

Get more than three colors from a dot-matrix LED

W KURDTHONGMEE, NAKORN SI THAMMARAT, THAILAND

Dot-matrix LEDs find wide use in advertising displays. Products now on the market range from an inexpensive 5×8 (row-by-column) single-color LED to an expensive 8×8 RGB device. The method provided here allows you to obtain more than three main colors from an 8×8 tricolor LED. In fact, tricolor dot-matrix LEDs have only two LED dies—red and green. When you apply current to one, you obtain a red or a green color. When you apply current to both, orange results. The circuit in **Figure 1**, used in conjunction with the MCS-51 code in **Listing 1**, works efficiently in controlling the LED to generate various shades of the three colors.

To add tones or shades of the main colors to the tricolor LED, you do not need to modify the circuit in **Figure 1**; you need only consider the software. Software modifications consist of adding more color planes or pages of display buffer, adding memory locations (mapped onto the LED dots), and increasing the number of refresh times, in which the controller updates all LED dots to cover all added color planes.

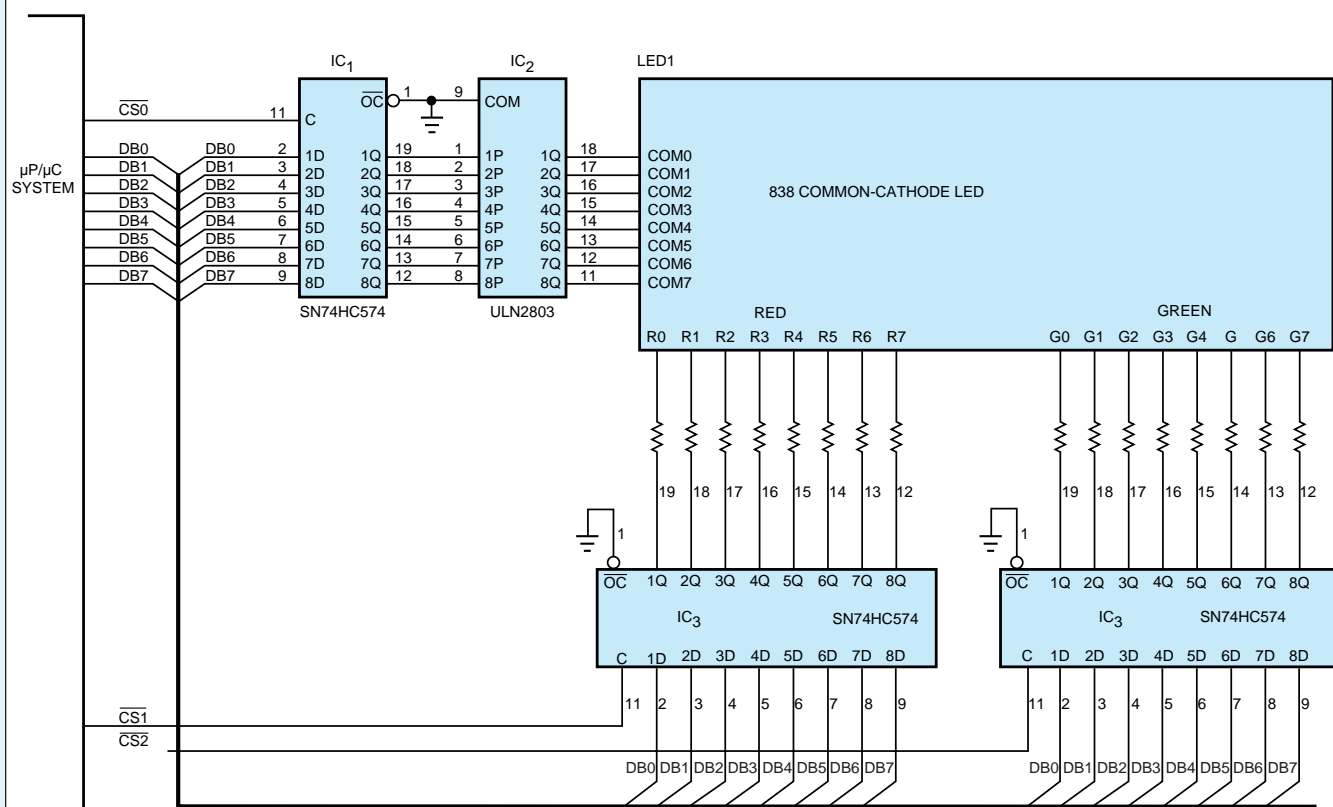
TABLE 1—VALUE IN COLOR PLANE MAPPED TO DOT I

Red 1	Red 2	Green 1	Green 2	Color
0(1)	1(0)	0	0	Red 50%
1	1	0	0	Red 100%
0	0	0(1)	1(0)	Green 50%
0	0	1	1	Green 100%
0(1)	1(0)	0(1)	1(0)	Orange 50%
1	1	1	1	Orange 100%
0	0	0	0	Blank

For example, if you decide to use four color planes, divided into two red and two green planes, for dot i of the dot-matrix LED, you'll obtain the shades listed in **Table 1**.

In addition, by allocating eight color planes (four red and

FIGURE 1



A few TTL circuits and some MCS-51 code allow you to obtain more than three colors from a tricolor dot-matrix LED.

four green), you can obtain the color shades listed in **Table 2**. Note that only the number of ones in the color planes controls color appearance. Therefore, the permutations do not change the color, as long as the numbers of ones in **Table 2** remain constant. For example, the values 0110, 1001, 1100, and 0011 for R1 through R4 all produce the same color: orange 50%. You can download **Listing 1**—as well as the MCS-51 code that produces 13 colors from an 8×8 tricolor LED—from *EDN's* Web site, www.ednmag.com. At the registered-user area, go to the “Software Center” to download the file from DI-SIG #2195.

Note that, in practice, bitwise output ports control the LED. To assign a color to a dot, the routine must extract a bit from a byte and then assign the bit value of the selected color plane by plane. (DI #2195) e

TABLE 2—VALUE IN COLOR PLANE MAPPED TO DOT 1

Red 1	Red 2	Red 3	Red 4	Green 1	Green 2	Green 3	Green 4	Color
0	0	0	1	0	0	0	0	Red 25%
0	0	1	1	0	0	0	0	Red 50%
0	1	1	1	0	0	0	0	Red 75%
1	1	1	1	0	0	0	0	Red 100%
0	0	0	0	0	0	0	1	Green 25%
0	0	0	0	0	0	1	1	Green 50%
0	0	0	0	0	1	1	1	Green 75%
0	0	0	0	1	1	1	1	Green 100%
0	0	0	1	0	0	0	1	Orange 25%
0	0	1	1	0	0	1	1	Orange 50%
0	1	1	1	0	1	1	1	Orange 75%
1	1	1	1	1	1	1	1	Orange 100%
0	0	0	0	0	0	0	0	Blank

To Vote For This Design, Circle No. 417

LISTING 1—MCS-51 CODE FOR SIX COLORS FROM A TRICOLOR LED

```

;*****
; Program Name : TriCol.asm
; Generate three colours from triple colours dot matrix LED.
;*****
; Note:
; Only a sample implementation on 8x8 triple colours LED
;*****
; Author
; : Wattanapong Kurthongmee,
; : School of Physics, Walailak University, Thaiburi,
; : Tha-Sa-La, Nakhon si Thammarat, 80160 Thailand.
;*****
;*****
; Global variable declarations:
;*****
temp equ 10h
dBuf equ 11h

;*****
; Interrupt vectors
;*****
org 0000h
jmp main
org 000bh
jmp Timer0SR
;*****
; Timer 0 interrupt service routine
; Update row-by-row the triple-colour 8x8 dot matrix LED.
; To make more tones of colour, this routine reads 4 set of display buffer
; and scan 4 times.
;*****
Timer0SR:
setb rs0 ; Select set of registers in bank 1
push dph
push dpl
push psw
push acc
mov th0,#0ffh ; Re-initialize timer registers
mov t10,#00h

mov dptr,#0a000h ; Clear enable controlled port
mov a,#00h ; before updating
movx @dptr,a

mov a,#dBuf
add a,r0
push acc
mov r1,a
mov a,@r1 ; Read from current address at a current

; plane and update column controlled port
; colour by colour
; Red colour port
mov dptr,#8000h
movx @dptr,a
pop acc
add a,#08h ; Difference display buffer between RED and
; green colour plane

mov r1,a
mov a,@r1
mov dptr,#9000h ; Green colour port
movx @dptr,a

mov dptr,#0a000h ; Enable current row (controlled by ULN2803
mov r3,a
movx @dptr,a
r1
mov r3,a

inc r0 ; Next row (there are 8-row for 8x8 LED)
cjne r0,#08h,T0SR_r
mov r0,#00h
mov r3,#01h

T0SR_r:
pop acc ; Restore registers
pop psw
pop dpl
pop dph
clr r0
reti

;*****
; Main program starting here
;*****
main: mov sp,#60h

setb rs0 ; Initialise register in bank 1 to be used
mov r0,#00h ; the timer interrupt service routine.
mov r3,#01h
clr r0
mov tmod,#21h ; timer 0: mode 1, timer 1: mode 2
mov tcon,#0ddh
mov th0,#0fdh ; Initial value for timer
mov t10,#00h
setb ea ; Enable all interrupts
setb tr0 ; Start timer
setb et0 ; Enable timer interrupt 0

; Sample patterns to show different shades
; colours on the LED.
mov r0,dBuf
mov r1,#00h

main_0: mov a,#01010011b
mov @r0,a
inc r0
inc r1
cjne r1,#08h,main_0
mov r0,dBuf+8
mov r1,#00h

main_1: mov a,#10101111b
mov @r0,a
inc r0
inc r1
cjne r1,#08h,main_1
sjmp $

end

```