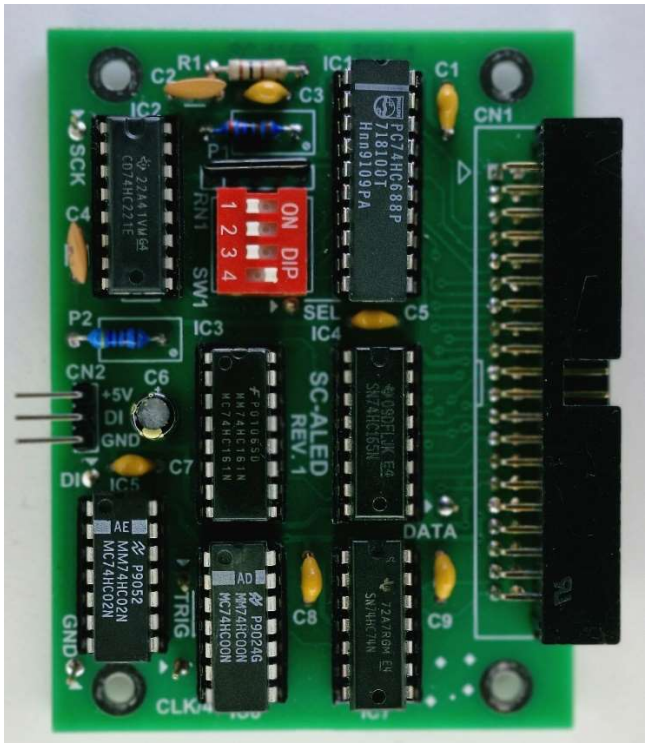
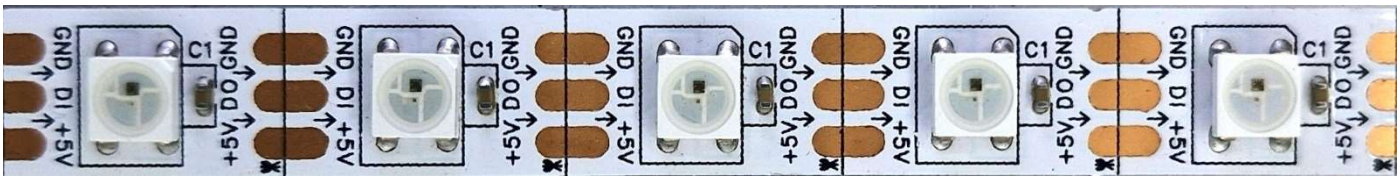
Addressable RGB LED Hardware Interface for the Southern Cross SBC



By Craig RS Jones

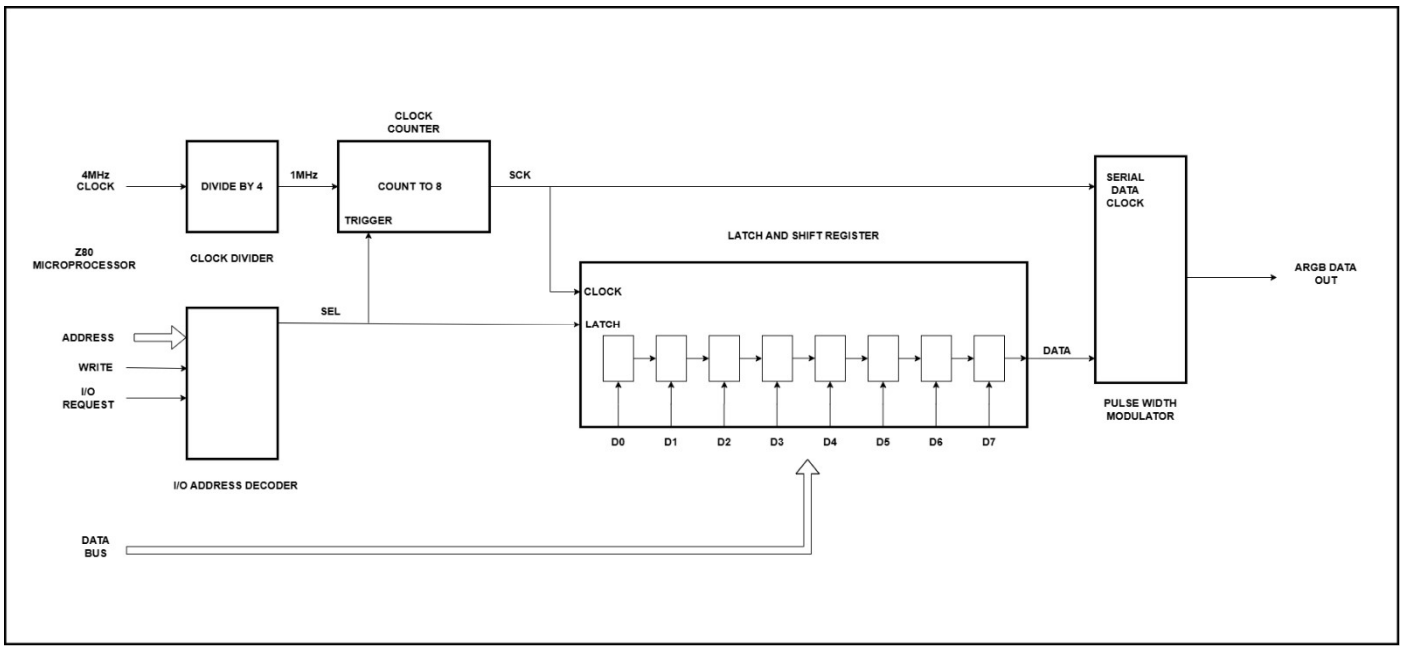


I have to admit it, I love BlinkenLEDs.

This Southern Cross SBC (SC) add-on board gives the 4MHz Z80 microprocessor a helping hand with some hardware acceleration, serialising RGB LED data and producing a PWM data stream suitable for WS2812B and other compatible Addressable RGB LED's (ARGB's).

To program these vibrant ARGB's, simply write the 0-255 bit Intensity values for red, green and blue to the ALED board and it will handle the rest.

Block Diagram



The 60 x 80mm board is packed with chips, seven in all, to provide the five functional blocks; I/O address decoder, latch and shifter, clock divider, clock counter and Pulse Width Modulator (PWM).

I/O Address Decoding

There are sixteen different I/O locations of four bytes each that the board can use via the four way DIP switch. The SEL output of the decoder is used to latch the parallel Z80 data bus byte into the shift register and also triggers the clock counter to start shifting out the serialised data.

Latch and shifter

Eight bit data from the data bus is latched into the shift register latch when the SEL line is low, Once latched the data is ready to be shifted out to the PWM, Most significant bit (bit 7) first.

Clock divider

The nominal frequency of operation of ARGB's is usually quoted as 800kHz, but for this board we are going to use a 1MHz clock divided down by four from the 4MHz SC crystal.

Clock Counter

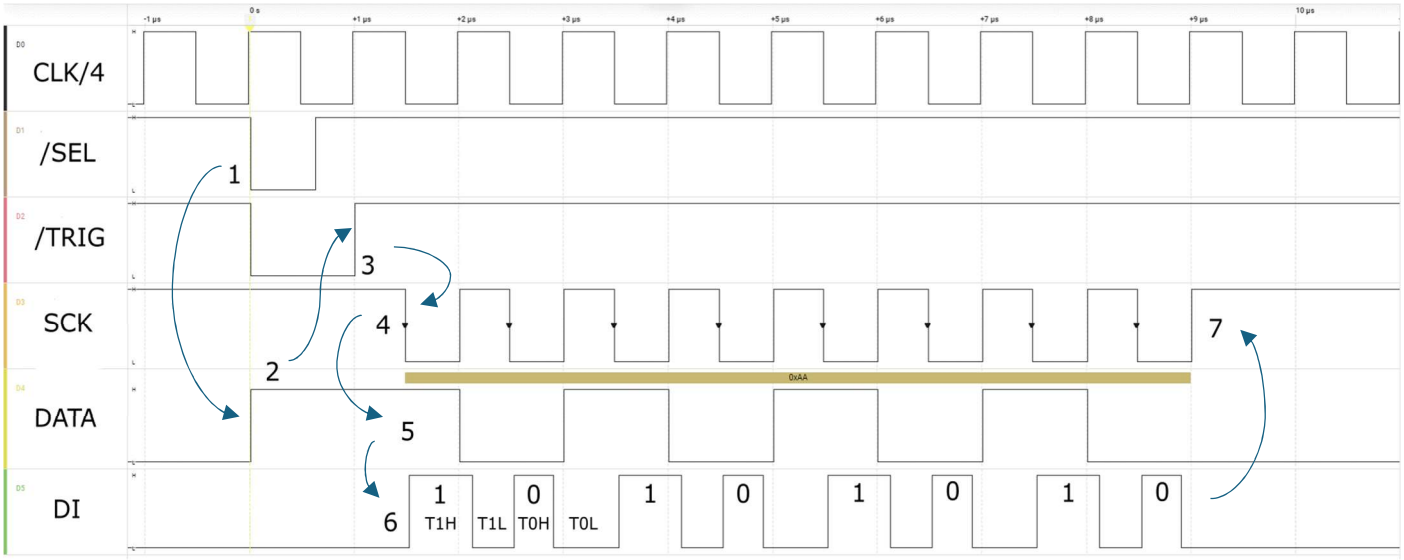
A counter is used to generate the eight clock pulses we need to clock out the shift register and is started after the data has been latched. After eight clock pulses the counter is disabled until the next data is latched.

PWM generator

The PWM generator is comprised of two monostable multivibrators, one set to deliver a 'High' PWM pulse and the other a 'Low' PWM pulse. For each bit, the low PWM pulse is always generated by SCK.

If the bit from the shift register is high then the high pulse is also generated, both the high and low pulses are OR'ed together to produce the output pulse train.

Timing Diagram



There are seven test points on the board, CLK/4, /SEL, /TRIG, SCK, DATA, DI and GND.
The timing diagram above shows the waveforms on each test point when a single byte is written to the I/O port.
Operation is as follows; (refer to the timing diagram above and the circuit diagram)

1. A Low level on /SEL produced by the I/O Address Decoder 74HC688 (IC1) when there is a write to the board latches the data from the data bus into the Latch and Shift register 74HC165 (IC4).

2. As the Data is latched, the DATA output, Pin 9 (QH) outputs the latched D7 value, a 1 in this case for the example data, %10101010, AAh.

3. The NAND gate IC6D, wired as an inverter, inverts and applies the /CSEL signal to the RS Flip-Flop formed by the NOR gates IC5C and IC5D, setting the /Q output, /TRIG Pin 10 Low.

   Preset data P3 - P0 (1000_2) is latched into the counter 74HC161 (IC3) when the /TRIG output returns High on the rising edge of CLK/4 which also resets the RS Flip-Flop.

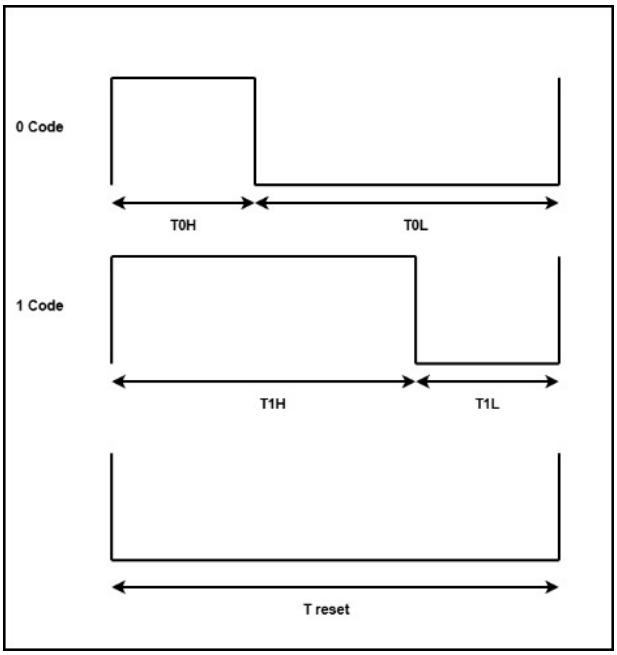
   The Counter, previously inhibited from counting by the Low Q3 (Pin 11) on PE, is set high by the P3 preset and starts to count CLK/4 clock pulses. The NAND gate (IC5A) now outputs the SCK.

4. The SCK now provides a low going pulse to trigger the 'LOW' PWM Pulse Monostable 74HC221 (IC2A) producing a T0H pulse for every clock pulse.

5. The SCK also clocks the data out of the Shift Register and triggers the 'HIGH' Pulse Monostable (IC2B) when the DATA is High on the falling edge of SCK producing a T1H pulse.

6. 'OR' gate made up of IC5A and IC5B combines the two pulse trains into one output, the longer 'High' PWM pulse overriding the 'Low' PWM pulse in the output stream when it occurs.

7. When the counter counts eight clock pulses output Q3 returns Low, returning PE Pin 7 Low stopping the counter. The circuit is then ready for the next byte of data.



IT'S ALL IN THE TIMING

To create the output PWM waveform for each bit the 4MHz oscillator of the SC is divided by four to provide a 1MHz clock.
The PWM pulses are created using monostable multivibrators.
The 'Low PWM Pulse' (T0H) is set to 300ns and the 'High PWM Pulse' (T1H) is set to 700ns.

Not all monostable multivibrators are the same and generally devices from different manufacturers require different timing components for the same output pulse width.

Here is a table for some devices I have tested and the required resistor values using a capacitor of 100pF;

Device	Manufacturer	P1 T0H = 300nS	P2 T1H = 700ns
74HC221	TI	2k2	5k6
74HC123	Toshiba	1k3	4k3
74HC123	Philips	3k6	9k1

Provision has been made on the board for either a fixed resistor or a trim pot if you want to adjust the timing yourself. You can adjust the timing of the monostables without any test equipment but it's easier with an oscilloscope or logic analyser. There is some example code to use when adjusting the monstable timing.
Use 1% resistor values and capacitors with a good dielectric material, such as NP0/C0G.
The PCB has provision for a through hole capacitor with a 5mm pitch or a 0805 Surface Mount capacitor.

SUPPLYING POWER

ARGB LEDs are very bright! Each LED in an ARGB LED can draw up to 20mA so for full brightness white (red+green+blue = ffh) 60mA will be drawn.
The SC itself draws very little power so it's possible to use the on-board 5V to drive a small number of ARGB's. It's a better idea to get the power for the ARGB's from another source.

THE CODE

```
shifter   .equ      10h       ;ALED board default I/O address
                              ; SW1:  SW1 to SW3 OFF, SW4 ON
;
; Make one WS2812B white
; WS2812B's accept GRB, Green, Red, Blue data
;
wrtARGB
          ld        a,0ffh          ; ffh is fully On
          out       (shifter),a     ;output Green colour data
          nop
          nop                       ;allow some time for the data
          nop                       ;to be clocked out!
          nop
          nop
          ld        a,0ffh
          out       (shifter),a     ;output Red colour data
          nop
          nop
          nop
          nop
          nop
          ld        a,0ffh
          out       (shifter),a     ;output Blue colour data
          nop
          nop
          nop
          nop
          nop
;
; finish the output sequence, delay for more than 280us
;
ResetDelay
          ld        b,$10
ResetDelay1   nop
          djnz      ResetDelay1
          ret
```

This code outputs to a single WS2812B, to output to more than one ARGB, repeat the writing sequence for the three colours for as many ARGB LEDs as you have and use the ResetDelay subroutine at the end of the complete writing sequence to tell all the LEDs to expect new data.

LINKS

Here are a few links that you may find helpful.

Adafruit's Neopixel UberGuide
https://learn.adafruit.com/adafruit-neopixel-uberguide

Ben Heck's SPI to Neopixel Adapter.
https://www.youtube.com/watch?v=aCK0SpZ2ueg&t=932s

What's Behind the Light? – How WS2812B LED Strips Work
https://www.youtube.com/watch?v=rHoFqKGOPRI

SPI to WS2812(B) Addressable RGB LED Converter - More LEDs, More Power!
https://www.youtube.com/watch?v=5z7CvM6QTDo

Neopixel (WS2812B) timing analysis.
https://wp.josh.com/2014/05/13/ws2812-neopixels-are-not-so-finicky-once-you-get-to-know-them/

https://wp.josh.com/2014/05/15/going-nsa-on-pixel-conversations/