Práctica 3: Bus I2C y memoria EEPROM

Prof. Alberto A. Del Barrio

1 Objetivos

En esta tercera práctica continuaremos profundizando en la utilización de la S3CEV40. En concreto:

- Estudiaremos el uso del bus IIC, también conocido como bus I2C, y cuyos principios teóricos se estudian en el tema 3.
- Leeremos y escribiremos en la memoria EEPROM disponible en la placa.
- Utilizaremos estos periféricos en combinación con los periféricos anteriormente estudiados.

2 Memoria EEPROM

EEPROM o E^2PROM son las siglas de *Electrically Erasable Programmable Read-Only Memory* (ROM programable y borrada eléctricamente). Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente, a diferencia de la EPROM que ha de borrarse mediante un dispositivo que emite rayos ultravioleta.

Estas memorias son no volátiles, como las ROM, y pueden leerse un número ilimitado de veces. Sin embargo, solo pueden ser borradas y reprogramadas entre cien mil y un millón de veces.

La EEPROM presente en la placa S3CEV40 es la *AT24C04* [2], fabricada por Atmel. En este caso, es una memoria de 4 kB conectada a la placa por medio de la interfaz I2C. En concreto, está conectada a los pines 1 y 2 del puerto F. Los 4 kB están organizados en 512 palabras de 8 bits, es decir, la AT24C04 dispone de 512 direcciones. Asimismo, posee dos velocidades de transmisión: la estándar (a 100 kb/s) y la rápida, o modo *fast*, a 400 kb/s. Debe notarse no obstante, que dichas velocidades son consecuencia del bus I2C.

Permite dos tipos de escritura:

- 1 byte.
- 1 página (16B).

y tres tipos de lectura:

- 1 byte. Este modo requiere primero una escritura ficticia.
- Secuencial.
- Última dirección accedida + 1.

Para realizar dichas operaciones habrá que acceder a la dirección de la EEPROM (0x80) y sumarle un desplazamiento debido a la posición que queramos leer/escribir.

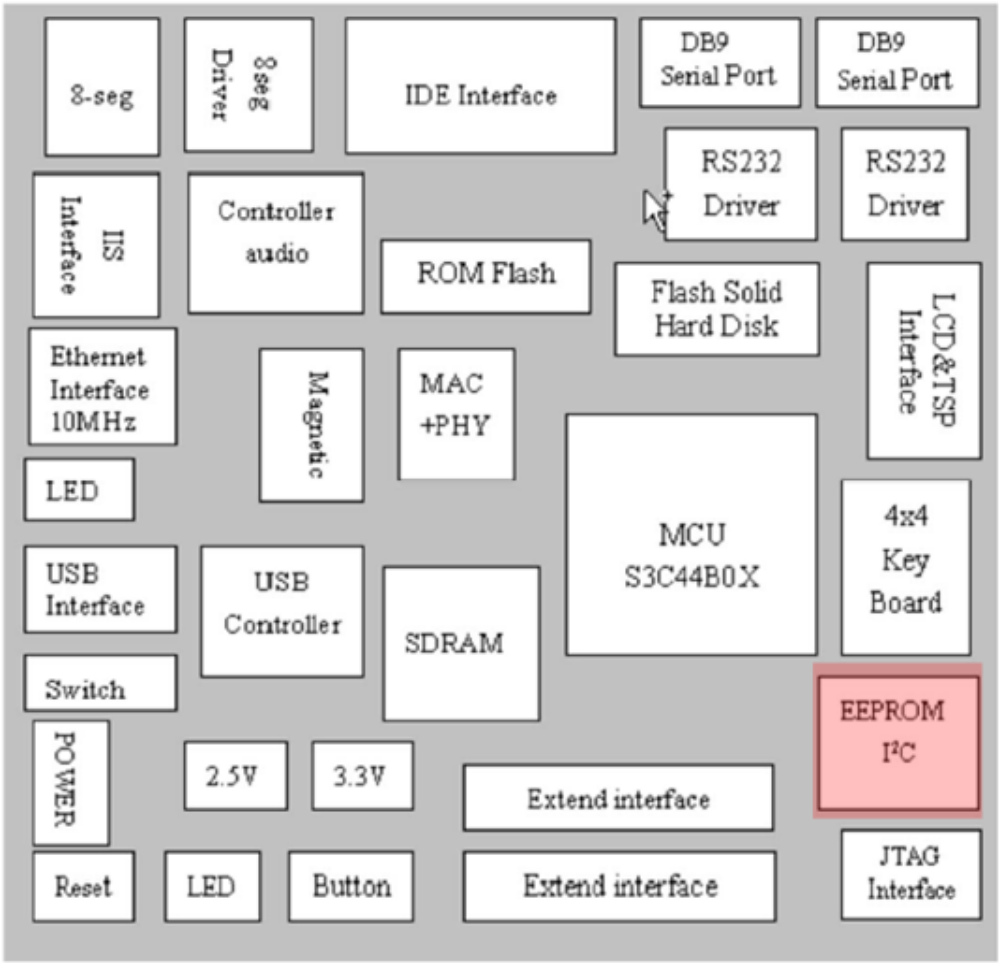


Figura 1: Localización de la memoria EEPROM dentro de la placa S3CEV40

3 Bus I2C

El microprocesador S3C44B0X tiene soporte para una interfaz *Inter Integrated Circuits*, también llamada IIC o I2C. Dicha interfaz es multi-máster y serie. Este bus está compuesto por una línea de datos dedicada (*SDA*) y otra línea para mandar el reloj (*SCL*) entre los *másters* y *slaves* que están conectados por I2C. Ambas líneas son serie y bidireccionales. Cuando el bus I2C esté libre, tanto SDA como SCL deberán estar en alta ('1' lógico).

En modo multi-máster, varios microprocesadores (másters) podrían enviar o recibir datos a/desde periféricos ejerciendo de slaves, aunque en nuestro caso solo trabajaremos con un microprocesador: el S3C44B0X. Ha de notarse que el máster siempre será el encargado de iniciar y terminar las transferencias en el bus.

Para controlar las operaciones sobre el bus I2C, debemos acceder a los siguientes registros:

- IICCON. Este es el registro de control del bus.

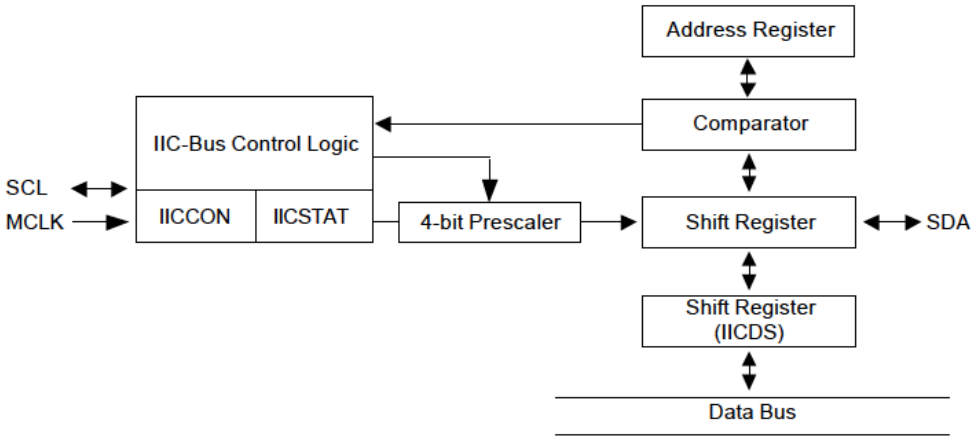


Figura 2: Diagrama de bloques del bus I2C

- IICSTAT. Este es el registro de estado del bus.
- IICADD. Este es el registro de direcciones del bus.
- IICDS. Este registro de desplazamiento se encarga de almacenar los datos que serán transmitidos (modo Tx) o recibidos (modo Rx). Es un registro de desplazamiento ya que en cada ciclo se enviará un bit en serie.

El diagrama de bloques aparece en la figura 2. Como puede observarse, además de los registros anteriormente mencionados, hay un módulo de preescalado de 4-bits y un comparador. Dicho módulo se utiliza para regular la frecuencia del reloj serie (SCL), mientras que el comparador sirve para comparar las direcciones que se ponen en el bus.

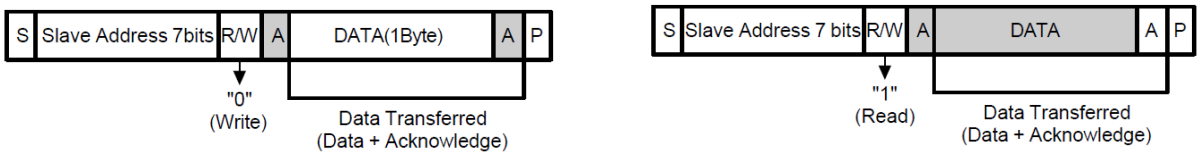
3.1 Protocolo de transmisión

Las direcciones de los dispositivos pueden ser de 7b ó 10b. Para explicar este apartado, supondremos que es de 7b. Con estas condiciones, el protocolo básico de comunicación consiste en los siguientes pasos:

1) El máster inicia la transmisión mandando el símbolo *start* al bus (transición 1-0 en SDA). A continuación todos los slaves se ponen en alerta.

2) El máster envía la dirección del slave (7b) y el tipo de operación: lectura ('1') o escritura ('0'). La figura 3 muestra cómo se realiza este paso tanto para lectura como para escritura. Nótese que los símbolos S, P y A representan las condiciones de *start*, *stop* y ACK, respectivamente.

   – Todos los slaves comparan la dirección con la suya y el slave aludido envía el mensaje de ACK.



(a) Modo escritura

(b) Modo lectura

Figura 3: Transmisión de la dirección del slave de 7-bits para a) escritura y b) lectura

– Si la dirección fuera de 10b, se enviaría fragmentada en dos mensajes.

3) Se transmiten un número indefinido de datos (en cada mensaje 8b) reconocidos individualmente (cada mensaje con un ACK de 1b). Esto puede verse en la figura 4. Como puede observarse, el símbolo ACK consiste en una senñal de baja en la línea SDA en el noveno pulso transmitido por SCL.

4) El máster finaliza la transmisión enviando el símbolo *stop* al bus (transición 0-1 en SDA).

3.2 Configurando el I2C

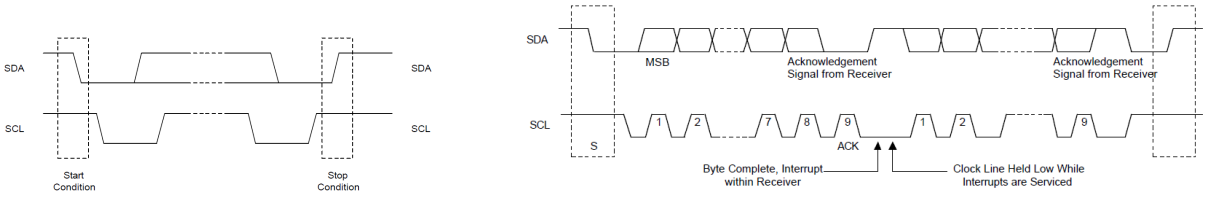
Al igual que en las prácticas anteriores tendremos que definir un método de inicialización para configurar el bus I2C. En este método tendremos que seguir los siguientes pasos:

1) Activar los bloques IIC, GPIO, BDMA a través del registro CLKCON.

2) Configurar los pines 0 y 1 del puerto F como IICSCL y como IICSDA, respectivamente. Esto se hace en el registro PCONF.

3) Activar las resistencias de pull-up de los pines 0 y 1 a través del registro PUPF.

4) Escribir la dirección del slave en el registro IICADD. Esto se hará utilizando la macro S3C44B0X_SLAVE_ADDRESS, mapeándola sobre los bits correspondientes del registro.



(a) Símbolos de *start* y *stop*

(b) Trasferencia de datos

Figura 4: Trasferencia de datos con símbolos de *start*, *stop* y *ACK*

5) Habilitar el uso de interrupciones y de mensajes de ACK a traveés del registro IICCON. Además, asignar el valor de preescalado.

6) Activar Rx/Tx a través del registro IICSTAT.

Debe tenerse en cuenta que nuevamente cada registro mencionado anteriormente tiene una variable asociada que empieza por r. Por ejemplo, rCLKCON o rIICSTAT.

Nota: cada vez que se quiera actualizar el contenido de IICSTAT, se recomienda no realizar operaciones lógicas sobre él, sino sobreescribirlo directamente.

4 Desarrollo de la práctica

Esta práctica está dividida en dos partes: una primera parte en la que habrá que configurar el bus I2C y usar la EEPROM y una segunda parte en la que incorporaremos otros periféricos ya estudiados.

4.1 Utilizando la EEPROM y el bus I2C

Para realizar esta parte de la práctica en primer lugar tendremos que descargar los ficheros *pr3Skeleton.rar* y *commonEclipseIIC.rar*. Como siempre, descomprimimos ambos ficheros y creamos un proyecto de forma similar a las anteriores prácticas.

Además de construir el proyecto y configurarlo como ya hemos visto, debemos situarnos en *Project* → *Properties* → *C/C++ Build* → *Settings*. En la pestaña *Tool Settings* iremos a *ARM Sourcery Windows GCC C Linker* y añadiremos lo siguiente:

- En *Libraries* las librerías *c*, *gcc* y *nosys*.
- En *Library search path* las rutas *${eclipse_home}/../sourcery-g++-lite-arm-2011.03/arm-none-eabi/lib* y *${eclipse_home}/../sourcery-g++-lite-arm-2011.03/lib/gcc/arm-none-eabi/4.5.2*.

Después de configurar el proyecto, habrá que completar los métodos que aparecen en el fichero *iic.c*. Si se completa correctamente, el programa debería inicializar la EEPROM y en el último bucle for mostrar los elementos de la memoria en el 8-segmentos.

4.1.1 Depurando con el I2C

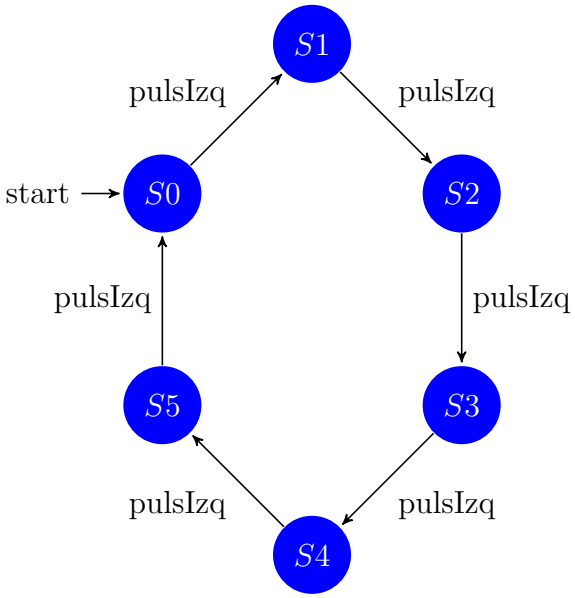
Dado que el contenido de la EEPROM tan solo es accesible a través del I2C, cuyos métodos hay que programar, esta práctica es difícil de depurar. Por ello, dentro de la carpeta *commonEclipse* se proporciona la librería *libsedLib.a*, que contiene funciones equivalentes a las

dadas en los ficheros *iic.c* y *at24c04.c* pero funcionales. La única diferencia consiste en el nombre de dichas funciones, que comienza por el prefijo *golden_*. Por ejemplo, si queremos utilizar las función de test correspondiente al método *at24c04_bytewrite*, habrá que invocar a la función *golden_at24c04_bytewrite* en su lugar.
Además del cambio de nombre, es necesario incluir la librería *sedLib* en la configuración de las librerías utilizadas por el Linker, y añadir *commonEclipse* al path de dichas librerías.

4.2 Combinando con otros periféricos

En esta segunda parte, se propone implementar un sistema que siga el comportamiento dado por la máquina de estados de la figura 5a. Este sistema leerá una dirección de 8 bits y un dato de 8 bits, y lo escribirá en la EEPROM, de acuerdo a la especificación dada por la tabla 5b.

Como puede observarse, las transiciones de estados suceden cada vez que se acciona el pulsador izquierdo. Las direcciones y los datos se leerán del teclado matricial. Además, **en cualquier momento** que se accione el pulsador derecho, el sistema deberá ser capaz de leer de la EEPROM la palabra que esté en la última dirección introducida, y mostrarla en el 8-segmentos.
Por tanto, es preciso añadir al proyecto las clases *button.c* y *keyboard.c*, configurar las interrupciones como hemos visto en las anteriores prácticas e implementar las rutinas de tratamiento de las mismas.



(a) Máquina de estados

Estado	Significado
S0	Estado inicial
S1	Lee la mitad más significativa de la dirección
S2	Lee la mitad menos significativa de la dirección
S3	Lee la mitad más significativa del dato a escribir
S4	Lee la mitad menos significativa del dato a escribir
S5	Escribe el dato en la dirección de la EEPROM

(b) Descripción de los estados

Figura 5: Comportamiento de la segunda parte

Bibliografía

[1] S3C44B0X RISC Microprocessor. Accesible en el Campus Virtual.

[2] AT24C04 Datasheet. Accesible en el Campus Virtual.