

Project Number: A3179

Composite Text Video Display

The Composite Video Text Display can generate a 40x25 character display on a standard television set through the composite video input jack. The circuit supports the North American NTSC standard and the PAL standard.

Theory of Operation

The circuit is based around the ATmega8 microcontroller. It has enough speed and internal resources to provide a basic black and white display with a minimum of external parts (see Fig. 1). The software takes only 3k of the program memory, leaving enough space to provide a 512 character font. Display data is stored in the internal RAM.

Program operation involves two processes (see Fig. 2). The main loop receives data from the Host device and updates the internal RAM accordingly. The 8-bit data presented at the input can contain either a character or command. A character can be any printable ASCII character such as letter, numbers, and punctuation marks. Commands include several ASCII control characters such as line feed, carriage return, tab, and backspace. In addition, the cursor can be placed anywhere on the screen, turned on and off, and displayed as underscore, block, blinking or solid.

To generate the composite video signal, you need to provide video synchronization (sync) pulses and picture data. One of the ATmega8's internal **timers is used to generate interrupts every 63.5µs. Interrupt service routines are used** to provide the necessary sync pulses via a digital I/O port. The picture data is generated by matching the character data from RAM with the appropriate font data in program memory. The font data is written to a shift register and the resulting serial data is combined with the sync pulses to generate the display.

The Interrupt Routines use Timer 1 in a two-stage method to ensure exact timing. Using Mode 4 (CTC), Output Compare Register A (OCR1A) is set to provide interrupts every 63.5µs. Output Compare Register B is set to interrupt the system just prior to OCR1A with enough time to save system settings and put the CPU into sleep mode. This ensures the OCR1A interrupt is responded to at exactly the same time (very important to proper sync timing). Here's how its looks:

```
TIM1_COMPB:                                ; IRQ service for OCR1B
    in     savsr, SREG                      ; save the status register
    sbi   portc, syncpin                   ; raise the sync pin (its low during VSYNC, else no
effect)
    sbi   portc, ackdpin                    ; raise dack pin (if it was low just before irq)
                                           ; this keeps host from waiting...
    sei                                       ; enable IRQ before we sleep so OCR1A can work
    sleep                                   ; wait for TIM1_COMPA IRQ
    out   SREG, savsr                       ; restore the status register
    reti                                    ; all done, go back to Main Prgm

TIM1_COMPA:                                ; IRQ service for OCR1A
    .include "vidgen.inc"                   ; file that holds the video generation code
    reti                                    ; done. Resume after the sleep instruction
                                           ; in the OCR1B routine above.
```

Using the Display

The circuit was assembled onto a double-sided PCB measuring only 2.5" x 1.9" (see fig. 3). The circuit can be connected to any device capable of producing the 8-bit data and an active high strobe. The Busy line should be monitored to ensure the previous command has completed. This is especially true during scrolling operations, as it can take several milliseconds to scroll the entire screen. If you cannot monitor the busy line, then give at least 25 milliseconds between characters to prevent data loss.

After power up, the circuit initializes itself and displays a startup banner (see Fig. 4). Initial state is Primary font selected, lower 128 character mode enabled, with a blinking underscore cursor. To clear the screen, you can send a

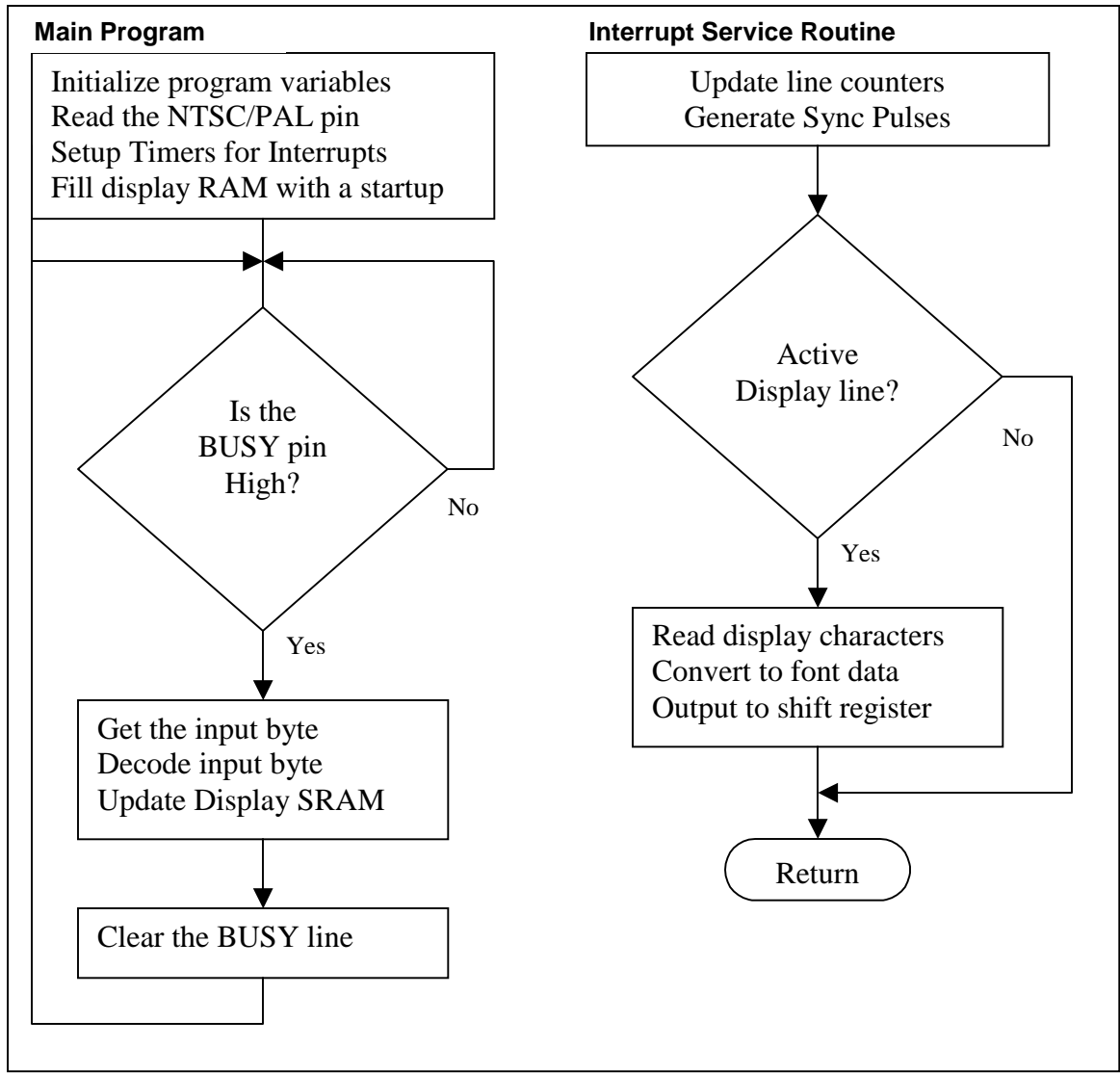


Figure 2. Software Block Diagram



Figure 4. Sample Display