UART

Tuesday, 9 November, 2021 11:34

- Se envía información de a "paquetes"
- Se transmite el LSB primero (MSB al final) (big endian)
- Comunicación asincrónica porque no hay una línea que una transmisor y receptor donde se envíe el clock sino que transmisor y receptor se ponen de acuerdo al principio para decidir a qué frecuencia va a ocurrir la comunicación
  - El baud rate generator nos permite dividir la frecuencia que le llega al periférico (CCLK/2, CCLK/4, CCLK/8…), y pasa a otro bloque donde se pasa a dos divisiones (una fraccional y una entera)
  - La idea de estos divisores es ajustar lo mejor posible el funcionamiento de los dos bloques, ajustando también la frecuencia de la comunicación
  - La salida Un_TXD hará la transmisión a una frecuencia 16 veces menor a la frecuencia del bloque
  - La entrada Un_RXD hará la toma de señal a una frecuencia 16 veces menor a la frecuencia del bloque
- La línea está en 1 cuando no se está transmitiendo y el trigger para comenzar la comunicación se da mediante un cero enviado por TX
- Pueden enviarse paquetes de 5, 6, 7 u 8 bits
- El bit de paridad sirve para chequear que la cantidad de '1's que haya en el mensaje se corresponda con la paridad indicada
- Si la configuración del receptor está en modo par, el bit de paridad le agrega un bit si la cantidad de bits enviados es impar, y viceversa (se genera una interrupción para avisar que hubo un problema con el mensaje)
- Transmisor:
  - Hay un registro que almacena lo que se quiere transmitir
  - Este dato se manda a una FIFO con 16 posiciones de 8 bits para acumular los datos a enviar
  - Cuando comienza la transmisión, la FIFO envía los datos al registro que envía de a uno los bits de información a través de Un_TDX
  - Se pueden enviar esos bits de información a través del DMA
- Receptor:
  - Hay un registro que guarda uno a uno los bits que llegan y los pasa a una FIFO (también de 16 posiciones de 8 bits cada una) cada 8 bits
  - La FIFO envía los datos de 8 bits (1 byte) al registro buffer del receptor para que el programador pueda acceder a la información
  - Podemos acceder a los datos que llegan a través del DMA
  - Podemos configurar si queremos una interrupción cada vez que llega un byte o a los 4 bytes.
- La línea de control y estado genera una interrupción si hubo un error
  - Permite generar una autoconfiguración de la frecuencia de comunicación
- El baud rate del UART se calcula de la forma:
  - UART_BR = PCLK/(16*(256*UnDLM + UnDLL)*(1 + (DivAddVal/MulVal)))
    - Al hacer 256*UnDLM + UnDLL lo que estamos haciendo es desplazar a la izquierda UnDLM 8 bits y concatenarlo con UnDLL para así generar un número de 16 bits
    - Para obtener la frecuencia de transmisión y recepción basta con pasar el 16 multiplicando a UART_BR
    - UnDLM y UnDLL son los encargados de realizar la división entera de frecuencia
    - MULVAL y DIVADDVAL son registros de 8 bits que generan la división fraccional de frecuencia
    - Debe cumplirse que:
      - 1 <= MULVAL <= 15
      - 0 <= DIVADDVAL <= 14
      - DIVADDVAL < MULVAL
- Procedimiento para calcular el baudrate (dado un PCLK y un BR definido, cómo calcular el resto):
  - Determinamos la división entera haciendo DL = PCLK/(16*BR)
  - Si DL es entero:
    - No hace falta parte fraccional
    - DIVADDVAL = 0 y MULVAL = 1
    - La división entera será dada por el valor DL calculado, haciendo la asignación de la forma:
      - DLM = <15:8> DL
      - DLL = <7:0> DL
  - Si DL no es entero:
    - Definimos una variable FR = 1.5
    - Establecemos que DL será la parte entera de PCLK/(16*BR*FR)
    - El nuevo valor de FR será PCLK/(16*BR*DL), usando el nuevo valor obtenido de DL
    - Si el nuevo valor de FR está entre 1 y 1.9:
      - Se definen DIVADDVAL y MULVAL en base al FR encontrado, buscando en una tabla que nos da los valores correspondientes para cada FR
    - Si el nuevo valor de FR no está entre 1 y 1.9:
      - Tomo otro valor para FR entre 1.1 y 1.9 y repito los cálculos
  - Con este procedimiento tendremos un error relativo menor al 1.1% de la frecuencia deseada
- Transferencia NONE BLOCKING:
  - Envía los datos en la FIFO de 16 bytes y libera al micro para que siga con lo suyo
    - Si se quiere enviar, por ejemplo, "*Hola mundo*", no hay problema porque tiene menos de 16 bytes, entonces la FIFO almacena todo, pero si se quiere enviar una cadena de más de 16 bytes con NONE_BLOCKING, se van a enviar sólo los primeros 16 bytes
- Transferencia BLOCKING:
  - Envía los datos en la FIFO de 16 bytes y se queda esperando por datos nuevos durante un tiempo de timeout
  - Sirve para enviar datos de mayor longitud que la capacidad de la FIFO
  - Si se quisiera utilizar NONE_BLOCKING para enviar datos de mayor longitud, deberían emplearse interrupciones
- Podemos tener interrupciones porque hay un problema en la línea (paridad, overrun), porque hay datos disponibles para enviar, porque la FIFO del transmisor se vació (no hay nada más que enviar) o porque hay datos disponibles en la FIFO del receptor para utilizar