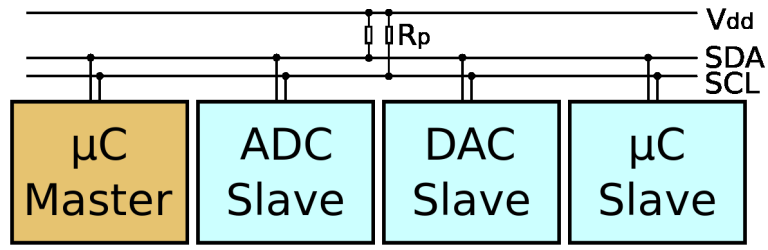
I²C



Un ejemplo esquemático con un maestro (un microcontrolador) y tres nodos esclavos (un ADC, un DAC, y otro microcontrolador) con resistencias pull-up R_p.

I²C es un bus de comunicaciones en serie. Su nombre viene de *Inter-Integrated Circuit* (Inter-Circuitos Integrados). La versión 1.0 data del año 1992 y la versión 2.1 del año 2000, su diseñador es Philips. La velocidad es de 100 kbit/s en el modo estándar, aunque también permite velocidades de 3.4 Mbit/s. Es un bus muy usado en la industria, principalmente para comunicar microcontroladores y sus periféricos en sistemas integrados (*Embedded Systems*) y generalizando más para comunicar circuitos integrados entre si que normalmente residen en un mismo circuito impreso.

La principal característica de **I²C** es que utiliza dos líneas para transmitir la información: una para los datos y otra para la señal de reloj. También es necesaria una tercera línea, pero esta sólo es la referencia (masa). Como suelen comunicarse circuitos en una misma placa que comparten una misma masa esta tercera línea no suele ser necesaria.

Las líneas se llaman:

- SDA: datos
- SCL: reloj
- GND: tierra

Las dos primeras líneas son drenador abierto, por lo que necesitan resistencias de pull-up.

Los dispositivos conectados al bus **I²C** tienen una dirección única para cada uno. También pueden ser *maestros* o *esclavos*. El dispositivo *maestro* inicia la transferencia de datos y además genera la señal de reloj, pero no es necesario que el *maestro* sea siempre el mismo dispositivo, esta característica se la pueden ir pasando los dispositivos que tengan esa capacidad. Esta característica hace que al bus **I²C** se le denomine bus multimaestro.

Las transacciones en el bus I2C tienen este formato:

| start | A7 A6 A5 A4 A3 A2 A1 R/W | ACK | ... DATA ... | ACK | stop | idle |

- El bus esta libre cuando SDA y SCL están en estado lógico alto.
- En estado bus libre, cualquier dispositivo puede ocupar el bus **I²C** como maestro.
- El maestro comienza la comunicación enviando un patrón llamado "start condition". Esto alerta a los dispositivos esclavos, poniéndolos a la espera de una transacción.
- El maestro se dirige al dispositivo con el que quiere hablar, enviando un byte que contiene los siete bits (A7-A1) que componen la dirección del dispositivo esclavo con el que se quiere comunicar, y el octavo bit (A0) de menor peso se corresponde con la operación deseada (L/E), lectura=1 (recibir del esclavo) y escritura=0 (enviar al esclavo).
- La dirección enviada es comparada por cada esclavo del bus con su propia dirección, si ambas coinciden, el esclavo se considera direccionado como esclavo-transmisor o esclavo-receptor dependiendo del bit R/W.
- El esclavo responde enviando un bit de ACK que le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse.
- Seguidamente comienza el intercambio de información entre los dispositivos.
- El maestro envía la dirección del registro interno del dispositivo que se desea leer o escribir.
- El esclavo responde con otro bit de ACK

- Ahora el maestro puede empezar a leer o escribir bytes de datos. Todos los bytes de datos deben constar de 8 bits, el número máximo de bytes que pueden ser enviados en una transmisión no está restringido, siendo el esclavo quien fija esta cantidad de acuerdo a sus características.
- Cada byte leido/escrito por el maestro debe ser obligatoriamente reconocido por un bit de ACK por el dispositivo maestro/esclavo.
- Se repiten los 2 pasos anteriores hasta finalizar la comunicación entre maestro y esclavo.
- Aun cuando el maestro siempre controla el estado de la línea del reloj, un esclavo de baja velocidad o que deba detener la transferencia de datos mientras efectúa otra función, puede forzar la línea SCL a nivel bajo. Esto hace que el maestro entre en un estado de espera, durante el cual, no transmite información esperando a que el esclavo esté listo para continuar la transferencia en el punto donde había sido detenida.
- Cuando la comunicación finaliza, el maestro transmite una "stop condition" para dejar libre el bus.
- Después de la "stop condition", es obligatorio para el bus estar idle durante unos microsegundos.

El código del kernel de Linux para el soporte I2C está separado en varias piezas lógicas:

- I2C chip driver (maneja uno de los chips conectados al bus I2C, tanto si se comporta como maestro o como esclavo)
- I2C bus driver
- I2C algorithm driver
- I2C core (la parte genérica del subsistema de I2C)

Alternativas

- SMBus
- SPI

Enlaces externos

- I2C NXP Semiconductors [1]
- I2C al detalle [2]
- I2C, descripción y funcionamiento [3]

Referencias

[1] http://www.nxp.com/products/interface_control/i2c/
[2] http://www.i2c-bus.org/
[3] http://robots-argentina.com.ar/Comunicacion_busI2C.htm

Fuentes y contribuyentes del artículo

I²C *Fuente*: http://es.wikipedia.org/w/index.php?oldid=75029532 *Contribuyentes*: Dondervogel 2, Gaijin, GermanX, Lelguea, Lsaavedr, Marcelo Huerta, Murphy era un optimista, PACO, Sanbec, SoMoS, Xuankar, 33 ediciones anónimas

Fuentes de imagen, Licencias y contribuyentes

Archivo:I2C.svg *Fuente*: http://es.wikipedia.org/w/index.php?title=Archivo:I2C.svg *Licencia*: GNU Free Documentation License *Contribuyentes*: en:user:Cburnett

Licencia