Laboratorio de Estructura de Computadores
Curso 2007-08
- 2ª Parte -

Práctica 5

Manejo de comunicaciones serie (unidad UART e interfaz IIC) y EEPROM

5.1. Práctica 5 - Apartado A. Manejo de la UART

5.1.1. Objetivos de la práctica

- Familiarizarse con la comunicación serie asíncrona.
- Manejo de la unidad UART del S3C44BOX.

5.1.2. Conceptos teóricos

Unidad UART del S3C44BOX

La unidad UART (*Universal Asynchronous Receiver and Transmitter*) proporciona dos puertos serie asíncronos independientes: UART0 y UART1. Estos puertos, también denominados canales, permiten una comunicación serie bidireccional de hasta 115.2 Kbps y su gestión puede llevarse a cabo mediante encuesta (*polling*), interrupciones o DMA. Cada canal consta de los siguientes elementos (ver figura 5.1): generador de frecuencia (baudios), módulo transmisor, módulo receptor y unidad de control. El generador de frecuencia emplea como entrada el reloj principal del sistema (MCLK). Tanto el transmisor como el receptor se componen de una FIFO de 16 bytes y de un registro de desplazamiento. Estas colas FIFO, si están activas, permiten desacoplar aún más la CPU y la comunicación de los datos por el puerto serie. En lugar de ir escribiendo/leyendo los caracteres uno a uno, la CPU escribe/lee bloques de hasta 16 caracteres. Es preciso destacar, no obstante, que la comunicación a través de la línea serie se realiza carácter a carácter debido a la naturaleza asíncrona de la misma. Los registros de desplazamiento se encargan de ir transmitiendo o recibiendo los caracteres, bit a bit, por la línea correspondiente TXDn o RXDn (donde n representa el nº del puerto).

Otras características destacables de la unidad UART son:

- Velocidad de transmisión programable.
- Capacidad de transmisión/recepción siguiendo el estándar de infrarrojos IrDA 1.0.
- Marco (*frame*) de datos programable (nº de bits de stop, tamaño de carácter, chequeo de paridad, ...).

A continuación describiremos brevemente el funcionamiento de la UART. La información detallada se encuentra en el Capítulo 10 del Manual de Referencia del S3C44B0X.

Transmisión de datos

Cada carácter a enviar por el puerto serie requiere información adicional para su correcta transmisión: delimitadores de comienzo y fin de carácter (1 bit de *Start* y 1 o 2 bits de *Stop*), y opcionalmente un bit para ajustar la paridad. Conviene señalar que los bits de comienzo y fin de carácter tienen un valor opuesto para garantizar la presencia de una transición al comienzo de la transmisión.

Al conjunto de bits formado por el propio carácter y estos bits adicionales se les conoce como marco de datos serie (*Serial I/O Frame*). En la UART del S3C44BOX su formato

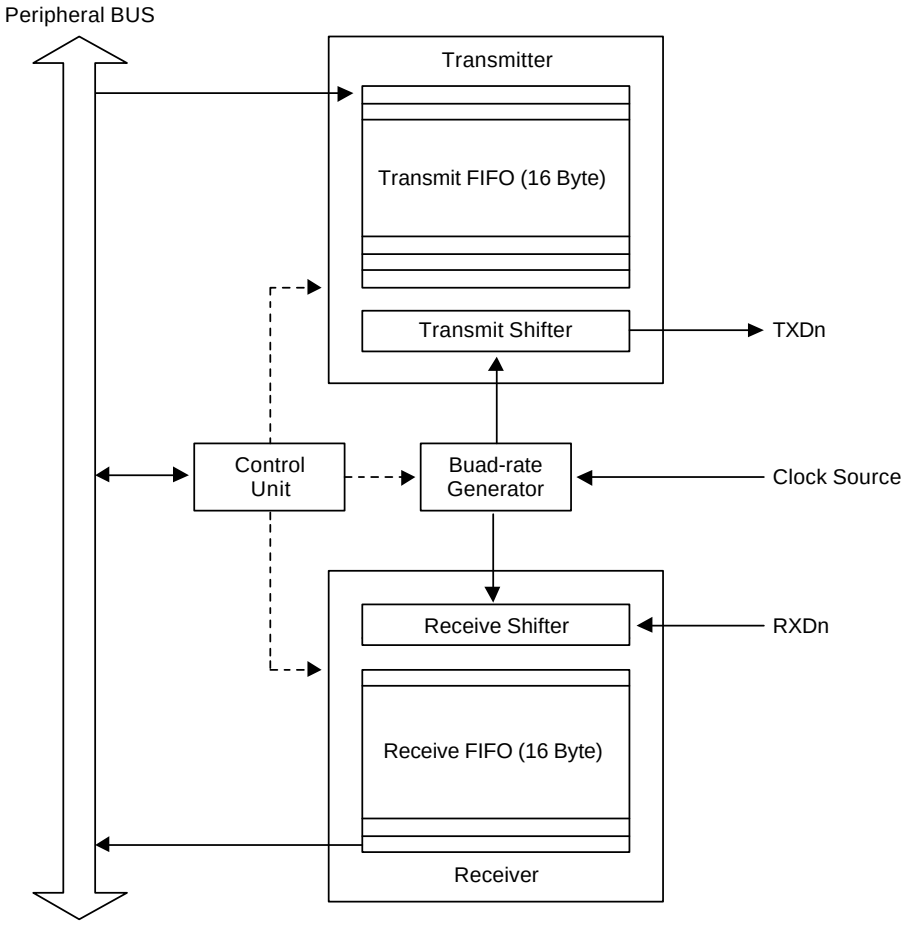

Figura 5.1: Estructura de un canal de la UART.

se configura mediante el registro de control de línea (UCONn). La figura 5.2 muestra un ejemplo de marco con el siguiente formato: 1 bit de *Start*, carácter de 8 bits, 1 bit de *Stop* y sin bit de paridad.

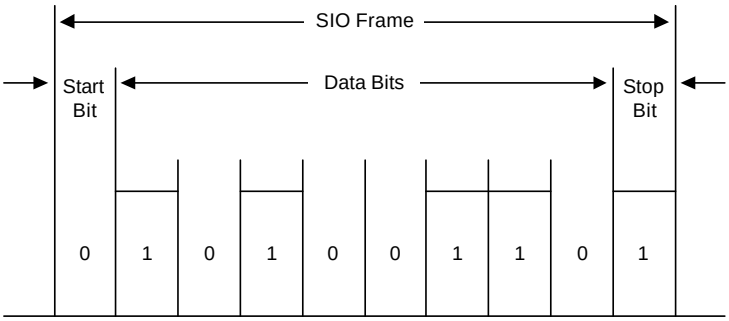

Figura 5.2: Diagrama temporal de un marco de datos serie (*SIO Frame*) con un tamaño de carácter de 8 bits, 1 bit de stop, sin paridad.

El transmisor también puede producir un *frame* de pausa o *break* que consiste en una señal de 0 lógico de un frame de duración. El propósito de este tipo de marco es informar al receptor de una pausa en la comunicación.

Recepción de datos

Al igual que para la transmisión, el formato del *frame* es configurable (registro UCONn). Como es lógico, para una correcta comunicación es necesario que la configuración en ambos sea la misma.

El receptor además es responsable de detectar varios tipos de error de comunicación:

- Error de superposición. Indica que un dato ha sido sobrescrito por otro más reciente antes de ser leído.
- Error de paridad. Indica que la paridad del dato recibido no concuerda con la esperada.
- Error de frame. Indica que el dato recibido no tiene un bit de *Stop* válido.
- Solicitud de break. Indica que se ha recibido un frame de pausa.
- Error de time-out (sólo para gestión por DMA usando colas FIFO). Indica que ha transcurrido un intervalo de tiempo de 3 *frames* sin recibir ningún dato, siendo el nivel de ocupación de la cola FIFO superior al esperado.

Control de flujo

La UART del S3C44BOX dispone de capacidad para control de flujo mediante las líneas nCTS [1] (*Clear To Send*) y nRTS (*Request To Send*). La gestión de estas líneas puede llevarse a cabo de manera automática por HW (*Auto Flow Control*), o bien manualmente por SW, escribiendo en los registros de control y estado del módem (UMSTATn y UMCONn).

[1] Es preciso destacar que aquí la "n" indica que la señal se activa a baja.

La idea en ambos casos consiste en que el emisor aguarde a que su línea nCTS se active antes de mandar datos, y esto sólo sucede si el receptor activa su línea nRTS (ver figura 5.3). La activación de esta línea está sujeta a disposición de espacio por parte del receptor. Si no tiene suficiente espacio para nuevos datos, éste la desactiva. Es preciso resaltar que el control de flujo automático sólo se puede emplear cuando se conectan dos UART entre sí. Para la conexión de un módem, es preciso usar el control SW, y, si se desea emplear un interfaz RS-232, es preciso usar además los puertos de E/S generales (ver figura 5.4).

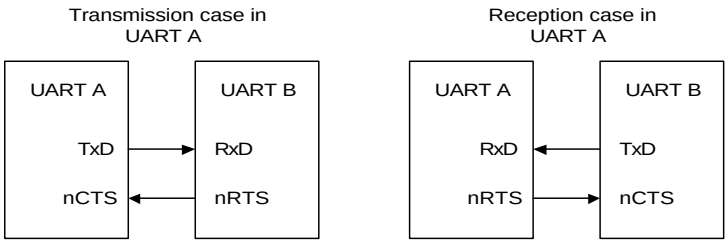

Figura 5.3: Conexión de dos UARTs empleando líneas de control de flujo.

Velocidad en baudios

La velocidad de trabajo de la UART se establece dividiendo la señal de reloj principal (MCLK) por un valor almacenado en un registro (UBRDIVn). Dicho valor se obtiene mediante la fórmula:

$$UBRDIVn = (round\_off)(MCLK/(bps \times 16)) - 1$$

El resultado de esta división debe estar comprendido entre 1 y $2^{16} - 1$.

Por ejemplo si la velocidad que se desea es de 115200 bps y la frecuencia de MCLK es de 40 MHz:

$$\begin{aligned} UBRDIVn &= (int)(40000000/(115200 \times 16) + 0{,}5) - 1 \\ &= (int)(21{,}7 + 0{,}5) - 1 \\ &= 22 - 1 = 21 \end{aligned}$$

Modo loop-back

La UART del S3C44BOX proporciona un modo de test que podríamos denominar de lazo cerrado o *loop-back*, cuyo propósito es ayudar a aislar errores en la comunicación y comprobar la corrección del SW. En este modo, el dato enviado es inmediatamente recibido por la propia UART.

La activación de este modo de test se realiza mediante un bit del registro de control de la UART (UCONn).

UART Special Registers

A continuación enunciaremos los principales registros de la UART, describiendo muy brevemente la funcionalidad de cada uno de ellos (para una descripción más detallada, consultar el Manual de Referencia del S3C44BOX):

- *UART Line Control Register (ULCONn).*
  - Activar/desactivar modo IrDA.
  - Configurar paridad (none/even/odd).
  - Determinar bits de stop (1/2).
  - Longitud del carácter (5/6/7/8).
- *UART Control Register (UCONn).*
  - Tipo de interrupción de envío/recepción (pulso/flanco).
  - Activar interrupción por timeout.
  - Activar interrupción por error (*break*, *frame*, *parity*, *overrun*).
  - Activar modo *loop-back*.
  - Mandar señal de *break*.
  - Determinar modo de transmisión para envío/recepción (desactivado, interrupción/encuesta o DMA).
- *UART FIFO Control Register (UFCONn).*
  - Habilitar FIFO.
  - Determinar el nivel de ocupación con el que se desencadenan las interrupciones de envío/recepción FIFO.
  - Borrar FIFO.
- *UART MODEM Control Register (UMCONn).*
  - Habilitar control de flujo automático (AFC).
  - Activar la señal RTS (Ready To Send) cuando el control de flujo es SW.
- *UART Tx/Rx Status Register (UTRSTATn).*
  - Determinar si el shifter de transmisión está vacío.
  - Determinar si el buffer de transmisión está vacío.
  - Determinar si el buffer de recepción tiene datos listos.
- *UART Rx Error Status Register (UERSTATn).*
  - Determinar el tipo de error que ha desencadenado una interrupción (*break*, *frame*, *parity*, *overrun*).
- *UART FIFO Status Registers (UFSTATn).*
  - Determinar si el FIFO de envío/recepción está lleno/vacío.
  - Determinar el nº de elementos presentes en la cola FIFO.
- *UART Modem Status Register (UMSTATn).*
  - Gestión de la señal CTS (*Clear To Send*) en el control de flujo SW

- *UART Transmit Holding Register (UTXHn) y UART Receive Holding Register (URXHn).*
  - Registros en los que se escriben/leen los datos.

  Nota: Cuando se produce un error de superposición (*overrun*) se debe leer URXHn o de lo contrario, al recibir el siguiente dato, se volverá a producir un nuevo error.

- *UART Baud Rate Division Register (UBRDIV).*
  - Permite determinar la frecuencia de comunicación.
    La fórmula para calcular el ratio y la velocidad en baudios se ha explicado anteriormente en esta sección.

Conexión de la UART a los puertos serie de la placa

La figura 5.4 muestra cómo se establece la conexión entre el S3C44BOX y los conectores DB9 de la placa S3CEV40. Como puede apreciarse en la figura, el conector UART0 dispone únicamente de dos líneas (RXD y TXD) y por lo tanto sólo puede emplearse para comunicaciones serie simples (sin control de flujo). El conector UART1 además de las líneas de transmisión y recepción, está conectado a 6 pines de E/S generales lo que permite que, con una gestión adecuada de los mismos, se pueda conectar a un modem RS-232.

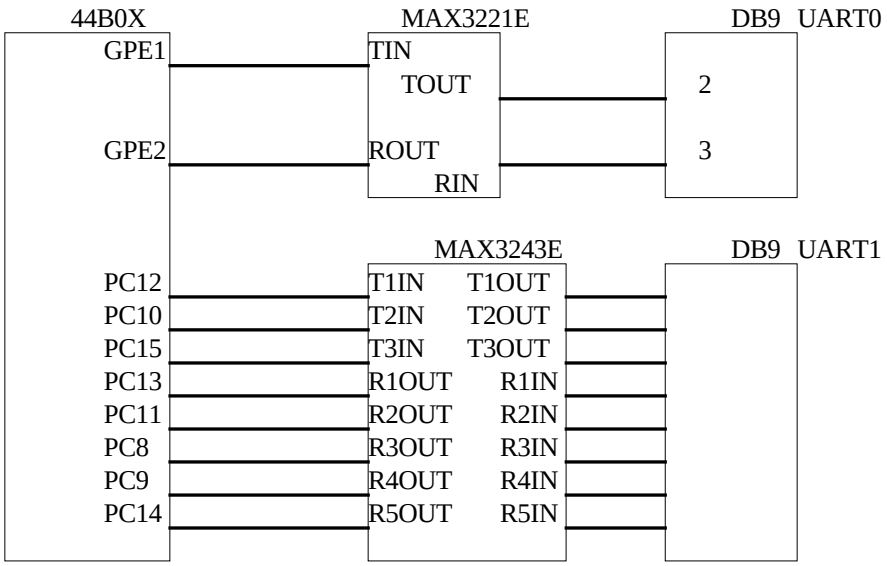

Figura 5.4: Conexión de la UART a los puertos serie de la placa (DB9).

5.1.3. Desarrollo de la práctica

1. Conectar el puerto UART0 al puerto serie del PC.

   a) Abrir el programa HyperTerminal (Accesorios –>Comunicaciones).

   b) Dar nombre a la conexión (ej. Embest).
   c) Pulsar en "Conectar usando" (ver figura 5.5) y seleccionar el puerto COM1.
   d) Introducir la siguiente configuración: 115200 bps, 8 bits, sin paridad, 1 bit de parada, sin control de flujo (ver figura 5.6).
   e) Encender la placa y comprobar que se muestra el mismo mensaje que en el LCD (ver figura 5.7).

2. Abrir el entorno de trabajo (`P5-ARM.ews`).
3. Comprender la estructura de ficheros y la configuración del proyecto.
4. Completar el código.
5. Cargar el código en la placa mediante el emulador UNetICE.
6. Ejecutar y comprobar que en el terminal se muestran las teclas que se van pulsando y que al pulsar ENTER se vuelve a mostrar la cadena completa.

```
LEC-ARM-P5 >Ejemplo de cadena de texto
Ejemplo de cadena de texto
LEC-ARM-P5 >
```

7. De lo contrario, depurar.

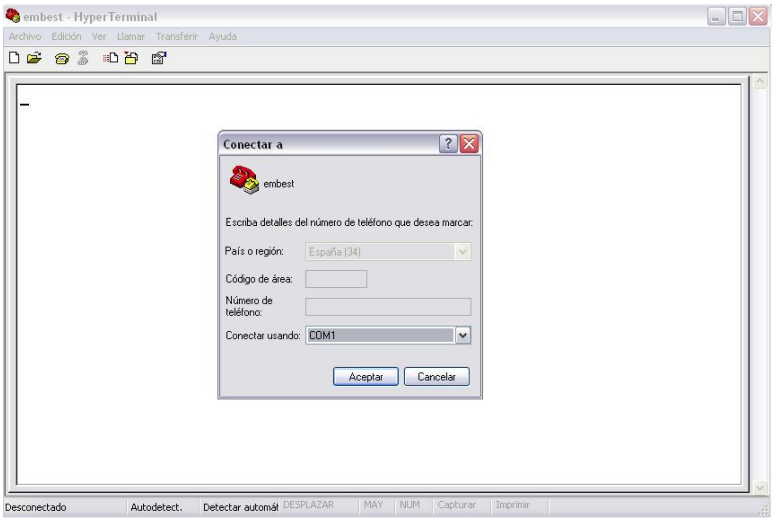

Figura 5.5: Ventana del programa HyperTerminal "Conectar a" .

El ejercicio consiste en completar el código de las funciones del fichero `uart.c`. Para alguna de ellas es preciso crear dos versiones: una que realiza la gestión de la UART mediante encuesta y otra que realiza la gestión mediante interrupciones. Por defecto, se emplea la versión de encuesta (*polling*), si se quiere usar la otra versión es necesario descomentar la siguiente línea y volver a compilar el proyecto:

```
//#define  UART_INT
```

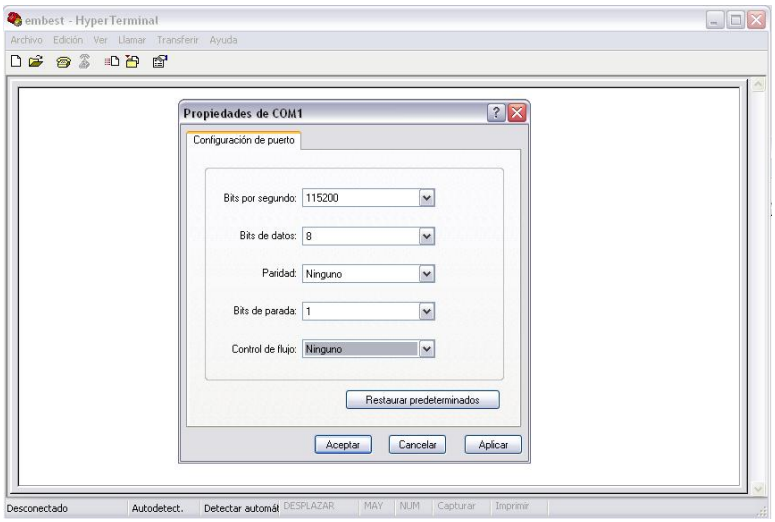

Figura 5.6: Ventana del programa HyperTerminal "Propiedades de COM1" .

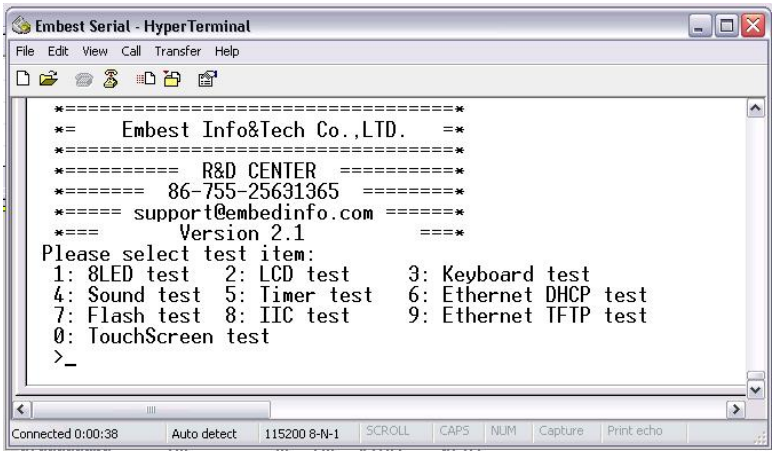

Figura 5.7: Mensaje por defecto de la placa S3CEV40 enviado por el puerto serie UART0.

Diagrama de flujo

La figura 5.8 muestra el diagrama de flujo del programa principal. Tal y como se muestra en la figura los principales pasos a seguir son:

1. Inicialización de interrupciones, puertos y UART. Se realizará con la rutina `sys_init()` que ya vimos en las prácticas 3 y 4. Esta rutina a su vez llama a la función `Uart_Init()` que es la responsable de configurar la UART (ambos canales).
2. Configuración adicional de la UART mediante la rutina `Uart_Config()`. Esta rutina sólo surte efecto en la versión de interrupciones.
3. Repetir:
   a) Lectura de un carácter del puerto serie mediante la función `Uart_Getch()`.
   b) Envío del carácter leído mediante la función `Uart_SendByte()`.

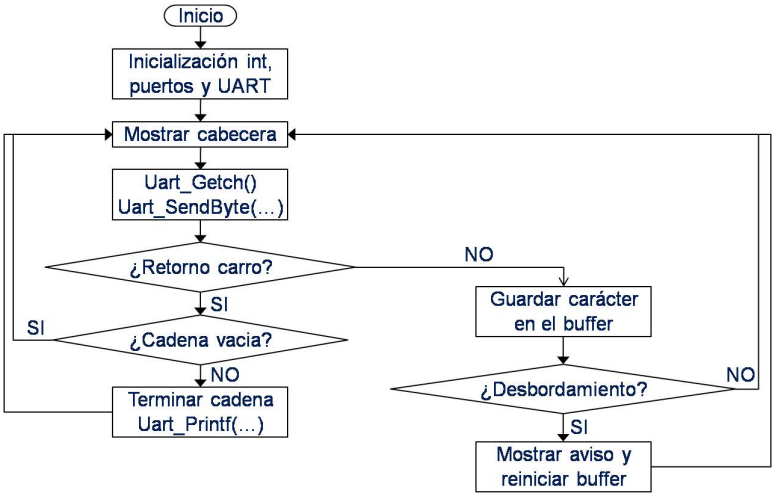

Figura 5.8: Diagrama de flujo del programa principal.

c) Cuando el carácter es un retorno de carro:

1) Terminación de la cadena de caracteres.
2) Envío de un salto de línea.

Organización del código de la práctica

La práctica consta de los siguientes ficheros:

1. `44b.h`, `44blib.h`, `def.h`, `option.h`, `44binit.s`, `44blib.c`, `ev40boot.cs`, `ram_ice.ld`
   Sus funcionalidades fueron descritas en la práctica 3.

2. `uart.c` y `uart.h`
   Estos archivos contendrán todas las funciones que hay que desarrollar para la gestión de la UART:

   - `void Uart_Config()`. Configuración de la UART0 para su uso mediante interrupciones.
   - `void Uart_TxEmpty()`. Función que espera a que el shifter de transmisión de la UART0 se vacíe.
   - `char Uart_Getch()`. Recepción de un carácter por la UART0.
   - `void Uart_SendByte()`. Envío de un carácter por la UART0.
   - `void Uart_SendString()`. Envío de una cadena de caracteres por la UART0.
   - `void Uart_Printf()`. Generación de una cadena de caracteres a partir de una especificación de formato y envío mediante `void Uart_SendString()`.

3. `main.c`.
   Programa principal.

```
int Main(void)
{
    char *pt_str = str;

    sys_init(); // inicializacion de la placa, interrupciones, puertos y UART

    Uart_Config();            // configurar UART
    Uart_Printf(str_send);    // mostrar cabecera
    Delay(200);       // por si el terminal es lento, esperar

    while(1)
    {
        *pt_str = Uart_Getch();     // leer carácter
        Uart_SendByte(*pt_str);     // enviar carácter

        if (*pt_str == CR_char)         // retorno de carro?
        {
           if (pt_str != str) { // si cadena no vacia, mostrar
              *(++pt_str) = '\0'; // terminar cadena antes
              Uart_Printf("\n");
              Uart_Printf(str);
           }
           Uart_Printf(str_send);      // preparar siguiente
           pt_str = str;
        }
        else if (++pt_str == str + 255) // desbordamiento?
        {
           Uart_Printf(str_error);  // avisar del desbordamiento
           pt_str = str;
        }
    }
}
```

5.2. Práctica 5 - Apartado B. Interfaz Bus IIC y EEPROM

5.2.1. Objetivos de la práctica

- Familiarizarse con la comunicación serie síncrona.
- Manejo del interfaz de bus IIC incluido en el S3C44B0X.
- Manejo de la EEPROM AT2404 de la placa S3CEV40.

5.2.2. Conceptos teóricos

Interfaz IIC del S3C44BOX

Introducción al bus IIC

El IIC (*Inter-Integrated Circuit*) es un bus basado en comunicaciones serie síncronas. Es muy utilizado en sistemas empotrados ya que tiene una buena velocidad de transferencia (100 Kbps en modo estándar y hasta 400 Kbps en modo mejorado) y un número muy reducido de líneas:

- SCL: reloj (bidireccional).
- SDA: datos (bidireccional).
- GND: tierra.

Dispone además de un mecanismo de arbitraje que permite emplearlo en configuraciones *multi-master*. Sin embargo, lo más habitual es que se emplee para comunicar un único *master* (microcontrolador) con varios *slaves* (periféricos).

Comunicación de datos mediante IIC

Debido a su naturaleza síncrona, la comunicación de datos mediante IIC tan sólo necesita delimitar el comienzo y el final de cada bloque de datos. Tanto la sincronización de bit como la de byte se llevan a cabo mediante la propia señal de reloj que genera el dispositivo que actúa como *master*. La transmisión de un bloque comienza cuando se produce una transición $0 \rightarrow 1$ en SDA mientras la línea SCL permanece a 1. Esto se conoce como *Condición de Start* o simplemente *Start* (ver figura 5.9). El bloque termina cuando se produce la transición inversa ($1 \rightarrow 0$) con el reloj a 1: *Condición de Stop*. En ambos casos el responsable de generar dichas transiciones en la línea SDA es el dispositivo *master* (en nuestro caso el S3C44BOX).

El primer byte que se envía tras el delimitador de comienzo (*Start*) contiene la dirección del *slave* con el que se quiere comunicar (7 bits) y un bit que indica el tipo de operación que se desea realizar (lectura/escritura). A continuación se leen/escriben el resto de bytes del bloque (>1 byte). El envío de cada uno de estos bytes, así como del byte inicial, suele ir acompañado de un bit de confirmación (*acknowledgement* o simplemente *Ack*) por parte del receptor. Esto además, permite llevar a cabo un cierto control de flujo ya que la comunicación queda "congelada" mientras se recibe y procesa esta confirmación (ver figura

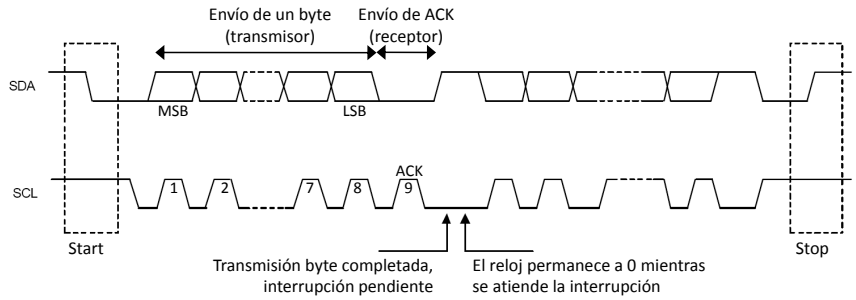

Figura 5.9: Diagrama temporal de una comunicación serie mediante bus IIC.

5.10). La figura 5.11 muestra el formato de bloque para un caso sencillo en el que sólo se lee/escribe un byte.

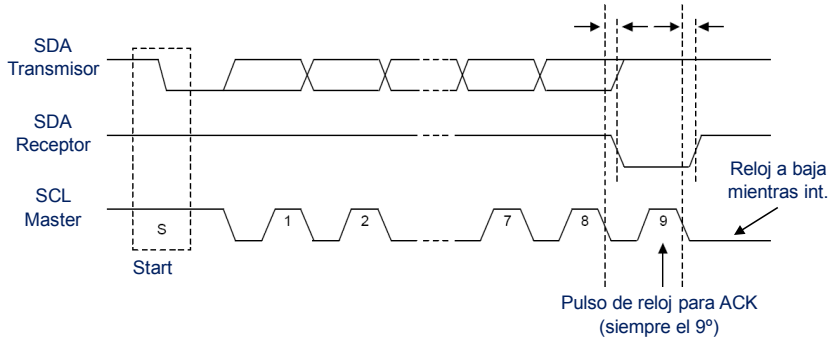

Figura 5.10: Señal de confirmación (*Ack*) del bus IIC.

Estructura del interfaz de bus IIC y modos de transmisión

El interfaz de bus IIC del S3C44BOX consta de los siguientes elementos (ver figura 5.12): lógica de control, registros y divisor de frecuencia. Las líneas SDA y SCL son ambas bidireccionales; el sentido depende del modo de transmisión que se esté empleando y de la fase concreta en la que se encuentre el envío del bloque. Aunque habitualmente sólo lo emplearemos como *master*, el interfaz soporta los siguientes modos:

- Master-transmisión.
- Master-recepción.
- Slave-transmisión.
- Slave-recepción.

Registros del Interfaz IIC

La comunicación con el interfaz IIC del S3C44BOX se lleva a cabo por medio de 4 registros, cuya funcionalidad se indica de manera muy resumida a continuación (para una información más detallada consultar el capítulo 16 del manual de referencia del S3C44BOX):

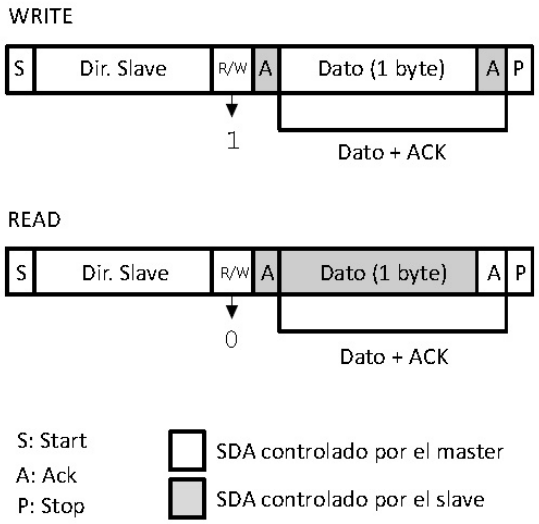

Figura 5.11: Formato de un bloque IIC con un solo byte de datos.

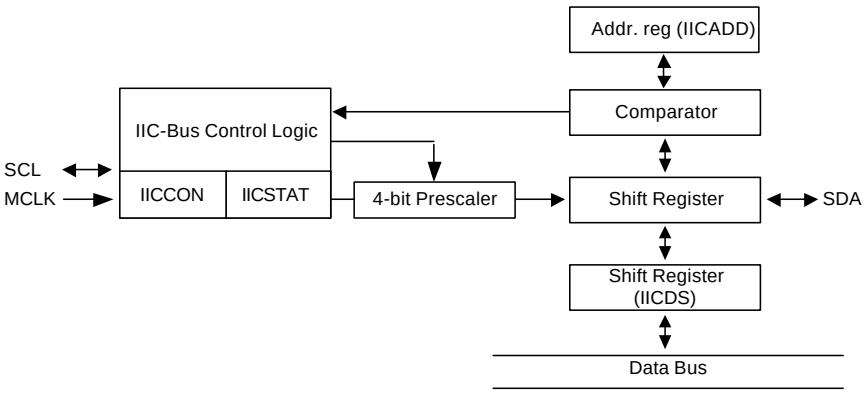

Figura 5.12: Estructura del interfaz de bus IIC del S3C44BOX.

- *Multi-Master IIC-BUS Control Register (IICCON)*
  - Habilitar/deshabilitar la generación de ACK.
  - Establecer frecuencia del reloj de transmisión.
  - Establecer frecuencia de transmisión.
  - Habilitar/deshabilitar interrupción.
  - Consultar/borrar interrupción pendiente.
- *Multi-Master IIC-BUS Control/Status Register (IICSTAT)*
  - Configurar modo operación (transmisión/recepción).
  - Mandar señal de START/STOP.
  - Habilitar salida serie.
  - Consultar estado arbitraje, dirección y ACK.
- *Multi-Master IIC-BUS Address Register (IICADD)*
  - Establecer la dirección del interfaz (cuando actúa como *slave*).
- *Multi-Master IIC-BUS Transmit/Receive Data Shift Register (IICDS)*
  - Leer/Escribir un byte de datos.

Manejo del Interfaz IIC

Antes de escribir o leer un bloque de datos es necesario realizar las siguientes acciones previas:

- Escribir la dirección propia en IICADD (por defecto, cuando se libera el bus, el interfaz actúa como *slave*).
- Configurar el interfaz mediante IICCON:
  - Habilitar interrupciones.
  - Definir periodo del reloj SCL.
- Habilitar TX/RX mediante ICCSTAT.

Una vez realizado este proceso previo, el envío y la recepción de un bloque de datos (como *master*) se realizan siguiendo los diagramas de flujo mostrados en las figuras 5.13 y 5.14.

Durante el envío, tras la transmisión un byte y el correspondiente ciclo de *Ack*, se libera la señal de reloj (se mantiene a 0) y de este modo se congela la comunicación (ver figura 5.10). Al recibir la confirmación (*Ack*) por parte del receptor, se genera una interrupción (IIC) que debe ser atendida por el microcontrolador. Hasta que esta no se procesa esta interrupción y se borra el bit de interrupción pendiente, no se reanuda la comunicación.

El proceso es similar para la recepción de datos, solo que en este caso la interrupción se produce como resultado de recibir un nuevo dato en lugar de un *Ack*.

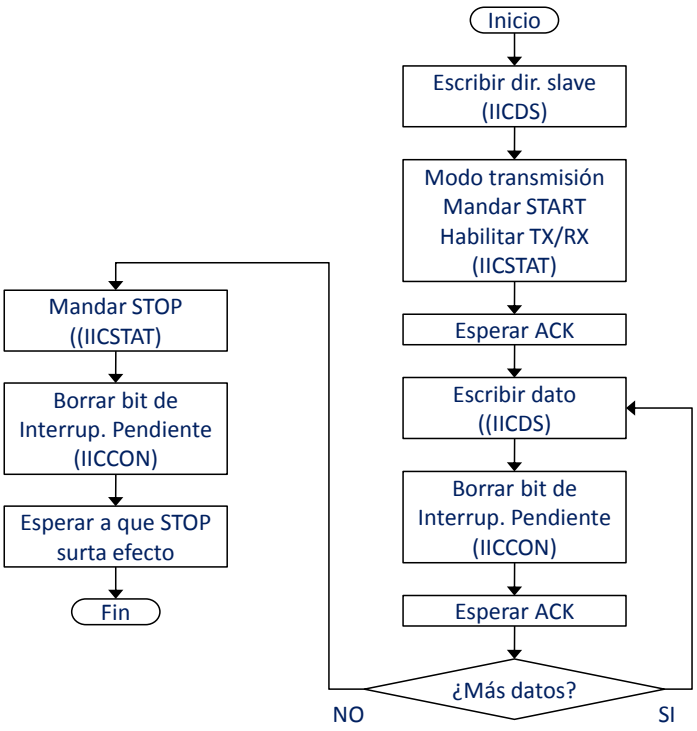

Figura 5.13: Diagrama de flujo para la transmisión de un bloque mediante el interfaz IIC.

EEPROM AT2404

Un tipo de dispositivo que habitualmente se conecta al microcontrolador mediante el bus IIC son las memorias EEPROM. La placa S3CEV40 de Embest dispone de una memoria de este tipo, en concreto una ATMEL AT2404 de 4 Kbits.

Pines de la EEPROM AT2404

La memoria EEPROM AT2404 consta de 8 pines, tal y como refleja la figura 5.15:

- Reloj (SCL). Sigue el estándar ISO/IEC7816, utiliza el flanco de subida para guardar los datos en la EEPROM y el de bajada para volcarlos hacia afuera.
- Entrada/Salida de datos (SDA). A diferencia del reloj, se trata de un pin bidireccional.
- Dirección (A2, A1, A0). En las EEPROM de la familia AT24xx estos pines permiten definir los 3 bits menos significativos de la dirección de dispositivo (¡no confundir con la dirección de palabra!). En el modelo AT2404, en concreto, A0 no se usa y su valor interno se fija a 0. Los 4 bits más significativos de la dirección de dispositivo en todos los modelos de la familia AT24xx tienen un valor fijo de 1010.
- Protección de escritura (WP). Cuando está conectado a tierra (GND) se puede leer y escribir en la EEPROM. Si por el contrario se conecta a VCC, la escritura en el

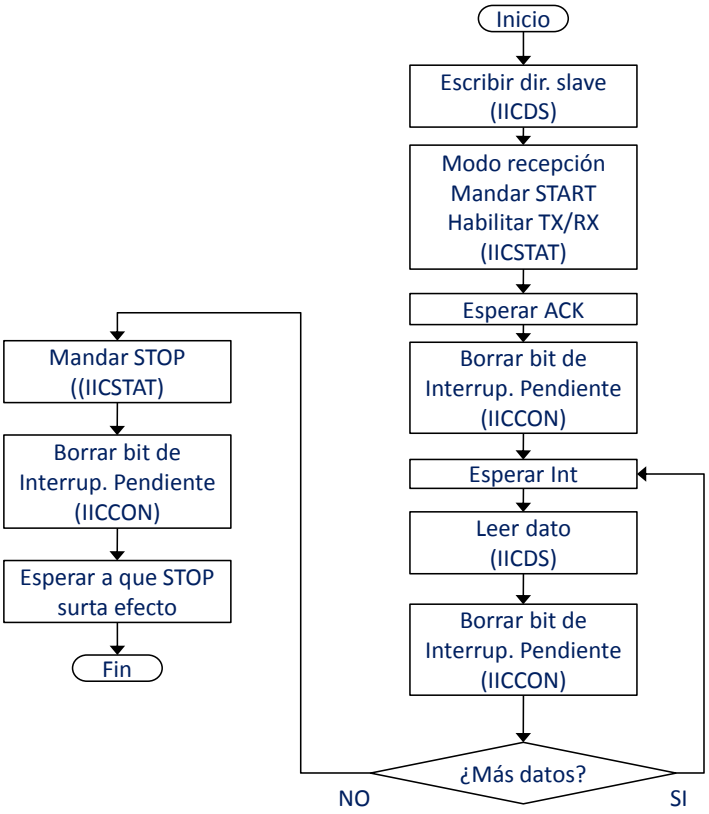

Figura 5.14: Diagrama de flujo para la recepción de un bloque mediante el interfaz IIC.

dispositivo queda inhabilitada.

- Tierra y alimentación (VCC/GND).

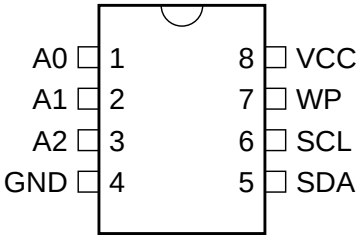

Figura 5.15: Memoria EEPROM AT2404.

Estructura y funcionamiento de la EEPROM AT2404

La estructura interna del chip AT2404 consta esencialmente de los siguientes elementos (ver figura 5.16: buffer de escritura, registro de dirección de palabra, comparador de dirección de dispositivo, lógica de control y array EEPROM).

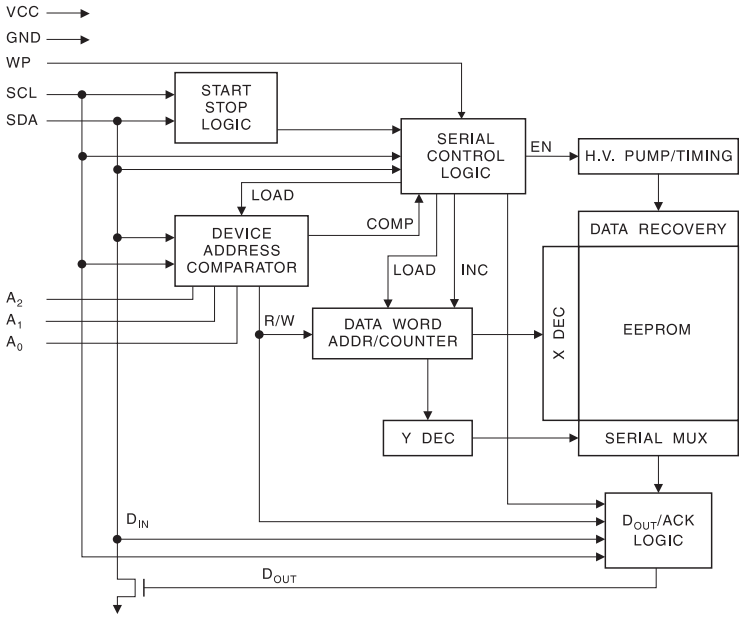

Figura 5.16: Estructura interna de la EEPROM AT2404.

La figura 5.17 muestra el diagrama temporal de una escritura en la EEPROM mediante el bus IIC. En primer lugar, se envía la dirección de la palabra (1 byte), que se guarda internamente en el registro de dirección de palabra del AT2404. La figura 5.20 muestra la correspondencia entre los bits enviados por el bus IIC y la dirección de palabra de la EEPROM. A continuación, se manda el byte que se desea escribir, y éste se guarda en el buffer interno para su posterior escritura en el array. Una vez realizada la escritura en el buffer, el bus IIC se puede liberar (mediante envío de *Stop*). No obstante, como la escritura

en el array es mucho más lenta, es necesario esperar un cierto intervalo de tiempo (5 ms) antes de poder realizar una nueva operación sobre la EEPROM (ver figura 5.19).

La lectura de la EEPROM se divide en dos fases, primero se envía la dirección (igual que para la escritura) y después se lee el byte (ver figura 5.18). La lectura del byte implica un cambio de sentido en la comunicación del bus y ello requiere volver a mandar *Start*, la dirección de dispositivo y el bit de operación (*Read*). Otra particularidad de la lectura, es que tras la recepción del dato, el microcontrolador no debe mandar confirmación (*Ack*).

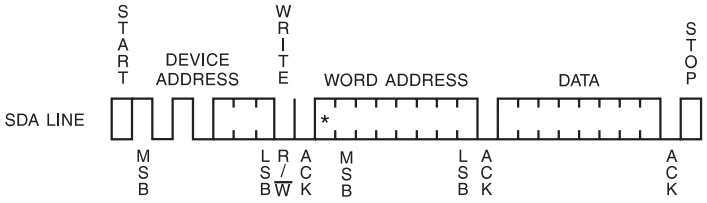

Figura 5.17: Diagrama temporal de una escritura en la EEPROM AT2404.

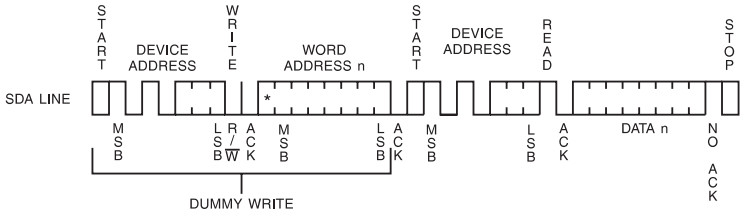

Figura 5.18: Diagrama temporal de una lectura en la EEPROM AT2404.

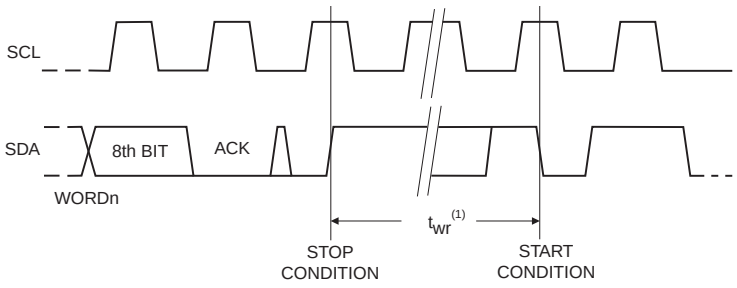

Figura 5.19: Retardo de escritura de la EEPROM AT2404 (Twr=5ms).

Conexión de la EEPROM al S3C44BOX

La memoria EEPROM AT2404 se conecta a la placa S3CEV40 de Embest tal y como indica la figura 5.21:

- Los pines A2, A1 y A0 se conectan a tierra (GND), lo que hace que la dirección de dispositivo sea 1010000.
- La protección de escritura se encuentra deshabilitada.
- Los pines SCL y SDA están conectados a los pines correspondientes del S3C44BOX.

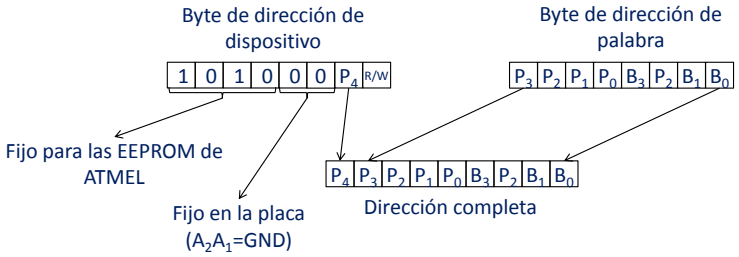

Figura 5.20: Formato de dirección de palabra de la EEPROM AT2404.

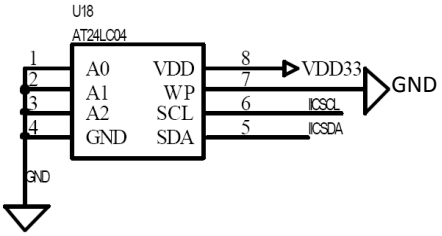

Figura 5.21: Conexión de la EEPROM AT2404 a la placa S3CEV40.

5.2.3. Desarrollo de la práctica

1. Conectar el puerto UART0 al puerto serie del PC como en el apartado anterior.
2. Abrir el entorno de trabajo (`P5-ARM.ews`).
3. Comprender la estructura de ficheros y la configuración del proyecto IIC.
4. Completar el código.
5. Cargar el código en la placa mediante el emulador UNetICE.
6. Ejecutar el programa
7. Seleccionar la opción de escritura (w) y teclear una cadena de caracteres.
8. Seleccionar la opción de lectura (r) y comprobar que se muestra la cadena anterior. *Nota: es posible apagar la placa entre la escritura y la lectura, pero es necesario cargar de nuevo el código.*
9. De lo contrario, depurar.

El ejercicio consiste en completar el código de las funciones del fichero `iic.c`. Para la comunicación por el puerto serie es necesario emplear el código desarrollado en el apartado anterior (versión de interrupciones únicamente).

Diagrama de flujo

La figura 5.22 muestra el diagrama de flujo del programa principal. Tal y como se muestra en la figura, los principales pasos a seguir son:

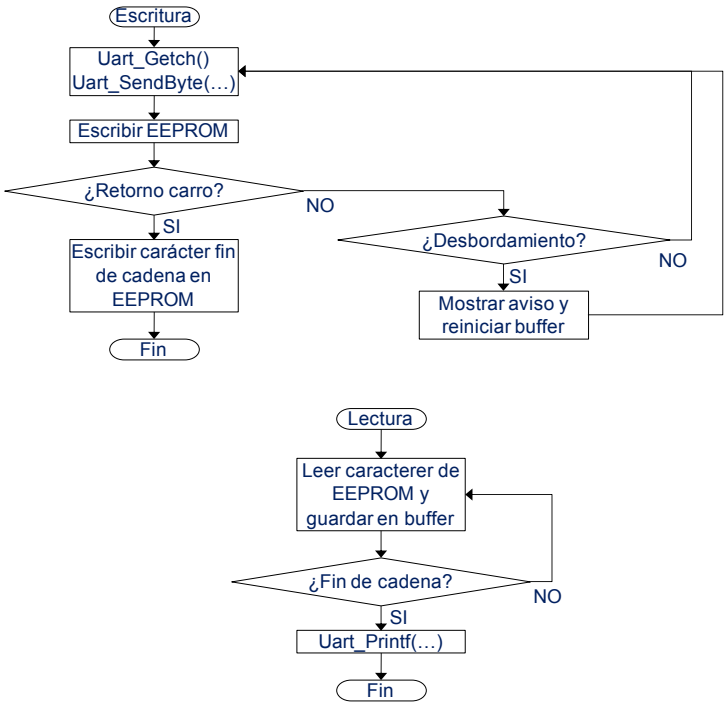

Figura 5.22: Diagrama de flujo del programa principal.

1. Inicialización de interrupciones, puertos y UART. Se realizará con la rutina `sys_init()` que ya vimos en las prácticas 3 y 4. Esta rutina se encarga de llamar a la función `Uart_Init()`, que es la responsable de configurar la UART (ambos canales).
2. Configuración adicional de la UART y del interfaz de bus IIC mediante las rutinas `Uart_Config()` y `IIC_Config()`.
3. Seleccionar una opción mediante el terminal serie (r: lectura; w: escritura).
4. Escritura:
   a) Lectura de un carácter del puerto serie mediante la función `Uart_Getch()`.
   b) Envío del carácter leído mediante la función `Uart_SendByte()`.
   c) Escritura del carácter leído en la EEPROM mediante la función `Wr24C040()` y actualización de la dirección de escritura.
   d) Cuando el carácter es un retorno de carro:
      1) Terminación de la cadena de caracteres.
      2) Salir del modo de escritura.
5. Lectura:
   a) Leer una cadena de la EEPROM (carácter a carácter) mediante la función `Rd24C040()` y guardarla en un buffer en memoria.
   b) Mostrar la cadena por el terminal serie mediante la función `Uart_Printf()`.

Organización del código de la práctica

La práctica consta de los siguientes ficheros:

1. `44b.h`, `44blib.h`, `def.h`, `option.h`, `44binit.s`, `44blib.c`, `ev40boot.cs`, `ram_ice.ld`
   Sus funcionalidades fueron descritas en la práctica 3.

2. `uart.c` y `uart.h`
   Estos archivos contienen las funciones de gestión de la UART desarrolladas en el apartado anterior.

3. `iic.c` y `iic.h`
   Estos archivos contendrán todas las funciones que hay que desarrollar en esta apartado de la práctica para la gestión del interfaz de bus IIC:

   - `void IIC_Config()`: Configuración del interfaz IIC.
   - `void Wr24C040()`: Escritura de un byte en la EEPROM AT2404.
   - `void Wr24C040()`: Lectura de un byte en la EEPROM AT2404.

4. `main.c`.
   Programa principal.