¿Que es el I2C?

El bus de comunicaciones I2C (nombrado a veces como I cuadrado C, I^2) es un protocolo que se efectúa por medio de DOS hilos. A través de estos dos hilos pueden conectarse diferentes dispositivos donde algunos de ellos serán MAESTROS en cuanto muchos otros dispositivos serán ESCLAVOS.

¿Como trabaja la comunicación I2C?

Para poder reconocer cada uno de los dispositivos conectados a los DOS hilos del bus I2C, a cada dispositivo se le asigna una dirección.

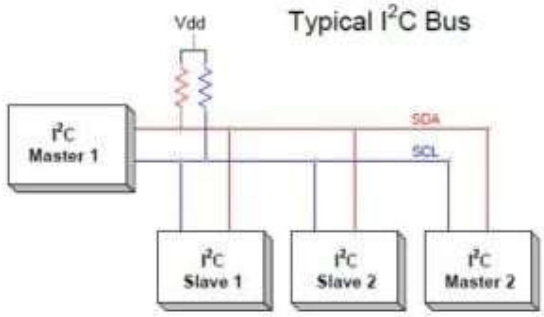
Así en este tipo de comunicaciones el MAESTRO es el que tiene la iniciativa en la transferencia y este es quien decide con quien se quiere conectar para enviar y recibir datos y también decide cuando finalizar la comunicación.

Los DOS hilos del BUS interfaz de comunicación I2C PIC son lineas de colector abierto donde una de las lineas lleva la señal de reloj y es conocida como (SCL), y la otra linea lleva los datos y es conocida como (SDA).

Los Pines SDA y SDL I2C se encuentran especificados en todos los componentes que usan este tipo de protocolo de comunicación.

Para que la comunicación funcione se deben utilizar unas resistencias PULL UP (resistencias conectadas a positivo) para asegurar un nivel alto cuando NO hay dispositivos conectados al BUS I2C.

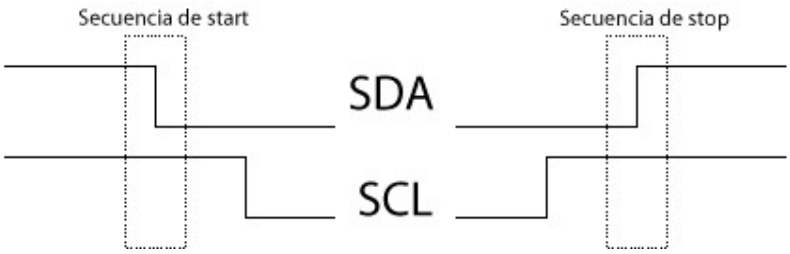
La conexión I2C entre un maestro y vários esclavos se muestra a continuación:



Conexión I2C

El número de dispositivos que pueden conectarse de esta forma y además la longitud del BUS es limitada por la capacidad de direccionamiento (de 7 a 10 bits) y por la máxima carga del BUS (400pF). La velocidad máxima estándar es de 100Kbps.

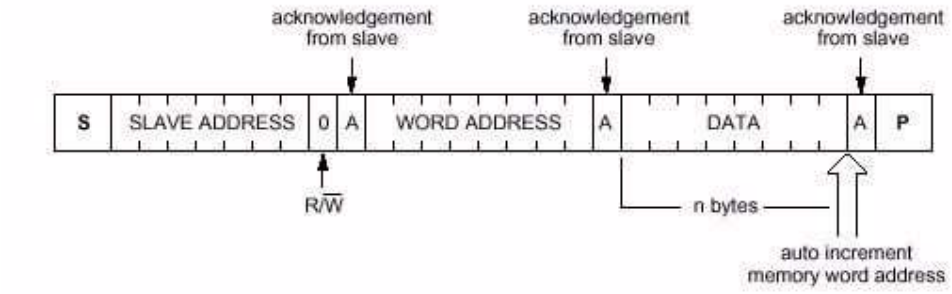
La transmisión de datos se inicia con un bit de inicio (START) y termina con un bit de finalización (STOP). El bit de START se reconoce porque la linea SDA pasa de un estado lógico alto para un estado lógico de bajo cuando la linea SCL esta en nivel alto. El STOP se establece cuando hay una transición de bajo a alto en la linea SDA, cuando SCL está en un nivel alto.



Cuando comienza la transmisión de datos, el MAESTRO envía la dirección del ESCLAVO con el cual se quiere comunicar, esta dirección puede ser de 7 o 10 bits con formato de byte (uno o dos bits respectivamente) Después de la dirección se adiciona 1 bit, que indica si se desea ESCRIBIR o LEER (R/W).

Cuando el Maestro envía estos DATOS para el esclavo. El ESCLAVO debe responderle al maestro con un bit de confirmación para informarle que escucho la solicitud del maestro y que esta a disposición de lo que él necesite. Este bit de confirmación se conoce como (ACK).

Si el maestro NO recibe este bit, la comunicación se interrumpe. Por otro lado, se puede dar el caso que un ESCLAVO está mandando alguna información al maestro, entonces el maestro también generará este bit de confirmación hacia el esclavo.



Puede darse el caso también de que una vez el MAESTRO se comunica con el ESCLAVO, el MAESTRO no abandone el BUS y continúe comunicándose con el ESCLAVO, para eso el MAESTRO debe generar una nueva condición de START que se conoce en la literatura como START REPETIDA (Sr), idéntica al START anterior solo que con un pulso de reconocimiento.

Para trabajar con el modulo de comunicación I2C PIC, se deben configurar los siguientes registros: SSPCON, SSPCON2, SSPADD, SSPBUF, SSPSTAT y SSPSR.

Protocolo de Comunicación I2C con CCS C

En CCS C PIC C, existen funciones que nos facilitan la implementación de este tipo de comunicación. Comencemos primero viendo el tipo de configuraciones que podemos llevar a cabo:

Para las configuraciones genéricas del i2c declaramos lo siguiente en el encabezado de nuestro programa:

#use i2c (opciones)

Estas opciones pueden ir separadas por comas y pueden ser cualquiera de las siguientes opciones

MULTI_MASTER	Establece modo Multimaestro.
MASTER	Establece modo maestro.
SLAVE	Establece modo esclavo.
SCL=pin	Especifica el pin **SCL**.
SDA=pin	Especifica el pin **SDA**.
ADDRESS=nn	Especifica la dirección en modo esclavo.
FAST	Utiliza velocidad alta.
SLOW	Utiliza velocidad baja.
RESTART_WDT	Borra el **WDT** mientras espera una lectura.
FORCE_HW	Utiliza las funciones *I²C* hardware.
NOFLOAT_HIGH	No permite señales flotantes.
SMBUS	Utiliza el bus en formato *SMBUS*.
STREAM=id	Asocia un identificar *stream*.

La directiva (#use i2c) tiene efecto en las funciones: I2C_START, I2C_STOP, I2C_READ. I2C_WRITE y I2C_POLL. Se utilizan funciones de Software a menos que se especifique FORCE_HW.

Como por ejemplo:

#USE I2C(master, sda=PIN_B0, scl=PIN_B1)

#USE I2C(slave, sda=PIN_C4, scl=PIN_C3, address=0XA0, force_hw)

#USE I2C(master, sda=PIN_B0, scl=PIN_B1, fast=450000)

Las funciones asociadas al i2c son:

i2c_start();

Para el modo maestro, esta función inicia la trasmisión. Después de la condición de arranque o START, el reloj es colocado en nivel lógico bajo hasta el momento en que se escriba con la función I2C_WRITE(); Si se llama otra función I2C_START antes que un I2C_STOP quiere decir que se está utilizando un START Repetido (Sr). Esta función depende de la respuesta del esclavo.

i2c_stop();

Finaliza la transmisión.

i2c_write(DATO);

En esta función «DATO» es un entero de 8 bits que se envía por el bus. Cuando un dispositivo que es MAESTRO aplica esta instrucción , se genera una señal de reloj que marca la velocidad de transmisión del dato; Cuando un ESCLAVO aplica esta instrucción se queda esperando la señal de reloj que genere el MAESTRO.

Esta función devuelve el bit de reconocimiento ACK que le envía el receptor cuando la transmisión ha terminado: 0 indica ACK; 1 indica NOACK y 2 indica una colisión en el modo MULTIMASTER.

Para saber la dirección de la transmisión, se observa el bit de menor peso (LSB) del primer dato transmitido tras un START (Si el bit es «0» quiere decir que la información se está transmitiendo de MAESTRO a ESCLAVO)

Dato=i2c_read();

Dato2=i2c_read([ack]);

Dato es un entero de 8 bits leído del BUS. Cuando un dispositivo que es MAESTRO aplica esta instrucción , se genera una señal de reloj que marca la velocidad de transmisión del dato; Cuando un ESCLAVO aplica esta instrucción se queda esperando la señal de reloj que genere el MAESTRO. No hay timeout por lo que esta instrucción se utiliza junto con i2c_poll() para prevenir bloqueos.

Opcionalmente se puede incluir un ACK donde 1 indica ACK; 0 indica NOACK

valor=i2c_poll();

Se utiliza solo si el PIC tiene modulo SSP. Devuelve un TRUE(1) si se ha recibido el dato en el buffer y un FALSE(0) si no se ha recibido. Cuando devuelve el TRUE(1) la función i2c_read() guarda el dato recibido.

estado=i2c_isr_state();

Se utiliza solo si el PIC tiene modulo SSP. Devuelve el estado del bus en modo ESCLAVO después de una interrupción.

Estado es un entero de 8 bits.

i2c_slaveaddr(int8 adr);

Se especifica la dirección del dispositivo en modo esclavo.