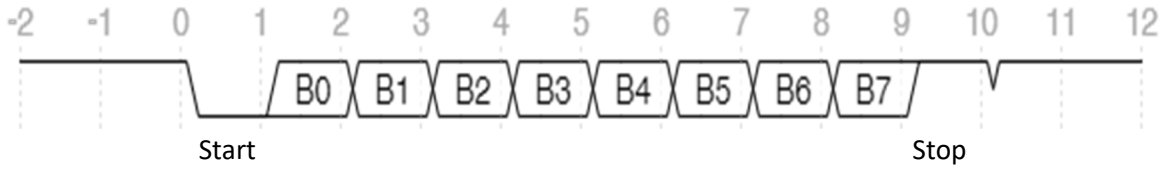
Bit banging serial on the Southern Cross SBC

Bit banging a serial port is an easy and cheap way to connect your SC to the outside world. In this article we will look at how bit banged serial transmit and receive have been implemented in Z80 assembly language.

By Craig RS Jones



Asynchronous Serial Character
8N1 = 8 bits, no parity, 1 stop bit, character length = 10 bits

An asynchronous serial byte or character is shown above, it's called **asynchronous** serial because there is no common clock signal between the receiver and the transmitter, instead each character or byte has start and stop bits added to provide synchronisation.
Each of the bits, including the start and stop bit are output for the same amount of time, this 'bit time' is equal to **1/ number of Bits Per Second(BPS) or Baud rate** . For example, at 4800 Baud the bit time would be 1/4800 = 208us.
The most important thing for successful transmission is to have the receiver and the transmitter using the same Baud rate!

Serial Transmit

There are three parts to the transmit code, the first is to send the start bit, the second to send the data and the last part to send the stop bit or bits.

1. Start Bit transmission

The FTDI USB to serial converter expects the RXI (Receive Input) to normally be High, so to send the start bit our code simply makes the data line Low and delays for one bit time.

2. Send the data byte

To send the data byte we must output each bit in turn for the bit time on the output pin. The serial data output is the Bit 6 position in the output register for the TEC so we have to shift all the bits in the data byte to the bit 6 position for output.
The RRC instruction shifts the bits in the register one bit to the right, shifting the bit in position 0 into the bit 7 position with each shift.
To start we put the loop or bit count in B, the first RRC instruction moves all the bits one position to the right, bit 0 moves into the now empty bit 7 position.

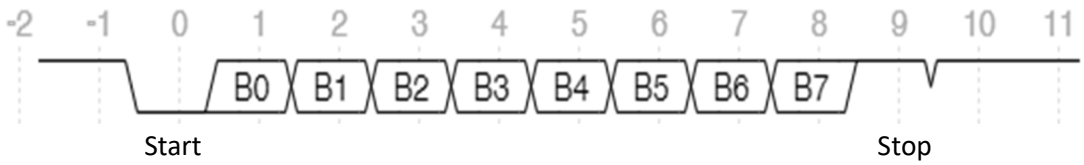
Now we just execute the loop 8 times, shifting each bit into the bit 6 position for output and then doing a bit time delay.

3. Stop Bit transmission

Finally, set the output pin high and output each stop bit for the bit time. The next character can then be sent, any additional delay here is usually called the 'inter-character delay'.

```
BAUD = delay constant for Baud rate

;-----------------------------------------
;transmit a serial character
;-----------------------------------------
; C = character to transmit
;
;
; send the start bit
;
        LD      a,$00
        OUT     (PORT),A      ;start bit = 0
        LD      HL,(BAUD)
        CALL    BITIME        ;one bit time delay
;
; set up to shift out the character
;
        LD      B,08H         ; 8 bits to transmit
        RRC     C
SHIFTOUT:
; move all the bits one position to the right
        RRC     C
        LD      A,C
        AND     40H
;output the bit and delay for the bit time
        OUT     (PORT),A
        LD      HL,(BAUD)
        CALL    BITIME        ;one bit time delay
;have we done all the bits?
        DJNZ    SHIFTOUT      ;jump if not finished
;
; send the stop bit (or bits)
;
        LD      A,$40
        OUT     (PORT),A      ;stop bit=1
        LD      HL,(BAUD)
        CALL    BITIME        ;send the stop bit
        CALL    BITIME        ;or two if required
        RET
```



Serial Receive Routine

Receiving a character is a little more complicated and starts with synchronising, waiting in a loop until the falling edge of the start bit arrives.

1. Start bit detection

Once the falling edge is detected we delay for only half a bit time, until time '0' in the diagram above. To calculate the delay time we divide the BAUD variable, which is the bit time delay value, by 2 by shifting the 16-bit HL register one bit to the right. The instruction, SRL (Logical shift right), shifts the byte in the H register one bit right, shifting Bit 0 into the carry, the next instruction, RR (Rotate Right through carry) shifts the bit out of the carry into the bit 7 position of the L register. Using these two instructions on the H and L registers performs a divide by 2 on the 16-bit register HL .

Once the half bit time delay finishes another check of the input level is done, if it is still low we have a valid start bit, if not then we have what is called a 'framing error', so we just go back and wait for the next falling edge of a start bit.

2. Receive the data byte

You can see in the diagram that we are now at time 0, halfway into the start bit, now we can simply read in the data byte delaying one full bit time before reading the middle of the next bit in the eight bit character.

Since we are reading our data in on bit 7of the input register, we do a RL, (Rotate Accumulator Left) on the input byte, shifting bit 7 into the carry bit and then doing a RR,(Rotate Right through carry) to move the received bit into the C register where our received byte will be assembled. After 8 bits have been received our character is in the C register.

3. Check the stop bits?

,We don't really care about checking our stop bits and there is probably something we want to do with our received byte, time is of the essence when receiving bit banged serial!

We don't bother checking the stop bits and just return the received character from the subroutine.

```
;----------------------------------------
; receive a serial character
;---------------------------------------- .
; wait for the start bit falling edge
STARTLOOP:
        IN      A,(KEYBUF)
        BIT     7,A
        JR      NZ,STARTLOOP
; delay for half the bit time
        LD      HL,(BAUD)
        SRL     H
        RR      L
        CALL    BITIME
; is the start bit still low?
        IN      A,(KEYBUF)
        BIT     7,A
        JR      NZ,STARTLOOP
; shift in the character
        LD      B,08H           ;8 bits to receive
SHIFTIN:
        LD      HL,(BAUD)
        CALL    BITIME          ;one bit time delay
        IN      A,(KEYBUF)
        RL      A
        RR      C
        DJNZ    SHIFTIN
        RET     ;character returned in c
;--------------
; BIT TIME DELAY
;--------------
BITIME:
        PUSH    HL              ;HL = delay time
        PUSH    DE
        LD      DE,0001H
BITIM1:
        SBC     HL,DE
        JP      NC,BITIM1
        POP     DE
        POP     HL
        RET
```

Conclusion

Bit Banging a serial port is a simple and effective way to connect a resource limited 'minimum system' like the SC to a computer. The ability to communicate this way opens up many possibilities for input and output, debugging, data logging, downloading/uploading code and data, communicating with serial peripherals; modems, radios etc.

Of course bit banged serial consumes all of the processor time during transmission and reception, which is why dedicated hardware devices such as the Z80 SIO and MC6850 ACIA were designed to offload the transmission and reception of serial data.

References:

Southern Cross Monitor TXDATA and RXDATA subroutines.
Jack Ganssle has a good description and some routines here;
http://www.ganssle.com/articles/auart.htm
Sparkfun has an extensive tutorial;
https://learn.sparkfun.com/tutorials/serial-communication