MEMORIA LCSE

2020

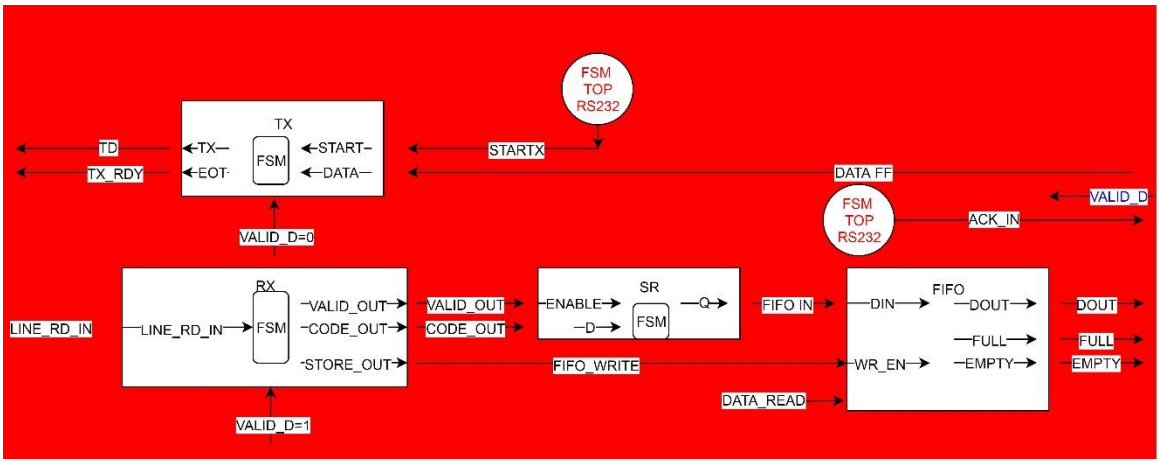
Mario Ruiz Ramírez de Arellano

Contenido

1. BLOQUE RS232

El bloque RS232 será el encargado recibir y entregar los datos con el entorno dividiendo en 2 bloques el conjunto:

- Bloque recepción: bloque RX, el bloque Shift Register y el bloque FIFO.
- Bloque transmisión: Compuesto por el bloque TX del RS232.



- Ilustración 1: Bloque RS232

Bloque recepción:

Los datos llegan al bloque RX de forma secuencial, bit a bit. Este bloque será el encargado de tomar la línea de bits y separar los datos de recepción de 8 bits en 8 bits.

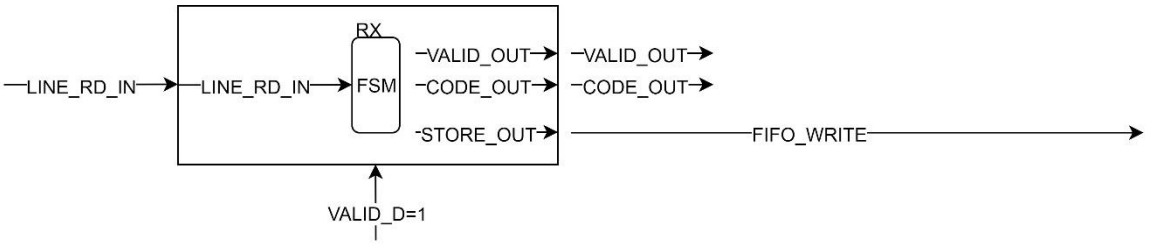


Ilustración 2: Bloque RX RS232

Por otro lado, el bloque shift register será el encargado de transformar la línea de 8 bits secuenciales de entrada, en un valor de 8 bits.

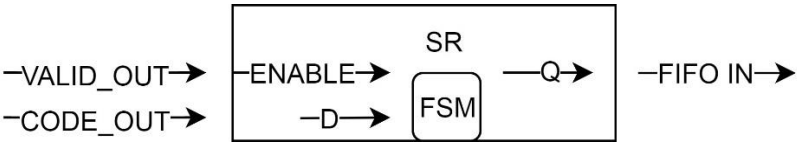


Ilustración 3: Bloque Shift Register RS232

Por último, el dato de 8 bits se enviará al bloque FIFO que se encargará de estar en comunicación con el bloque DMA para el envío de datos

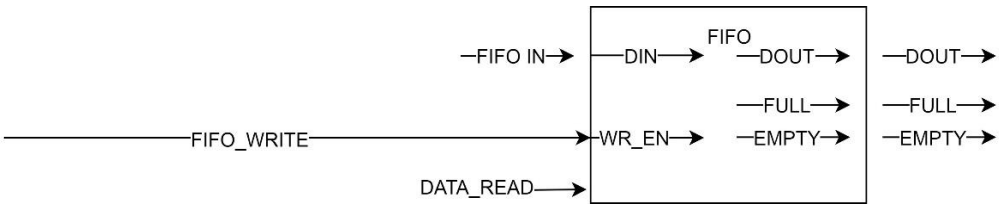


Ilustración 4: Bloque FIFO RS232

Bloque transmisión:

Este bloque se encargará de recibir un valor de entrada de 8 bits del bloque DMA. Transformándolo en una cadena de pulsos en la línea TD, por el que los 8 bits que se transmitirán.

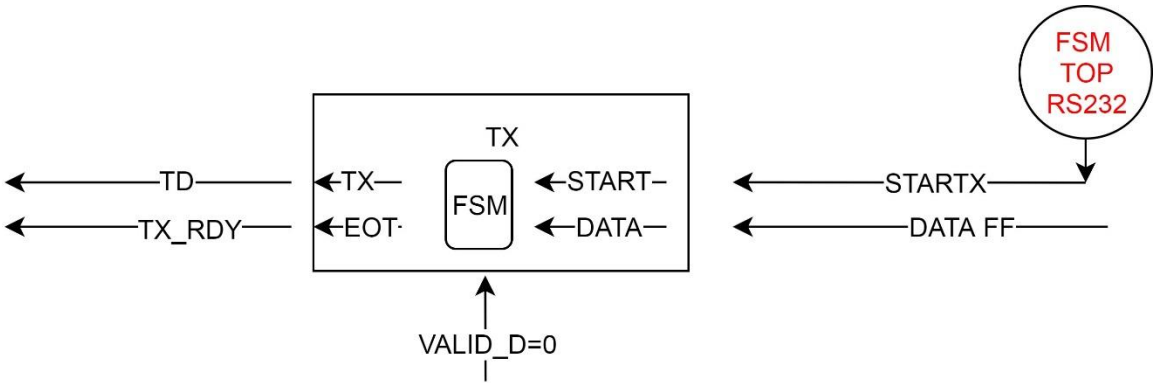


Ilustración 5: Bloque TX RS232

1.1. Funcionamiento RS232

El bloque RS232 está compuesto por el transmisor y el receptor del sistema. las decisiones principales que se han tenido en cuanta para el diseño de estos bloques son-,

- El dispositivo funciona a 115200 bps, a una frecuencia de 20 MHz. Por lo tanto, para implementar ambos bloques y poder controlar la entrada y salida de los bits, se ha calculado que cada 174 pulsos de reloj corresponden a un bit.
- En el transmisor y en el receptor se ha creado un proceso que nos permitirá,llevar la cuenta de los pulsos de reloj, almacenando el número de pulsos en la variable "s_pulse_width". Alcanzando el valor de 174, se detecta que se ha leído un bit.

1.2. Transmisor

- Se ha implementado una máquina de estados formada por 4 estados: Idle, Start_Bit, Send_Data, StopBit.
    - Idle Corresponde con el estado inicial. Cuando el sistema detecta que Start_Bit ha sido activado, el sistema pasa al estado Start_Bit.
    - Start_Bit Realizará una cuenta para detectar el primer bit de envío. Una vez finalizado la cuenta, el sistema pasa al estado Send_Data.
    - Send_Data Este es el estado encargado de realizar el envío. Con ello se contara cada uno de los bits que se quieren transmitir, en total 8 bits. Por lo tanto, en cada ciclo de cuenta se irá asignando, bit a bit, a la salida el valor a transmitir. Una vez el sistema detecta que se han transmitido los 8 bits correctamente, pasamos al estado final. el sistema pasa al Stop_Bit.
    - Stop_Bit Nos indica el fin de la transmisión.

El transmisor cuenta con una salida "EOT". Esta salida estará activa a nivel alto en el estado de Idle, indicando que el transmisor no está transmitiendo. Esta salida relaciona una señal del sistema DMA. Cuando se produce el fin de la transmisión, el DMA recibe la señal, la cual le indica que puede enviarle al RS232 los datos que el sistema quiera transmitir.

1.3. Receptor

Para la implementación del receptor se ha implementado una máquina de estados formada por 4 estados como se describe en el enunciando del proyecto: Idle, Start_Bit, RcvData, StopBit.

- Idle: La entrada del receptor es la entrada del sistema, por medio del que nos llega la información del exterior. Está información llegará bit a bit. Cuando el sistema está en este estado, detectará que la entrada está siempre a nivel alto. En el momento en que el sistema de recepción detecte un cero a la entrada, significa que hay un dato que ha de procesarse.

- Start_Bit Tras realizar un conteo se pasará de estado.

- RcvData Es en este estado en el que se gestiona la recepción de cada uno de los datos recibidos. Estos datos pasarán bit a bit por el receptor y se verán a su salida, la cual está conectada con un Shift Register, el cual ordenara la información que nos llega en serie, en un vector de datos de 8 bits (cambio serie - paralelo). Para que el envío de datos entre el receptor y el shift register se implementará un proceso que activará el shift register (enable).
  Una vez se han recibido los 8 bits, y se han validado para su conversión serie – paralelo, se produce el cambio de estado.

- Stop_Bit Este estado representa el fin de la recepción de datos. En este instante la salida Store_Out se pondrá a nivel alto, indicándole a la FIFO que la recepción de datos ha terminado durante un ciclo de reloj, y permite que el shift register pueda almacenar los datos en la memoria FIFO.

2. BLOQUE DMA:

En el proyecto, el bloque DMA es el encargado de proporcionar la comunicación entre los bloques de comunicación RS232 y la memoria RAM.

Éste será el encargado de, mediante la utilización de un bus de datos, escribir en unas posiciones de memoria. Además, otra de las funciones de este bloque será la de leer de la memoria RAM, y enviarle la información almacenada al bloque Transmisor RS232.

El bus de datos es un recurso que comparte el bloque DMA y el microprocesador. Éste último será el encargado de permitir o bloquear al DMA, acceder al bus de datos para comunicarse con la memoria RAM.

Componentes bloque DMA:

Entradas	Salidas	Bidireccionales
Reset	Data_Read	Databus
CLK	Valid_D	
RCVD Data	TX_Data	
Rx_Full	Address	
Rx_Empty	Write_en	
ACK_Out	OE	
TX_RDY	DMA_RQ	
DMA_ACK	READY	
Send_comm		

Tabla 1 : Perifericos DMA

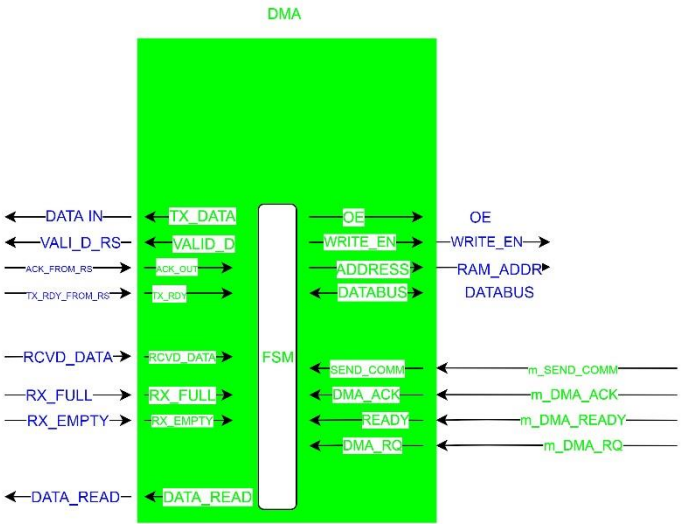


Ilustración 6: Bloque DMA

2.1. Funcionamiento DMA

El bloque DMA, se compone de dos etapas bien diferenciadas:

2.1.1. Escritura en RAM:

1. Cuando la señal RX_EMPTY pasa a nivel bajo, es una señal que indica que el bloque RS232 tiene un dato para enviar a la memoria RAM, activando el proceso de CONTROL RX
2. Los datos salen del bloque RS232 y entran por la línea RCVD_Data al bloque DMA
3. Al comenzar la recepción se solicitara el bus al microprocesdor y para hacer la transmisión nada mas recibir el dato. Cuando se permita el acceso al bus de datos, mediante la señal DMA_ACK, el bloque DMA activará señal Write_en, indicando al bloque RAM que se va a proceder a escribir.
4. Se vuelcan los datos almacenados del bloque RS232 en las direcciones indicadas de la memoria. Una vez finalizada la escritura de esos tres datos, se escribirá un FF en la dirección de memoria 0x03, a modo de FLAG para indicar que se han escrito todos los datos.
5. Por último, el bloque DMA desactivará la petición del bus de datos al procesador.

2.1.2. Lectura de la RAM:

1. Para comenzar, el DMA solicitará el Bus de datos al procesador.
2. Una vez el procesador conceda el bus de datos, se activará la señal de OE indicando a la RAM que va a ser leída.
3. El bloque DMA recorrerá las posiciones y tomará los datos, para posteriormente transmitir al boque RS232.
4. Una vez finalizada la transmisión, se devuelve automáticamente el control del bus de datos al procesador principal.
5. Por último, el bloque DMA enviará al bloque TX del RS232 la información, para que posteriormente la saque bit a bit por la línea TD

2.2. Estados del bloque DMA

- Escritura en RAM: estará involucrados los estados de Idle, CONTROL RX y la rama en función a la iteración que parte del control del contador.

- Lectura en RAM: estará involucrados los estados de Idle, CONTROL TX y la rama secuencial.

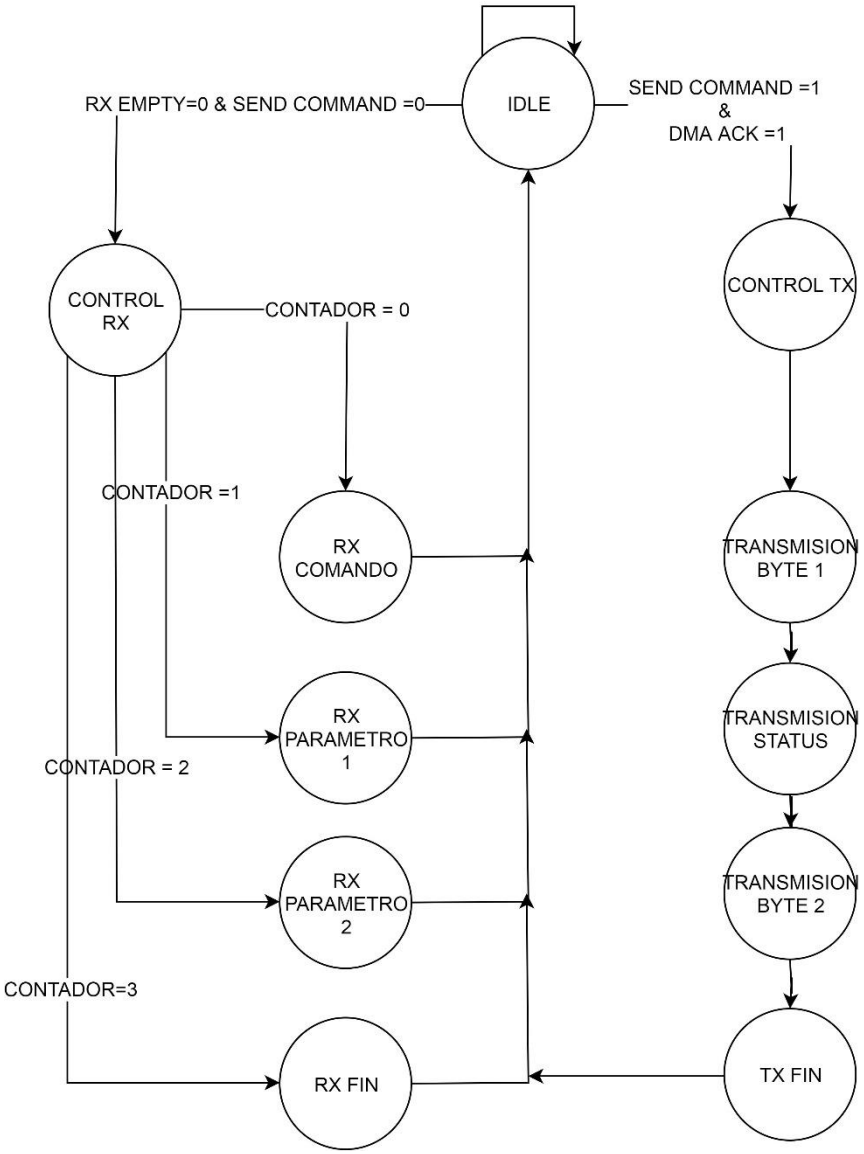


Ilustración 7: Diagrama DMA

3. MEMORIA RAM

En este prototipo del microcontrolador utilizaremos una entidad RAM que implementará una memoria RAM de propósito general y también una serie de posiciones de memoria (registros de E/S) para acceso a los diferentes periféricos que el sistema completo debe controlar, así como las posiciones de memoria empleadas por el controlador de acceso directo a memoria. La entidad contará con una serie de líneas de salida para control de los algunos de los periféricos.

Para su correcta implementación, se ha dividido la memoria RAM en dos entidades: una de propósito general, y otro de propósito específico. Con esta separación se consigue que la herramienta de síntesis haga un mejor uso del hardware, debido a que una de las memorias (la específica) lleva un reset del sistema. El uso del reset hace que el sintetizador haga uso de flip- flops discretos hasta completar la memoria.

Componentes bloque RAM:

Entradas	**Salidas**	**Bidireccionales**
Reset	Switches	Databus
CLK	Temp_L	
Write_en	Temp_H	
oe		
address		

Tabla 2 : Perifericos RAM

3.1. Funcionamiento RAM

3.1.1. Memoria específica

Corresponde a las posiciones de memoria que van desde la posición 0x00 hasta la posición 0x3F. Estas posiciones, al activar el reset del sistema, deberán poner el valor de su contenido a un valor determinado. Para ello, cada vez que se hace reset, el valor de la posiciones, mediante el bucle, se pone a cero. Todas, menos las posiciones de memorias relativas a la temperatura (inicializada en 20º) y a las posiciones reservada del DMA.

En función de estado de las entradas write_en u oe, la memoria se encargará de almacenar los datos que nos llegan por el bus bidireccional en la posición de memoria que indique el sistema, o de poner en el bus de direcciones un valor deseado almacenado en una dirección concreta si el sistema quiere leer de la memoria.

3.1.2. Memoria general

Corresponde a las posiciones de memoria que van desde la posición 0x40 hasta la posición 0xFF. Esta memoria se ha diseñado de forma idéntica a la anterior en relación con la escritura y la lectura. La diferencia principal es que no incluye un reseteo de los valores contenidos en las diferentes direcciones que componen esta parte de la memoria.

4. Funcionamiento general

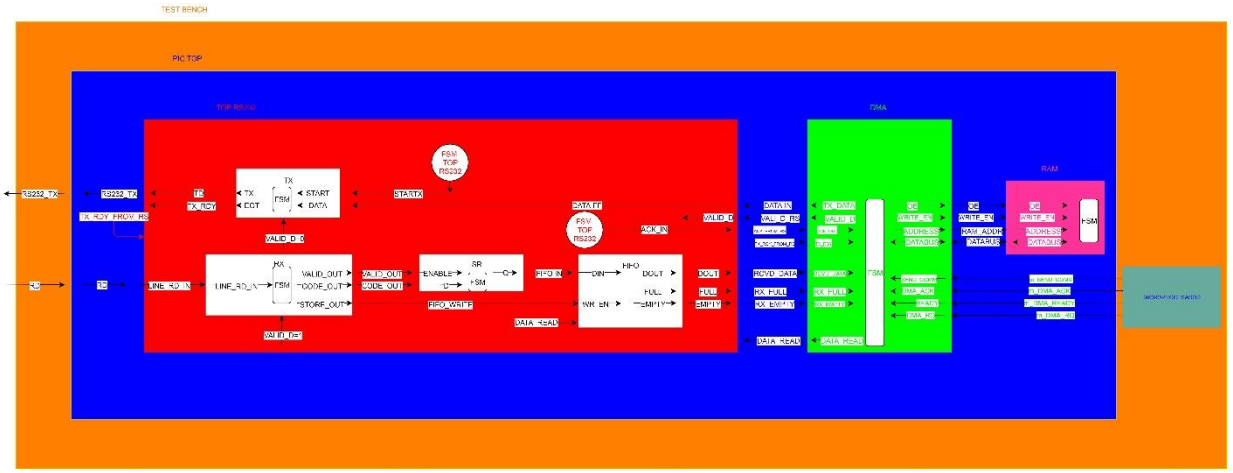


Ilustración 8: Plano General de funcionamiento

Para comprobar que el funcionamiento es correcto, hemos procedido a realizar una simulación que se muestra a continuación. En ella, podemos observar como el bloque inicia una fase de escritura desde el RS232 a la memoria RAM de los valores 49, 34, 31 y ff para la primera secuencia, 77,55,BB para la segunda secuencia y 33,AB, 43

Posteriormente se leerán unos valores almacenados en las posiciones de memoria 0x3C y 0x3C de la memoria RAM y se mostrarán por la línea RS232_TX:

Ilustración 9: Transmisión TX


PlCtop_tb_behav.wcfg
Name
Value
PlCTOP
PlCtop
RS232
TX
ACK_FROM_RS
RX
Clk
Reset
LineRD_in
Valid_Out
s_valid_out_aux
Code_out
Store_out
s_RX_current_state
s_RX_next_state
s_RX_dataCount[2:0]
SR
SR
Enable
D
Q[7:0]
register_aux[7:0]
FIFO
FIFO
din[7:0]
wr_en
rd_en
dout[7:0]
srst
full
empty
DMA
DMA
s_DMA_current_state
idle
Idle
1,250.000000 us

Ilustración 10: Transmision RX

5. Extras

- Todo el codigo usado se encuentra en el repositorio : https://github.com/mariorra/LCSEL
- Los diagramas se pueden acceder . https://github.com/mariorra/LCSEL/Diagramas , se podrá visualizar abriéndolo en draw.io el archivo XML Diagrama Bloques LCSE.XML