17-3-2020

Lisbeth Martínez Velázquez

ing.mecatrónica 8-at/m prof:CARLOS gARABITO



I2C

Programación de sistemas de embebidos

**I2C**

I2C es un bus de comunicaciones serie. Su nombre viene de Inter-Integrated Circuit (Circuitos Inter-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 100Kbits por segundo en el modo estándar. Es un bus muy usado en la industria, principalmente para comunicar microcontroladores y sus periféricos en sistemas embebidos (Embedded Systems).

La principal característica de I2C es que sólo usa dos hilos para transmitir la información: por uno van los datos y por otro la señal de reloj que sirve para sincronizarlos. También es necesaria una tercera línea, pero esta sólo es la referencia (masa). Las líneas se llaman:

**SDA:**datos

**SCL:** reloj

**GND:**masa

Las dos primeras líneas son drenado 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.

Muchos de los equipos electrónicos que nos llegan al taller incluyen circuitos integrados con el bus I2C, como por ejemplo, las memorias 24Cxx, los procesadores de señal o "jungla" en televisores (LA7610, TA1223, DTC810), codificadores de video de reproductores de DVD (SAA 7128, TC 90A32F,...), preamplificadores de video en monitores (KB 2502), etc.

Muchas veces al reparar estos equipos nos encontramos con la dificultad de no comprender la lógica de estos circuitos, lo que nos impide realizar un buen diagnóstico.

Lo que sigue es una breve descripción del bus I2C y algunas recomendaciones para diagnosticar fallas en el bus.

**Las características más salientes del bus I2C son:**

* Se necesitan solamente dos líneas, la de datos (SDA) y la de reloj (SCL).
* Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software. Habiendo permanentemente una relación Master/ Slave entre el micro y los dispositivos conectados
* El bus permite la conexión de varios Masters, ya que incluye un detector de colisiones.
* El protocolo de transferencia de datos y direcciones posibilita diseñar sistemas completamente definidos por software.
* Los datos y direcciones se transmiten con palabras de 8 bits.

**Funcionamiento del bus I2C**

Como dijimos, las líneas SDA y SCL transportan información entre los dispositivos conectados al bus (figura 1).

Cada dispositivo es reconocido por su código (dirección) y puede operar como transmisor o receptor de datos. Además, cada dispositivo puede ser considerado como Master o Slave.

| http://usuaris.tinet.cat/fbd/comunicaciones/i2cbus/I2C_html_m5abc26ce.gif |
| --- |

El Master es el dispositivo que inicia la transferencia en el bus y genera la señal de Clock.

**El Slave (esclavo)** **es el dispositivo direccionado.**

Las líneas SDA (serial Data) y SCL (serial Clock) son bidireccionales, conectadas al positivo de la alimentación a través de las resistencias de pull-up. Cuando el bus está libre, ambas líneas están en nivel alto.

La transmisión bidireccional serie (8-bits) de datos puede realizarse a 100Kbits/s en el modo standard o 400 Kbits/s en el modo rápido.

La cantidad de dispositivos que se pueden conectar al bus está limitada, solamente, por la máxima capacidad permitida: 400 pF.

**Condiciones de START y STOP:**

Antes de que se establezca un intercambio de datos entre el circuito Master y los Esclavos, el Master debe informar el comienzo de la comunicación (condición de Start): la línea SDA cae a cero mientras SCL permanece en nivel alto. A partir de este momento comienza la transferencia de datos. Una vez finalizada la comunicación se debe informar de esta situación (condición de Stop). La línea SDA pasa a nivel alto mientras SCL permanece en estado alto. Ver Figura 2

**Transferencia de datos:**

El Maestro genera la condición de Start. Cada palabra puesta en el bus SDA debe tener 8 bits, la primera palabra transferida contiene la dirección del Esclavo seleccionado.

Luego el Master lee el estado de la línea SDA, si vale 0 (impuesto por el esclavo), el proceso de transferencia continúa. Si vale 1, indica que el circuito direccionado no valida la comunicación, entonces, el Maestro genera un bit de stop para liberar el bus I2C.

Este acuse de recibo se denomina ACK (acknowledge) y es una parte importante del protocolo I2C.

Al final de la transmisión, el Maestro genera la condición de Stop y libera el bus I2C, las líneas SDA y SCL pasan a estado alto.

**Fallas en el bus:**

Ante una falla en el funcionamiento de alguno de los integrados conectados al bus y antes de probar cambiando CIs, tenemos que hacer algunas verificaciones.

Como primera medida comprobamos el estado de las resistencias de pull up.

Luego con una punta lógica verificamos que cuando el bus está inactivo, las dos líneas se encuentren en estado alto.

A continuación, chequeamos que el Máster direccione alguno de los integrados, esta verificación también podemos hacerla con la punta lógica, pero tengamos en cuenta dos cosas: con la punta lógica solamente estamos detectando actividad en el bus, pero no podemos saber si se lleva a cabo satisfactoriamente.

En segundo lugar, si hay más de un integrado conectado al bus, no podremos determinar cuál de ellos está siendo solicitado.

Para analizar más a fondo las comunicaciones y avanzar en el diagnóstico, tendremos que contar con un osciloscopio digital o construir un sencillo probador con unos pocos componentes (ver: Analizador del bus I2C).

Un caso especial son las memorias 24Cxx que trabajan con el protocolo I2C y son utilizadas en televisores, monitores, reproductores de DVD, etc.

Será muy útil hacer un back up de cada una de las memorias que llegan a nuestro taller, en muchos casos nos puede interesar copiarlas sin desoldarlas del circuito y sin necesidad de la PC, para lo cual podemos utilizar el probador mencionado anteriormente (para más información ver Proyectos en CeCaT).

Si utiliza una PC, puede construir el Programador de EEPROM (24Cxx y 24LCxx)

En algunos casos nos encontraremos con microcontroladores que tengan más de un bus I2C, aplicaremos la misma lógica de análisis indicada anteriormente, pero teniendo en cuenta que en este caso puede haber comunicación en tiempo compartido.

**Bibliografía:**

[1] Manuel Coutinho, José Rufino, Carlos Almeida, “Control of Event Handling

Timeliness in RTEMS”, Proceedings of the 17th IASTED International Conference

on Parallel and Distributed Computing and Systems (PDCS 2005), Phoenix, AZ,

USA, November 14 - 16, 2005.

[2] Helen Custer, “Inside Windows NT”, Microsoft Press, 1992.

[3] A. Damm, J. Reisinger, W. Schwabl, y H. Kopetz. “The Real-Time Operating

System of MARS”. Operating System Review, 23(3): 141-157, Julio 1989.