ésta se mantiene a baja.

Si se medita lo que implica este mecanismo de arbitraje puede llegarse acertadamente a la conclusión de que múltiples maestro pueden gobernar simultáneamente el bus, y hasta el final, siempre que se dirijan al mismo esclavo y pretendan realizar absolutamente lo mismo con él. Por ejemplo, si tres maestros pretenden leer los contenidos de las mismas posiciones de memoria de una EEPROM, y coinciden al iniciar la transacción, se tendrá que los tres finalizarán con éxito el proceso de lectura, aunque se emplee para ello el reloj del sistema más lento.

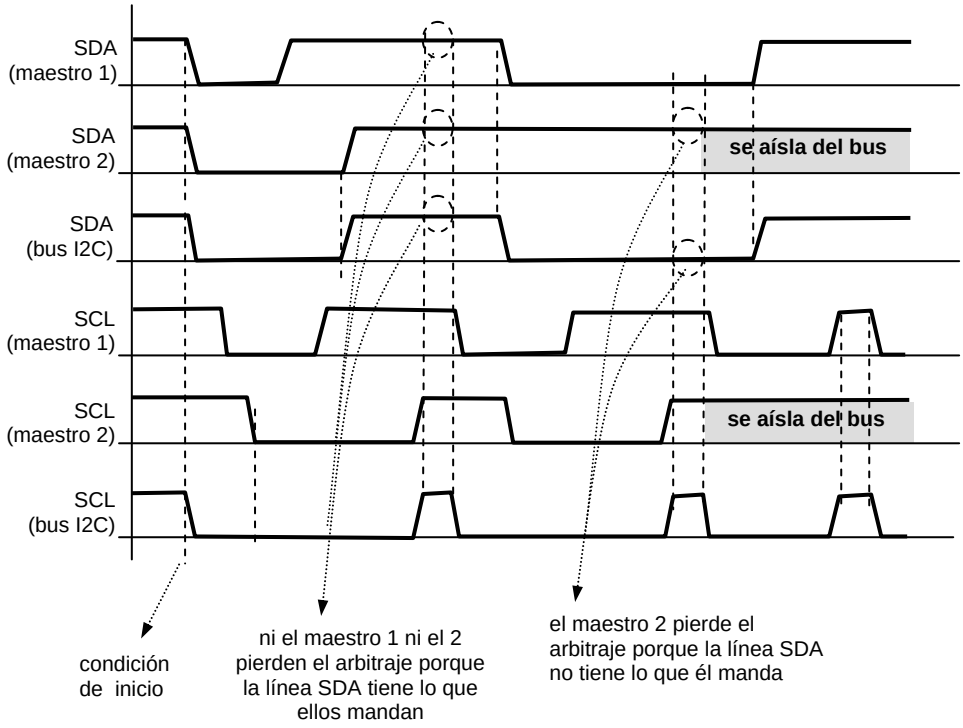


Fig. 5.12: Ejemplo de arbitraje en el bus I2C

En el ejemplo de la figura 5.12 puede verse cómo los maestros han de verificar la coincidencia de su dato con el efectivamente presente en el bus, y esto se hace no cuando cada maestro activa su SCL sino cuando la SCL del bus está activa, puesto que no debe olvidarse que los maestros han de admitir el mecanismo de sincronización. Obsérvese que el mecanismo de sincronización da como resulta una señal de reloj que tiene la frecuencia de la del más lento, y el tiempo a alta del que lo tenga más corto. En el ejemplo, el más lento es el maestro 2, y el tiempo a alta más corto es el del 1, aunque sus dos primeros impulsos tienen una fase activa notablemente más ancha debido al mecanismo de sincronización forzado por el maestro 2, hasta que éste pierde el arbitraje y se aísla entonces del bus.

5.4. EL MODO DE ALTA VELOCIDAD

La norma I2C contempla tres modos de funcionamiento: normal, rápido y de alta velocidad. El primero admite tasas de hasta 100 kilobits por segundo, el segundo –introducido en la revisión 1.0 de 1992– de hasta 400 kbps, y el tercero –introducido en la revisión 2.0 de 1998– de hasta 3'4 megabits por segundo. Existe una compatibilidad descendente, de tal manera que un dispositivo capaz de funcionar en modo de alta velocidad también lo hará en modo rápido, y uno en modo rápido también lo hará en modo normal. Actualmente, todo nuevo dispositivo I2C ha de ser capaz de funcionar al menos en modo rápido; no obstante, los circuitos capaces del

modo de alta velocidad resultan una excelente solución para aquellas aplicaciones de altas prestaciones en las que se demande una elevada tasa de transferencia. Este tipo de circuitos necesita que la electrónica de excitación en los maestros posea unas características especiales con el fin de acelerar las transiciones de baja a alta en la línea de reloj, puesto que de actuar tan sólo la resistencia de elevación sería imposible conseguir las elevadas tasas transferencia de hasta 3'4 megabits. No se olvide que la resistencia de elevación actuará en las transiciones 0→1 como vía de carga de la capacidad concentrada en el bus (hasta 400 pF según la norma), y por ello en la línea SCL se tendrá la típica forma exponencial de carga de un condensador, en el que la constante de tiempo $\tau=R_eC$ limitará la máxima velocidad alcanzable. Los aspectos topológicos así como de tipo eléctrico y temporal se comentarán más adelante, y nos centraremos ahora en comentar los aspectos funcionales y de protocolo que caracterizan al modo de alta velocidad. Por todo lo comentado, los circuitos I2C de alta velocidad denominan a sus señales SDAH y SCLH en lugar de SDA y SCL; no obstante, pueden existir circuitos que ofrezcan ambos tipos de señales, en cuyo caso las normales pueden utilizarse para otros fines si no se utilizan. Esta dualidad de señales resulta útil en topologías con velocidades mixtas.

Por lo que respecta a las características de protocolo, en primer lugar hay que decir que este modo no admite ni el arbitraje ni la sincronización bit a bit puesto que tal cuestión iría en contra del objetivo perseguido de conseguir la máxima velocidad posible de transferencia. Esto es, todo dispositivo maestro capaz de la alta velocidad deberá realizar transacciones sólo con circuitos también capaces de la alta velocidad. Por supuesto, si un maestro es capaz de la alta velocidad pero funciona en un modo inferior (rápido o normal) entonces podrá dirigirse a cualquier circuito capaz de tales modos inferiores. El arbitraje se realiza en un modo que no es de alta velocidad, como se analizará más adelante.
En este momento puede resultar interesante comentar un aspecto de las topologías mixtas en las que existen circuitos de alta velocidad (*Hs* de aquí en adelante) junto a otros rápidos o normales. Para evitar cargar excesivamente el bus (a mayor número de circuitos mayor capacidad equivalente existirá en él) los circuitos normales y rápidos se pueden separar de los de alta velocidad mediante sendos *puentes* en SDA y SCL, que no son más que unos circuitos especiales diseñados para tal caso.

5.4.1. El formato de una transferencia de alta velocidad

Todo circuito *Hs* tiene la obligación de entrar en modo *rápido* a la puesta en alimentación. Un maestro *Hs* que quiera iniciar una transacción podrá hacerlo tal y como ya se sabe si va a emplear una velocidad rápida o normal, pero si desea emplear la alta velocidad entonces antes deberá realizar una *petición de alta velocidad*, que consiste en lo que sigue:

1º) Generar una condición de inicio.

2º) Emitir su código de maestro *Hs*, que es 00001XXX. Esto implica que no se admiten topologías con más de ocho maestros *Hs*. Realmente deberían ser siete, puesto que la norma recomienda reservar el código 00001000 para que pueda ser utilizado por equipos de análisis y diagnóstico de buses I2C.

3º) Al código de maestro *Hs* nadie debe darle su reconocimiento.

En el transcurso de estas tres fases, que se realizan o bien en modo normal o bien en modo rápido, se admite tanto el arbitraje como la sincronización. Por otra parte el código 00001XXX sirve de indicador de que a continuación se va a comenzar una transacción a alta velocidad.
El código concreto de un maestro *Hs* (desde 00001000 hasta 00001111) es de libre elección por parte del diseñador de un sistema, y ningún circuito que pueda comportarse como tal lo lleva pre-asignado por el comité I2C.
El maestro *Hs* que, tras el no-reconocimiento, logre el gobierno del bus pondrá a alta su señal SCLH y a partir de este mismo instante t_H se conmutará al modo *Hs*. En este momento el maestro *Hs* generará una condición de reinicio a la que le seguirá la dirección (7 o 10 bits) del esclavo con el que se desea realizar la transacción, con una mecánica de funcionamiento lógico enteramente similar a la ya conocida.

Dado que en el modo *Hs* se admite tan sólo la sincronización carácter a carácter, para hacerla posible el maestro ha de desactivar su excitador SCLH (es una fuente de corriente) en los instantes tras cada condición de reinicio y tras cada tiempo de reconocimiento (se dé o no se dé). De esta manera sólo queda operativa la resistencia de elevación y así es posible que un posible esclavo receptor pise efectivamente la señal SCLH sin temor a que se destruya su propio excitador SCLH local. El maestro *Hs* reactivará su excitador especial SCLH en el momento en que detecte que nadie le está pisando su SCLH local.
Todo lo comentado puede observarse en la figura que sigue.

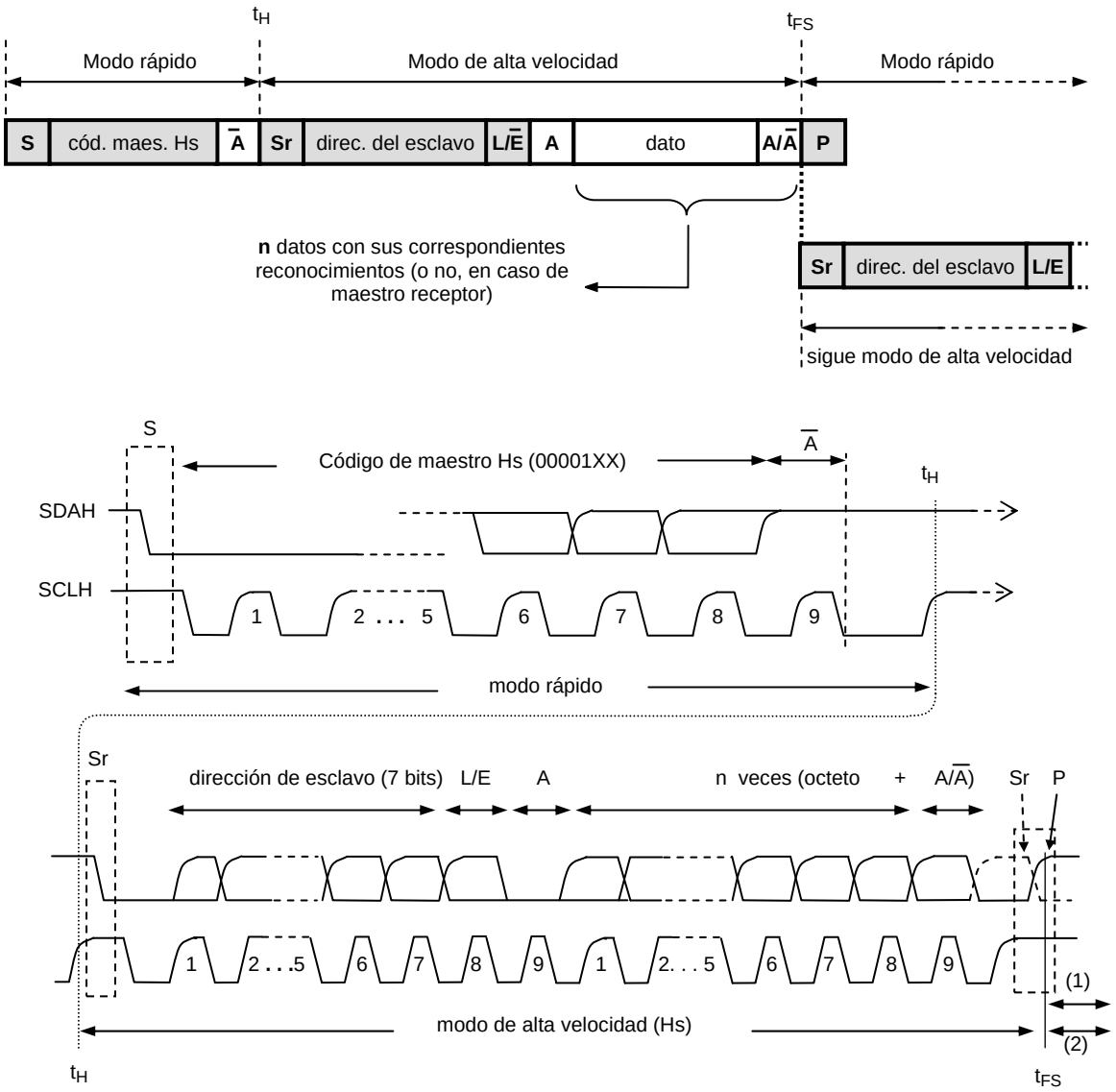


Fig. 5.13: Formatos de una transacción en modo de alta velocidad (Hs)

El cronograma que se muestra en la figura 5.13 se ha dividido en dos mitades por falta de espacio; no obstante, el lector debe considerarlo como uno solo en el que las líneas del extremo derecho en la primera mitad (la superior) continúan por el extremo izquierdo de la

segunda mitad (la inferior); el instante de nexo entre las dos mitades es el indicado t_H, que es en el que el maestro entra en el modo de alta velocidad.

5.5. Las especificaciones eléctricas y temporales

En la figura 5.14 puede observarse el aspecto de la lógica electrónica de excitación y de recepción de un dispositivo I2C de tecnología MOS, válida para los modos normal y rápido. Los transistores de excitación de las líneas SDA y SCL del bus I2C son a drenador abierto, y por ello las líneas SDA y SCL del bus deben tener conectadas a V_{DD} sendas resistencias de elevación que marquen el estado lógico 1 cuando el respectivo transistor no conduzca. También puede verse cómo el estado de ambas líneas del bus puede ser sondeado mediante el correspondiente búfer de entrada (disparador de Schmitt). Esto, que puede parecer lógico para la línea de datos serie, SDA, si se desea poder actuar tanto como emisor como receptor, también es necesario para la señal de reloj SCL si se desea tener la capacidad de sincronización.
Como puede verse, el nivel que se desea aplicar a una línea del bus se aplica invertido a la puerta del transistor, de manera que si el dato es un 0, al invertirse a 1 hace que el transistor conduzca; como la resistencia drenador-surtidor es muy baja en estas condiciones, entonces el nivel que se tiene a la salida es prácticamente el del común de señal, es decir, un cero lógico. Y si el dato es un 1 lógico, al invertirse a 0 la tensión asociada hace que el transistor no conduzca, ofreciendo una elevadísima resistencia drenador-surtidor. Esta enorme impedancia de salida trae como efecto el práctico aislamiento eléctrico del excitador con respecto a la línea del bus. En este caso, el nivel existente en el bus es un 1 lógico al actuar la resistencia externa como una de elevación.

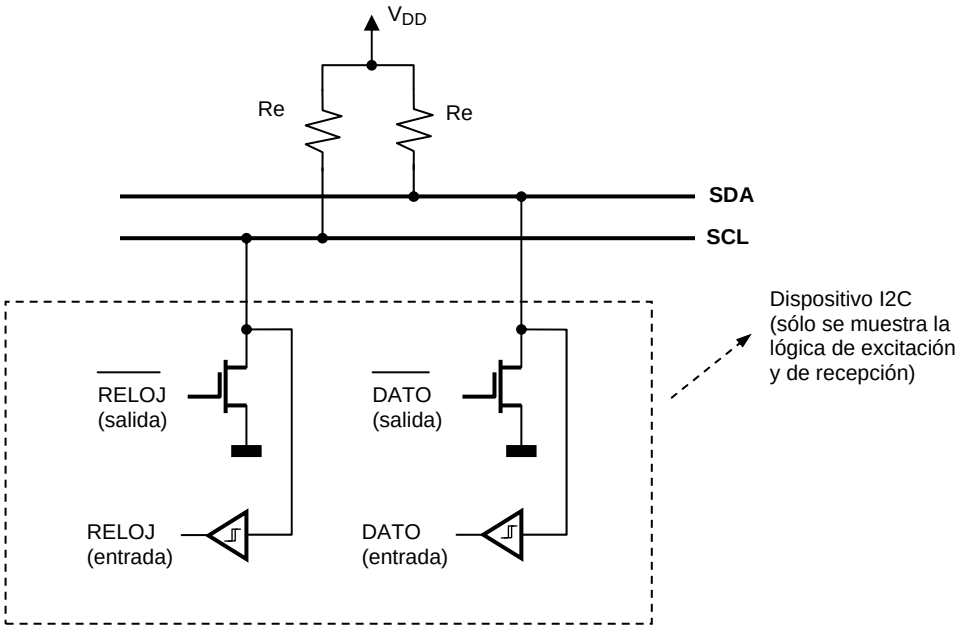


Fig. 5.14: Etapa de excitación y recepción en un circuito I2C normal o rápido

Cuando el bus está libre los excitadores están inactivos, y por tanto son las resistencias de elevación las que marcan los niveles a alta en las líneas.
Para los circuitos en modo rápido las resistencias de elevación pueden ser tales si las cargas conectadas al bus no exceden los 200 pF, pero si están entre este valor y los 400 pF entonces deben emplearse fuentes de corriente (3 mA máximo) o una red resistiva conmutada. En el modo de alta velocidad el umbral está en los 100 pF
Por otra parte, entre cada línea del bus y su terminal de cada circuito I2C se pueden conectar una resistencia en serie si se desea proteger a los circuitos ante posibles picos de tensión que puedan producirse en las líneas. El valor de esta resistencia serie R_S no debe exceder los 100 ohmios. En los circuitos que admiten la alta velocidad estas resistencias en serie además

sirven para minimizar fenómenos como las oscilaciones espurias, reflexiones de señal y demás interferencias.
La electrónica de excitación incluye una circuitería cuya misión es controlar la velocidad de cambio en las líneas SDA y SCL, manteniéndola dentro de unos límites concretos. Esta limitación de la tasa de cambio (*slew rate* en inglés) persigue minimizar la generación de interferencias electromagnéticas hacia los circuitos en las proximidades del bus; no se olvide que los cambios rápidos en una señal digital contienen un espectro de altas frecuencias capaces de inducir IEM.
Cuando se comentó el modo de alta velocidad se dijo que estos circuitos incluyen un excitador especial –fuente de corriente– que tiene como misión conseguir aumentar la velocidad de cambio de la línea SCLH. En este modo las resistencias de elevación llegan a constituir un problema debido a la constante de tiempo asociada, que es el producto de la capacidad acumulada en una línea por el valor de la resistencia de elevación. De hecho el problema se produce esencialmente en los cambios de baja a alta; y esto es debido a que, como ya se ha dicho, cuando un excitador genera un 1 lógico realmente no lo hace puesto que no conduce y ofrece una elevada impedancia de salida. Por tanto el nivel en la línea queda determinado por la forma en que se carga una capacidad. En los cambios de alta a baja se produce una rápida descarga de esta capacidad concentrada equivalente, a través del transistor del excitador, que en este caso posee una resistencia colector-emisor o drenador-surtidor muy reducida. Así, en los excitadores especiales SCLH se tiene una fuente de corriente que en las transiciones de baja a alta se encarga de inyectar una elevada corriente a la línea SCLH forzando una rápida carga de la capacidad equivalente en la línea. Esto puede observarse en la figura que sigue.

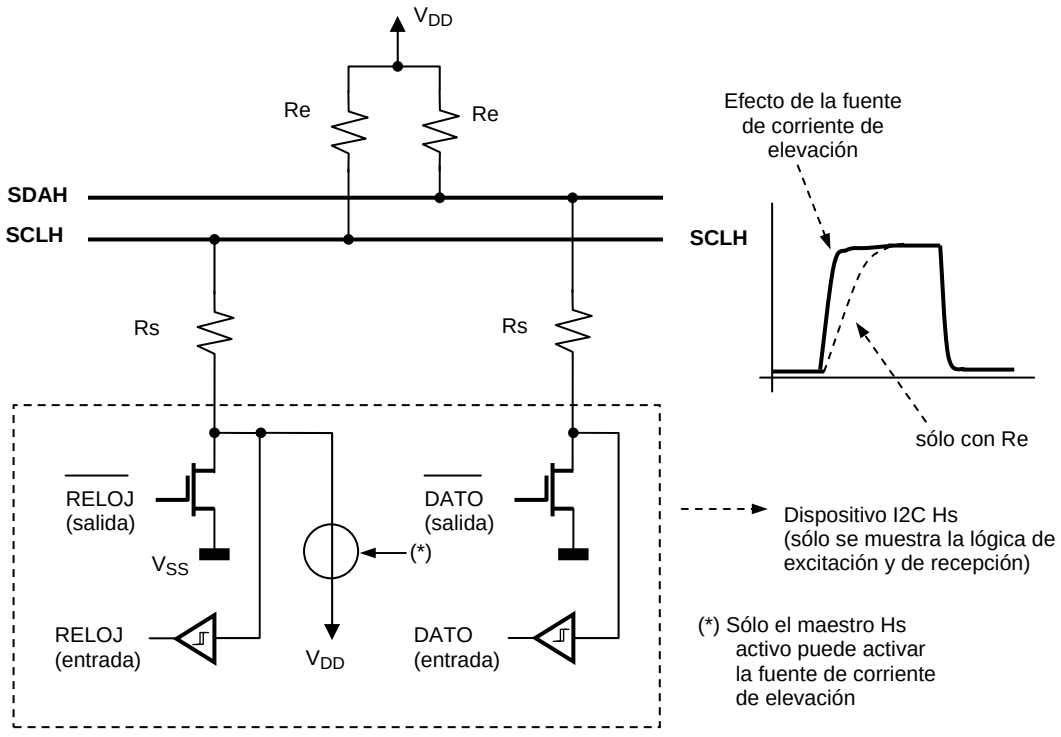


Fig. 5.15: Etapa de excitación y recepción en un circuito maestro I2C Hs

Topologías con velocidades mixtas

Como ya se ha apuntado anteriormente, es posible trabajar con topologías mixtas en las que existan circuitos normales, rápidos y de alta velocidad. En este caso es posible conseguir distintas velocidades de transferencia haciendo uso de un puente de interconexión entre la sección del bus al que se conectan los circuitos Hs y la que tiene los rápidos y normales.

Existen circuitos maestros Hs que ya integran este puente; en la norma I2C versión 2.1 pueden obtenerse los detalles de la filosofía operativa de estos puentes. Aquí tan sólo indicaremos en la tabla siguiente las posibles velocidades que se pueden conseguir

Tabla 5.2: Velocidades alcanzables en topologías mixtas

TRANSFERENCIA ENTRE. . .	CONFIGURACIÓN DEL BUS I2C			
	Hs + RÁPIDO + NORMAL	**Hs + RÁPIDO**	**Hs + NORMAL**	**RÁPIDO + NORMAL**
Hs ↔ Hs	0 ~ 3'4 Mbits/s	0 ~ 3'4 Mbits/s	0 ~ 3'4 Mbits/s	---
Hs ↔ Rápido	0 ~ 100 kbits/s	0 ~ 400 kbits/s	---	---
Hs ↔ Normal	0 ~ 100 kbits/s	---	0 ~ 100 kbits/s	--
Rápido ↔ Rápido	0 ~ 100 kbits/s	---	---	0 ~ 100 kbits/s
Rápido ↔ Normal	0 ~ 100 kbits/s	0 ~ 100 kbits/s	---	0 ~ 100 kbits/s
Normal ↔ Normal	0 ~ 100 kbits/s	---	0 ~ 100 kbits/s	0 ~ 100 kbits/s

Los maestros Hs en topologías mixtas utilizarán las señales SDAH, SCLH y SDA, SCL, manejadas por el puente, cada una de las cuales se conectarán a las secciones oportunas del bus I2C. Las SDAH y SCLH siempre se utilizarán, independientemente del modo empleado de la transferencia, mientras que las SDA y SCL tan sólo serán utilizadas en los modos normal y rápido. Si un usuario utiliza un maestro Hs esta cuestión es totalmente transparente puesto que es gestionada por la capa física de la interfaz que integre tal dispositivo.

5.5.1. Los niveles de tensión y de corriente

Dado que la norma admite la conexión al bus de circuitos de muy diferentes tecnologías, se especifican de varias maneras los niveles de tensión de las señales. A continuación se indica de manera resumida esta cuestión.

- Para aquellos circuitos que se alimentan a 5 voltios (sean TTL, CMOS, etcétera), se tiene lo siguiente:
  - Tensión mínima de entrada a alta: V_{IHmin}=3 V
  - Tensión máxima de entrada a baja: V_{ILmax}=1'5 V

- Para aquellos circuitos que admiten amplios márgenes de alimentación se tiene que:
  - Tensión mínima de entrada a alta: V_{IHmin}=70% de V_{DD}
  - Tensión máxima de entrada a baja: V_{ILmax}=30% de V_{DD}

- Por otra parte, ha de cumplirse que, independientemente de la tensión de alimentación V_{DD}, la tensión de salida a baja V_{OL} no ha de superar los 0'4 voltios para una corriente de salida de 3 mA.

- La corriente de entrada en SDA o SCL de un dispositivo I2C, para un nivel alto, no excederá los -10μA.

En la tabla que sigue se indican los principales parámetros eléctricos en los modos normal, rápido y de alta velocidad, en lo que atañe a las etapas excitadoras y receptoras en un circuito I2C.

Tabla 5.3: características de las etapas de E/S de SDA y SCL en un dispositivo I2C normal o rápido

PARÁMETRO	SÍM-BOLO	MODO NORMAL		MODO RÁPIDO		UNI-DAD
		mínimo	máximo	mínimo	máximo	
Tensión de entrada para nivel BAJO: Niveles de entrada fijos Niveles relativos a V_{DD}	V_{IL}	 -0'5 -0'5	 1'5 30% V_{DD}	 no aplicable -0'5	 no aplicable 30% V_{DD} (1)	V
Tensión de entrada para nivel ALTO: Niveles de entrada fijos Niveles relativos a V_{DD}	V_{IH}	 3'0 70% V_{DD}	 V_{DDmax}+0'5 V_{DDmax}+0'5	 n. a. 70% V_{DD} (1)	 n. a. V_{DDmax}+0'5	V
Histéresis de los disparadores de Schmitt en las entradas: V_{DD} > 2 V V_{DD} < 2 V	V_{hys}	 n. a. n. a.	 n. a. n. a.	 5% V_{DD} 10% V_{DD}	 ---- ----	V
Tensión de salida a nivel bajo (colector o drenador abierto) sumiendo 3 mA: V_{DD} > 2 V V_{DD} < 2 V	 V_{OL1} V_{OL3}	 0 n. a.	 0'4 n. a.	 0 n. a.	 0'4 20% V_{DD}	V
Tiempo de bajada, desde V_{IHmin} hasta V_{ILmax}, con una capacidad en el bus desde 10 pF hasta 400 pF	t_{of}	----	250 (3)	20+0'1C_b (2)	250 (3)	ns
Anchura de los picos espurios que deben ser suprimidos por el filtro de las entradas	t_{SP}	n. a.	n. a.	0	50	ns
Corriente de entrada en cada patilla de E/S con una tensión de entrada entre el 10 y el 90% de V_{DDmax}	I_i	-10	10	-10 (4)	10 (4)	μA
Capacidad de entrada de patilla de E/S	C_i	----	10	----	10	pF

(1) Aquellos dispositivos con una alimentación no estándar, que no se ajustan a los niveles pensados para un sistema I2C, deben relacionar sus niveles de entrada a la tensión V_{DD} a la que se conectan las resistencias de elevación Re.

(2) C_b = capacidad de una línea del bus, en picofaradios (es la capacidad concentrada del cable y de las patillas de E/S)

(3) Este valor máximo de t_{of} para las salidas de los excitadores es menor que el indicado t_f en la tabla 5.5 para las líneas SDA ySCL (300 ns). El motivo es permitir la conexión como protección de resistencias en serie, Rs, entre la salida de cada excitador y las líneas SDA y SCL, sin exceder tal valor t_f.

(4) En los dispositivos que admitan el modo rápido las patillas de E/S no deben bloquear las líneas SDA y SCL si V_{DD} se apaga o suprime.

Tabla 5.4: características de las etapas de E/S de SDA(H) y SCL(H) en un dispositivo I2C de alta velocidad

PARÁMETRO	SÍM-BOLO	MODO Hs		UNI-DAD
		mínimo	máximo	
Tensión de entrada para nivel BAJO:	V_{IL}	-0'5	30% V_{DD}(1)	V
Tensión de entrada para nivel ALTO:	V_{IH}	70% V_{DD}(1)	V_{DD}+0'5(2)	V
Histéresis de los disparadores de Schmitt en las entradas:	V_{hys}	10% V_{DD}	---	V
Tensión de salida a nivel BAJO (colector o drenador abierto) sumiendo 3 mA: V_{DD} > 2 V V_{DD} < 2 V	V_{OL}	 0 0	 0'4 20% V_{DD}	V
Resistencia, en estado activo, de la puerta de transferencia, para ambos sentidos de la corriente a nivel VOL entre SDA y SDAH o SCL y SCLH, a 3 mA	R_{onL}	----	50	Ω
Resistencia, en estado activo, de la puerta de transferencia, entre SDA y SDAH (o SCL y SCLH) si ambos están al nivel V_{DD}	R_{onH}(2)	50	---	KΩ
Corriente de elevación, de la fuente de corriente de SCLH. Es válido para niveles entre el 30 y el 70% de V_{DD}	I_{CS}	3	12	mA
Tiempo de subida, en la salida (con la fuente de corriente activa), y Tiempo de bajada, para SCLH con una capacidad de carga de entre 10 y 100 pF	t_{rCL} t_{fCL}	10	40	ns
Tiempo de subida, en la salida (con la fuente de corriente activa), y Tiempo de bajada, para SCLH con una fuente de corriente externa de 3 mA y capacidad de carga de hasta 400 pF	t_{rCL}(3) t_{fCL}(3)	20	80	ns
Tiempo de bajada, en la salida SDAH, con una capacidad de la carga en el bus de 10 pF hasta 100 pF	t_{fDA}	10	80	ns
Tiempo de bajada, en la salida SDAH, con una capacidad de la carga en el bus de 400 pF	t_{fDA}(3)	20	160	ns
Anchura de los picos espurios en SDAH y SCLH que deben ser suprimidos por el filtro de las entradas	t_{SP}	0	10	ns
Corriente de entrada en cada patilla de E/S con una tensión de entrada entre el 10 y el 90% de V_{DD}	I_i	----	10	μA
Capacidad de entrada de patilla de E/S	C_i	----	10	pF

(1) Aquellos dispositivos con una alimentación no estándar, que no se ajustan a los niveles pensados para un sistema I2C, deben relacionar sus niveles de entrada a la tensión V_{DD} a la que se conectan las resistencias de elevación Re.

(2) Los dispositivos que ofrezcan la función de ajuste de niveles deben admitir una tensión máxima de entrada de 5'5 V en SDA y SCL.

(3) El valor de los tiempos de subida y de bajada debe interpolarse linealmente para capacidades de carga entre los 100 y 400 pF.

(4) Las etapas SDAH y SCLH de un dispositivo esclavo Hs deben poseer salidas flotantes si su alimentación se ha suprimido o apagado. Este requisito no es obligatorio para las etapas de E/S SDAH y SCLH en un dispositivo maestro Hs; ello es debido al circuito fuente de corriente de salida, que normalmente posee un diodo recortador conectado a V_{DD}. Esto significa que la fuente de alimentación de un dispositivo maestro en modo de alta velocidad, Hs, no puede apagarse sin afectar a las líneas SDAH y SCLH.

5.5.2. La especificación de la temporización

A continuación se indicarán los parámetros temporales definidos por la norma I2C. En la tabla 5.6 para los circuitos en modo normal o rápido, y en la tabla 5.7 para los de alta velocidad. En los cronogramas mostrados en las figuras 5.16 (modos *normal* y *rápido*) y 5.17 (modo de *alta velocidad*) se puede observar el significado de cada uno de estos parámetros.

Modos rápido/normal

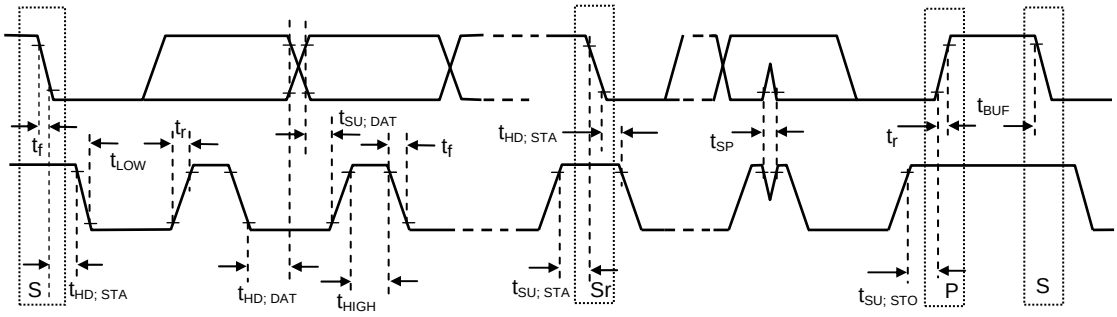


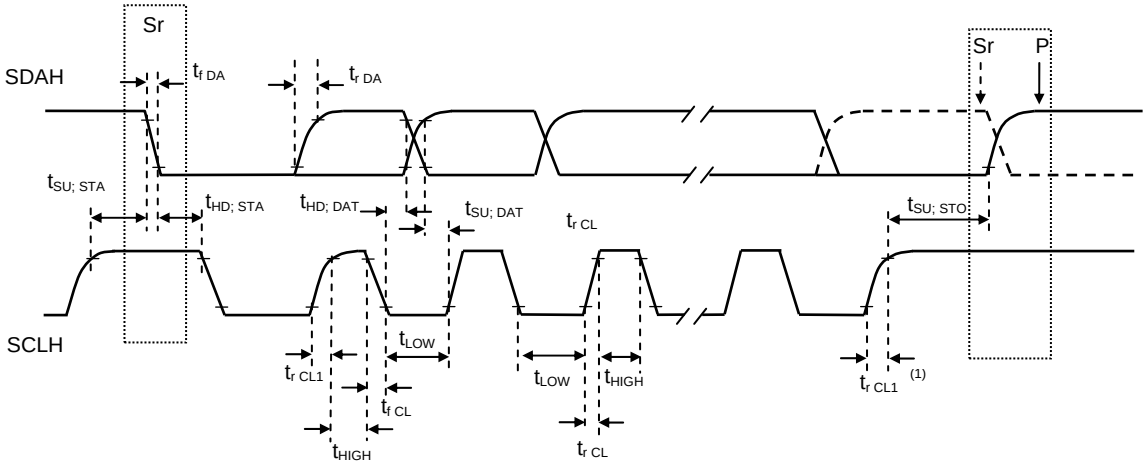
Fig. 5.16: Parámetros de temporización en el modo normal y rápido del bus I2C

Tabla 5.5: Parámetros de temporización en los modos normal y rápido(1)

PARÁMETRO	SÍMBOLO	MODO NORMAL		MODO RÁPIDO		UNI-DAD
		mínimo	máximo	mínimo	máximo	
Frecuencia del reloj SCL	f_{SCL}	0	100	0	400	khz
Tiempo de retención en la condición de (RE-) INICIO. Tras este instante se genera el primer impulso de reloj	$t_{HD;\ STA}$	4'0	----	0'6	----	µs
Periodo a BAJA de la señal de reloj SCL	t_{LOW}	4'7	----	1'3	---	µs
Periodo a ALTA de la señal de reloj SCL	t_{HIGH}	4'0	----	0'6	----	µs
Tiempo de asentamiento para una condición de reinicio	$t_{SU;STA}$	4'7	----	0'6	----	µs
Tiempo de retención de un dato: - Para maestros compatibles CBUS - Para dispositivos I2C	$t_{HD;DAT}$	 5'0 0(2)	 ---- 3'45(3)	 ---- 0(2)	 --- 0'9(3)	µs
Tiempo de asentamiento de un dato	$t_{SU;DAT}$	250	----	100(4)	---	ns
Tiempo de subida de SDA y de SCL	t_r	----	1000	$20+0'1C_b$(5)	300	ns
Tiempo de bajada de SDA y de SCL	t_f	----	300	$20+0'1C_b$(5)	300	ns
Tiempo de asentamiento para una condición de PARADA	$t_{SU;STO}$	4'0	----	0'6	----	µs
Tiempo de bus libre entre una condición de PARADA y una de INICIO	t_{BUF}	4'7	-----	1'3	----	µs
Capacidad de la carga en cada la línea	C_b	----	400	----	400	pF
Margen de ruido a nivel BAJO para cada dispositivo conectado el bus (incluyendo histéresis)	V_{nL}	10% V_{DD}	----	10% V_{DD}	----	V
Margen de ruido a nivel ALTO para cada dispositivo conectado el bus (incluyendo histéresis)	V_{nH}	20% V_{DD}	----	20% V_{DD}	----	V

(1) Todos los valores están referidos a los niveles V_{IHmin} y V_{ILmax}. (véase la tabla 5.3)
(2) Un dispositivo debe procurar un tiempo de retención de al menos 300 ns para la señal SDA (con respecto a la señal SCL en su valor V_{IHmin}) para así cubrir la región no definida en el transcurso del cambio a baja de SCL.
(3) El valor máximo de $T_{HD;DAT}$ sólo debe satisfacerse si el dispositivo no alarga la fase a BAJA (t_{LOW}) de la señal SCL.
(4) Un dispositivo en modo rápido I2C puede utilizarse en un sistema I2C en modo normal, pero ha de cumplirse el requisito de que $t_{SU;DAT} \geq 250$ ns. Automáticamente este será el caso si el dispositivo no alarga la fase a BAJA de la señal SCL. Si el dispositivo sí que lo hace entonces deberá lanzar el siguiente bit por la línea SDA un tiempo $t_{r(max)}+t_{SU;DAT}=1000+250=1250$ ns antes de que la línea SCL sea relajada (valores acordes a la especificación del modo normal).
(5) Cb= capacidad total de una línea del bus, en picofaradios. En topologías mixtas se admiten tiempos de bajada más rápidos, conformes a la tabla 5.6.

Modo de alta velocidad (Hs)



Con este grafismo se quiere significar la acción de la fuente de corriente del maestro Hs

Con este grafismo se quiere significar la acción de la resistencia de elevación Re

(1) Primer flanco de subida de la señal SCLH tras cada bit de reconocimiento

Fig. 5.17: Parámetros de temporización en el modo de alta velocidad (Hs) del bus I2C

Tabla 5.6: Parámetros de temporización en el modo de alta velocidad (1)

PARÁMETRO	SÍMBOLO	C_b = 100 pF máximo		C_b = 400 pF (2)		UNI-DAD
		mínimo	máximo	mínimo	máximo	
Frecuencia del reloj SCLH	f_{SCLH}	0	3'4	0	1'7	Mhz
Tiempo de asentamiento en la condición de (RE-) INICIO	$t_{SU;STA}$	160	----	160	----	ns
Tiempo de retención en la condición de (RE-) INICIO.	$t_{HD;STA}$	160	----	160	----	ns
Periodo a BAJA de la señal de reloj SCLH	t_{LOW}	160	----	320	---	ns
Periodo a ALTA de la señal de reloj SCLH	t_{HIGH}	60	----	120	----	ns
Tiempo de asentamiento de un dato	$t_{SU;DAT}$	10	----	10	---	ns
Tiempo de retención de un dato:	$t_{HD;DAT}$	0(3)	70	0(3)	150	ns
Tiempo de subida de SCLH	t_{rCL}	10	40	20	80	ns
Tiempo de subida de SCLH tras una condición de REINICIO y tras un bit de reconocimiento	t_{rCL1}	10	80	20	160	ns
Tiempo de bajada de SCLH	t_{fCL}	10	40	20	80	ns
Tiempo de subida de SDAH	t_{rDA}	10	80	20	160	ns
Tiempo de bajada de SDAH	t_{fDA}	10	80	20	160	ns
Tiempo de asentamiento para una condición de PARADA	$t_{SU;STO}$	160	-----	160	----	ns
Capacidad de la carga en SDAH o SCLH	C_b (2)	----	100	----	400	pF
Capacidad de la carga en SDAH+SDA o SCLH+SCL	C_b	----	400	----	400	pF
Margen de ruido a nivel BAJO para cada dispositivo conectado el bus (incluyendo histéresis)	V_{nL}	10% V_{DD}	----	10% V_{DD}	----	V
Margen de ruido a nivel ALTO para cada dispositivo conectado el bus (incluyendo histéresis)	V_{nH}	20% V_{DD}	----	20% V_{DD}	----	V

(1) Todos los valores están referidos a los niveles V_{IHmin} y V_{ILmax}. (véase la tabla 5.4)

(2) Los valores de temporización han de interpolarse linealmente para cargas en la línea, C_b, comprendidas entre los 100 y los 400 pF.
(3) Un dispositivo de proporcionar internamente y tiempo de retención de dato para cubrir la zona indefinida del cambio a baja de SCLH entre los instantes V_{IH} y V_{IL}. Para minimizar este problema un dispositivo ha de procurar tener este umbral lo más pequeño posible.

Ejemplo de implementación de la capa lógica de la interfaz I2C gestionada por programa

Como ejemplo de aplicación, se partirá de un microcontrolador 89C52. Para implementar un bus I2C se utilizarán las líneas oportunas de un puerto de E/S para generar las señales SDA y SCL, y todo lo relativo al protocolo I2C se realizará enteramente por programa.
Por lo que respecta a la capa física de la interfaz, es necesario que al menos sea capaz de lo siguiente:

- Ofrecer dos líneas con salida a drenador (o colector) abierto, para SDA y SCL.

- Ofrecer dos entradas digitales, una para sondear SDA y otra para sondear SCL.

Si no fuese éste el caso, entonces sería necesario utilizar la circuitería externa precisa. Por ejemplo, unos búferes a drenador abierto.
En el caso del microcontrolador 89C52, de la familia MCS51, se tiene que posee puertos de E/S de tipo cuasi bidireccional, esto es, permiten utilizar una línea de E/S tanto como entrada como salida sin tener que programar el sentido (la única precaución es que si se utiliza como entrada entonces el cerrojo de salida asociado ha de estar a alta para evitar que si la entrada es una tensión alta pueda dañar el excitador de la etapa de salida). La estructura básica de una línea de un puerto de E/S en un controlador MCS51 es similar a la de un circuito I2C salvo que también integra la resistencia de elevación. Por tanto, un cero de salida lo marca el transistor de salida pero un uno de salida es marcado por la resistencia interna de elevación. Esta característica resulta especialmente interesante para implementar con ellos una interfaz I2C por programa. En nuestro ejemplo se utilizarán las líneas P1.0 como SDA y P1.1 como SCL. Por otra parte, se utilizarán sendas resistencias externas de elevación, de valor 18 KΩ, debido a que las internas resultan excesivamente pequeñas como para satisfacer las características eléctricas y temporales indicadas por la norma.
Debido a las características de esta capa física I2C, es muy importante que la capa lógica de gestión tenga las siguientes precauciones en lo que atañe al control de la capa física del bus:

- **Si se actúa como maestro emisor**: no es necesario ninguna precaución.

- **Si se actúa como maestro receptor**: La línea de E/S que implemente la señal SDA deberá tener a 1 su cerrojo de salida asociado. Esto equivale a aislar del bus la señal SDA del excitador local, y poder utilizarla como entrada.

- **Si se actúa como esclavo emisor**: La línea de E/S que implemente la señal SCL debe tener a 1 el registro asociado de salida. De esta manera el excitador local se aísla del bus y sólo es operativo el búfer receptor. En cualquier caso, si se desea emplear el mecanismo de *sincronización* entonces se podrá a 0 el cerrojo asociado de salida, debiendo volverse a poner a1 (aislamiento del bus) al finalizarlo.

- **Si se actúa como esclavo receptor**. Exactamente igual que en el caso anterior, pero además en la línea de E/S que implemente la señal SDA el cerrojo de salida asociado debe estar a 1 para que el excitador SDA esté aislado del bus.

Diseño de la capa lógica

A continuación se mostrarán algunos ejemplos de rutinas básicas que implementas las funciones elementales de gestión de un bus I2C, independientemente de la función concreta que se desee realizar con un circuito: generación o detección de las condiciones de inicio y de parada, serialización de un carácter y recepción de un carácter (en ambos modos, maestro y

esclavo), generación de reconocimiento, espera de reconocimiento, etcétera. Se supondrá que el microcontrolador 89C52 funcionará a 12 Mhz; es decir, el ciclo máquina tendrá una duración de 1 μs.

Rutinas básicas en modo maestro:

```
SDA     EQU     90h     ; dirección de bit de P1.0
SCL     EQU     91h     ; dirección de bit de P1.1

; ============================================================================
; =                    RUTINA QUE GENERA LA CONDICIÓN DE INICIO              =
; ============================================================================

; SDA  ¯¯¯¯¯¯¯¯¯¯¯¯|_____________
;                  |← t →|
; SCL  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|_____
;                  |     |
;                 (1)   (2)

; se parte de la condición de bus libre (SDA=SCL=1)
GENERA_INICIO:  CLR     SDA             ; (1)
                NOP                     ; (t) mantiene la condición de inicio durante 3 ciclos
                NOP
                CLR     SCL             ; se vuelve con SCL ya a baja  (2)
                RET


; ============================================================================
; =                    RUTINA QUE GENERA LA CONDICIÓN DE PARADA              =
; ============================================================================

; SDA  XXXXXX_____________|¯¯¯¯¯¯¯¯¯¯
;                 |← t1 →|← t2 →|
; SCL  ___________|      |¯¯¯¯¯¯¯¯¯¯¯¯¯¯
;                 |      |      |
;                (1)    (2)    (3)

GENERA_PARADA:                          ; condición de entrada: SCL debe estar a baja
                CLR     SDA             ; (1)  nos aseguramos que SDA parte de valor 0
                NOP                     ; (t1)
                SETB    SCL             ; (2) activa el reloj
                NOP                     ; (t2)
                NOP
                SETB    SDA             ; (3) cambio 0 → 1 de la línea SDA, estando a 1 SCL
                NOP
                RET


; ============================================================================
; =              RUTINA MAESTRA QUE LANZA (CON O SIN INICIO) UN CARÁCTER     =
; ============================================================================

; SDA  ¯¯¯|_____XX_________XX_________XX(. . .)XX_________XX¯¯¯¯¯¯¯¯¯¯¯¯
;          |    |← t4 →|
; SCL  ¯¯¯¯|____|      |____|¯¯¯¯¯¯|¯¯¯¯¯¯¯|.)   (.|¯¯¯¯¯¯¯|¯¯¯¯¯¯|¯¯¯¯¯¯|_____
;          |    |      |    |      |       |      |       |      |      |
;        (1) (2)  (3)(4)   (5)  (3)(4)    (5)           (3) (4)    (5)

; condición de entrada: el carácter con la dirección y L/E estará en Acumulador

LANZA_DIRECCION_CON_INICIO:
                CALL    GENERA_INICIO
LANZA_DIRECCION:                ; condición de entrada: el carácter con la dirección y L/E estará en Acumulador
LANZA_CARÁCTER_M:
                MOV     R7,#8           ; hay que lanzar los ocho bits del carácter
LANZA_BIT:      RLC     A               ; rota a la izquierda (se empieza por el más significativo)
                MOV     SDA,C           ; (3) lanza el bit a la línea de datos SDA
                NOP                     ; (t3)
                SETB    SCL             ; (4) validarlo con SCL
                JNB     SCL,$           ; (t4) espera por si hay sincronización por parte del esclavo
```

```
                NOP                     ; SCL está 5 µs a alta (el mínimo obligatorio son 4'7 en modo normal)
                NOP
                CLR     SCL             ; (5) desactivar SCL
                NOP                     ; SCL está 5 µs a baja (el mínimo obligatorio son 4 en modo normal)
                DJNZ    R7,LANZA_BIT
                RET                     ; SCL está 4 µs a baja en el último bit


; ============================================================================
; =            RUTINA MAESTRA QUE TOMA UN CARÁCTER Y LO RECONOCE             =
; ============================================================================

TOMA_CARÁCTER_Y_RECONOCELO_M:
                MOV     R7,#8           ; hay que tomar los ocho bits del carácter
OTRO_BIT:       CALL    TOMA_BIT
                DJNZ    R7,OTRO_BIT     ; repite si quedan bits por recibir
                CALL    GENERA_RECONOCIMIENTO_SIENDO_MAESTRO
                RET

TOMA_BIT:       SETB    SCL             , valida el bit enviado por el esclavo emisor
                JNB     SCL,$           ; espera por si el esclavo pide sincronización
                NOP
                MOV     C,SDA           ; toma el bit desde la línea de datos SDA
                RLC     A               ; mételo en el acumulador y rótalo a la izquierda
                CLR     SCL             ; termina la validación del bit
                RET


; ============================================================================
; =            RUTINA MAESTRA QUE TOMA UN CARÁCTER SIN RECONOCERLO           =
; ============================================================================

TOMA_CARÁCTER_M:
                MOV     R7,#8           ; hay que tomar los ocho bits del carácter
OTRO_BIT2:      CALL    TOMA_BIT2
                DJNZ    R7,OTRO_BIT2    ; repite si quedan bits por recibir
                CALL    GENERA_NO_RECONOCIMIENTO_SIENDO_MAESTRO
                RET

TOMA_BIT2:      SETB    SCL             , valida el bit enviado por el esclavo emisor
                JNB     SCL,$           ; espera por si el esclavo pide sincronización
                NOP
                MOV     C,SDA           ; toma el bit desde la línea de datos SDA
                RLC     A               ; mételo en el acumulador y rótalo a la izquierda
                CLR     SCL             ; termina la validación del bit
                RET


; ============================================================================
; =                RUTINA MAESTRA QUE PIDE RECONOCIMIENTO                    =
; ============================================================================

; condición de salida: acarreo=0 indica que se ha recibido, acarreo a 1 indica que no

PIDE_RECONOCIMIENTO:

                SETB    SDA             ; asegura que SDA está aislada del bus
                SETB    SCL             ; validarlo
                JNB     SCL,$           ; en previsión de que existe sincronización
                MOV     C,SDA           ; toma el estado de la línea de datos SDA (reconocimiento)
                NOP
                CLR     SCL             ; finalizar la validación
                NOP
                RET


; ============================================================================
; =                RUTINA MAESTRA QUE DA EL RECONOCIMIENTO                   =
; ============================================================================

GENERA_RECONOCIMIENTO_SIENDO _MAESTRO:
                CLR     SDA             ; reconocer
                NOP
VALIDA_REC:     SETB    SCL             ; validarlo
```

   a.caparros.aceyte@gmail.com
Universidad de Córdoba (España). Depto. Arquitectura de Computadores, E. y T. Electrónica

```
                JNB     SCL,$           ; en previsión de que exista sincronización
                NOP
                NOP
                CLR     SCL             ; terminar la validación
                NOP
                SETB    SDA             ; se aísla SDA por si el maestro actúa como receptor
                RET

; ============================================================================
; =                  RUTINA MAESTRA QUE NO DA EL RECONOCIMIENTO               =
; ============================================================================

GENERA_NO_RECONOCIMIENTO_SIENDO_MAESTRO:
                SETB    SDA
                JMP     VALIDA_REC
```

Rutinas básicas en modo esclavo:

```
SDA     EQU     90h     ; dirección de bit de P1.0
SCL     EQU     91h     ; dirección de bit de P1.1
SDA_A   EQU     0E0h+(SDA AND 0Fh)      ; dirección de bit de SDA en el acumulador
SCL_A   EQU     0E0h+(SCL AND 0Fh)      ; dirección de bit de SCL en el acumulador

; ============================================================================
; =                    RUTINA QUE SONDEA SI HAY INICIO                        =
; ============================================================================

SONDEA_INICIO:  SETB    SDA             ; nos aseguramos de que el excitador SDA está aislado del bus
                MOV     C,SDA
                JC      SDA,NO_LO_HAY
                JB      SDA,NO_LO_HAY   ; nos aseguramos, para evitar posibles ruidos espurios
                JB      SCL,$   ; para retornar se espera a que SCL esté inactiva (condición de cambio)
NO_LO_HAY:      CPL     C       ; se retorna con C=0 si no hay inicio, y C=1 si lo hay
                RET

; ============================================================================
; =                    RUTINA QUE SONDEA SI HAY PARADA                        =
; ============================================================================
; Condición de llamada: SCL=0
; condición de salida: C=0 indica que no hay parada; C=1 indica que sí.
SONDEA_PARADA:
                MOV     A,P1                    ; toma simultáneamente SDA y SCL
                JBN     SCL_A,SONDEA_PARADA     ; si SCL=0  entonces esperar validación
                JB      SDA_A,NO_HAY_PARADA     ; si SDA=1 entonces no puede darse una condición parada
ESPERA_PARADA:MOV       A,P1                    ; como SDA=0, volver a sondear SDA y SCL
                JNB     SCL_A,NO_HAY_PARADA     ; fin de validación, es decir, no hay parada
                JNB     SDA_A,ESPERA _PARADA    ; SDA sigue a cero: volver a  sondear SDA y SCL
HAY_PARADA:     SETB    C                       ; SDA ha cambiado a 1 estando SCL a 0: hay parada
                RET
NO_HAY_PARADA:
                CLR     C
                RET

; ============================================================================
; =                  RUTINA ESCLAVA QUE TOMA UN CARÁCTER                      =
; ============================================================================
; condición de entrada: el carácter con la dirección y L/E estará en Acumulador
TOMA_DIRECCIÓN:
TOMA_CARÁCTER_E:
                MOV     R7,#8           ; hay que tomar los ocho bits del carácter
                ; aquí, de ser el caso, iría la sincronización carácter a carácter
TOMA_BIT_E:     JNB     SCL,$           ; espera la validación del bit
                NOP
                MOV     C,SDA           ; toma el bit desde la línea de datos SDA
                RLC     A               ; mételo en el acumulador y rótalo a la izquierda
                JB      SCL,$           ; espera a que termine la validación
                ; aquí, de ser el caso, iría la sincronización bit a bit
                DJNZ    R7,TOMA _BIT_E
                RET
```

```
; ================================================================================
; =                    RUTINA ESCLAVA QUE DA EL RECONOCIMIENTO                   =
; ================================================================================

GENERA_RECONOCIMIENTO_SIENDO_ESCLAVO:
                    CLR     SDA             ; reconoce
                    JNB     SCL,$           ; espera la validación del maestro
                    JB      SCL,$           ; espera el fin de la validación
                    SETB    SDA             ; se aísla SDA del bus
                    RET
```

Las rutinas aquí mostradas abordan el problema de la gestión del protocolo I2C enteramente por programa. Se ha tenido cuidado con el cumplimiento de las temporizaciones; no obstante, y dado que en algunos casos se han desmenuzado en distintos niveles y que se ha supuesto que el microcontrolador funciona a sólo 12 Mhz, puede darse el caso de que se presente algún problema con alguna de las rutinas especificadas. Por ejemplo, en la rutina esclava que da el reconocimiento a un carácter pudiera darse el caso de un comportamiento anómalo si el maestro funciona en modo rápido (400 kbps), puesto que éste podría validar el reconocimiento antes de que el esclavo receptor, para reconocer, ponga a cero la línea SDA. No obstante, se deja a la perspicacia del lector la adecuada adaptación del código propuesto para satisfacer las condiciones de temporización del caso concreto que pueda considerarse.
Igualmente, se deja para el lector la construcción de rutinas que implementen la fase de arbitraje así como otras de capa lógica superior que realicen las funciones concretas propias de un determinado circuito.

Autor: Antonio Moreno Fernández-Caparrós
Fecha del borrador: Abril, 2004. (Córdoba, España)

Referencia bibliográfica:

- The I2C Bus Specification (version 2.1, January 2000)
  URL http://www.semiconductors.philips.com/acrobat/literature/9398/39340011.pdf