Application Report
SLVA267B–Mar 2007–Revised September 2018

Grayscale and Dot-Corrected LED Display Using TLC5941 and MSP430F427

ABSTRACT

The TLC5941 LED driver uses a serial data interface to receive grayscale and dot correction data. This application report describes the hardware and firmware to drive the TLC5941 with an MSP430F427 microcontroller (MCU).

Contents

Trademarks

MSP430 is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

The TLC5941 is a 16-channel LED driver with 12-bit grayscale dimming and 6-bit dot correction capabilities typically used for high-quality LED displays. The 192 bits of grayscale data and 96 bits of dot correction data are entered into the TLC5941 through a serial interface. This application report presents an example of the firmware necessary to drive the TLC5941 with an MSP430F427 MCU.

The serial interface is comprised of a shift register that is controlled by two signals, SLCK and SIN. One bit of serial data is placed on the SIN pin of the TLC5941; then, a positive edge on the SCLK pin shifts the data into the TLC5941 input register. Once all of the 96 or 192 bits have been shifted in, the XLAT pin is pulsed high to latch the data from the input register into the grayscale or dot correction registers.

The TLC5941 also requires a clock signal, GSCLK, for the grayscale PWM function. This clock is used to drive internal counters to derive the 12-bit grayscale dimming function. After 4096 grayscale clocks, the internal counter must be reset to zero by pulsing the BLANK signal high. This clocking function has also been included in the example MSP430F427 firmware to provide a stand-alone solution.

Copyright © 2007–2018, Texas Instruments Incorporated

2 TLC5941 Driver Implementation Using the MSP430F427

Figure 5 and Figure 6 show the schematic of the hardware for the implementation of the circuit with the MSP430F427 and TLC5941. The schematic shows the MSP430F427, the TLC5941, and power circuits needed to implement a complete 16-LED display drive. Additional TLC5941s can be cascaded on the serial communication bus to accommodate any number of LEDs.

The hardware is configured so that the TLC5941 is controlled by six general-purpose I/O (GPIO) pins on the MSP430F427. All six of these GPIO pins are configured as outputs on port one of the MSP430F427. One output pin (P1.1) is configured to be driven by the internal system clock of the MSP430™ MCU. This output pin drives the GSCLK of the TLC5941. The system clock, internal to the MSP430F427, also supplies the clock to a 16-bit counter which also is internal to the MSP430F427. The counter is configured to count from 0 to 4097 with each clock pulse. When the counter reaches 4097, it generates an interrupt. An interrupt routine then pulses the output bit that connects to the BLANK signal of the TLC5941. Figure 1 shows a simplified block diagram of this clocking scheme. This clocking scheme makes the GSCLK and BLANK signal generation transparent to the rest of the software. The firmware only needs to send dot correction and grayscale data to drive the LEDs.

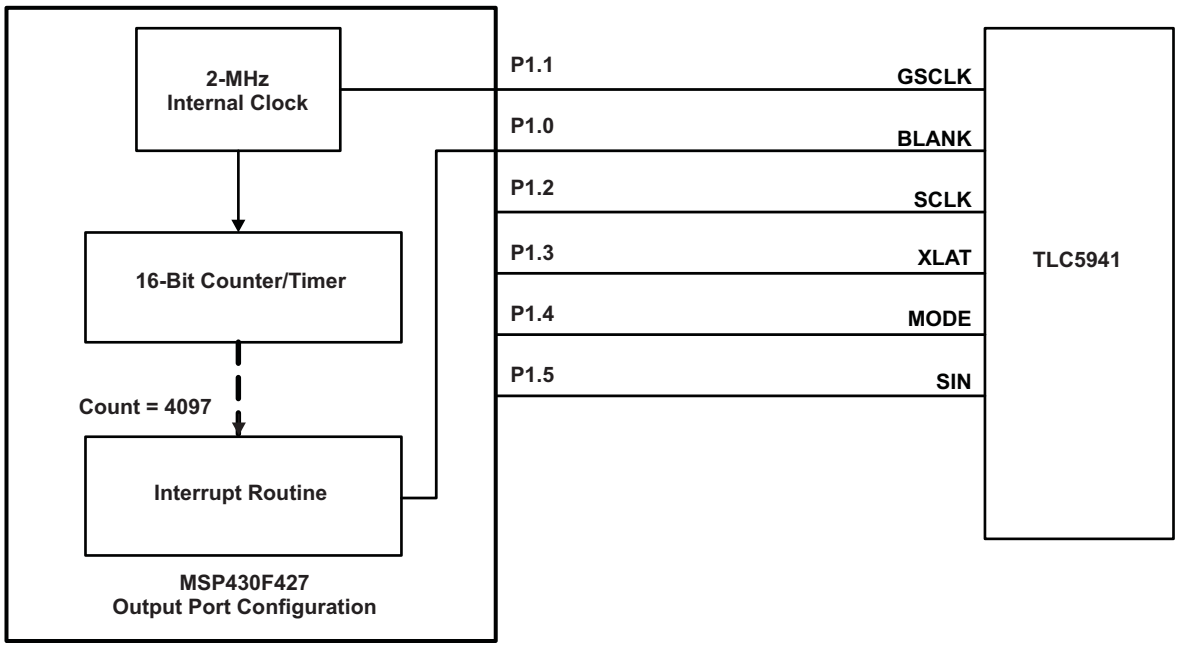

Figure 1. Simplified Block Diagram Of Clocking Scheme

Figure 2 shows a flow chart for the main loop of the firmware. The major portions of the communications firmware resides in three subroutines which makes the main loop very simple. The main loop simply initializes the output port, turns off the watchdog timer and configures the counter. Once these initialization steps are complete, the main loop simply sends all of the dot correction and grayscale data stored in RAM to the TLC5941 and then enters an infinite loop. The main loop ends in an infinite loop instead of halting the processor so that the interrupt routines continue to operate and drive the LEDs.

The GSOUT routine is used to send all of the grayscale data stored in RAM to the TLC5941. The 12-bit grayscale data is stored in RAM as left-justified, 16-bit words. Therefore, the lower four bits of each grayscale data in RAM are ignored by the firmware and are not sent to the TLC5941.

The DCOUT routine is used to send all of the dot correction data stored in RAM to the TLC5941. The 6-bit dot correction data is stored in RAM as left-justified, 8-bit bytes. Therefore, the lower two bits of each dot correction data in RAM are ignored by the firmware and are not sent to the TLC5941. Figure 3 shows the flow charts for the interrupt, GSOUT, and GSOUT routines

Copyright © 2007–2018, Texas Instruments Incorporated

The GSOUT and DCOUT routines are similar and could be combined, together with the addition of some flags and conditional statements. This was not done for this example to clarify the difference between grayscale and dot correction communications. Both of these routines determine which byte of data in RAM is to be sent, how many bits of that byte are to be sent, and then passes these two parameters to the SHIFTOUT subroutine. The SHIFTOUT subroutine simply takes the byte of data and drives the bits of the output port that drive the serial communication pins of TLC5941. Figure 4 is the flow chart for the SHIFTOUT subroutine.

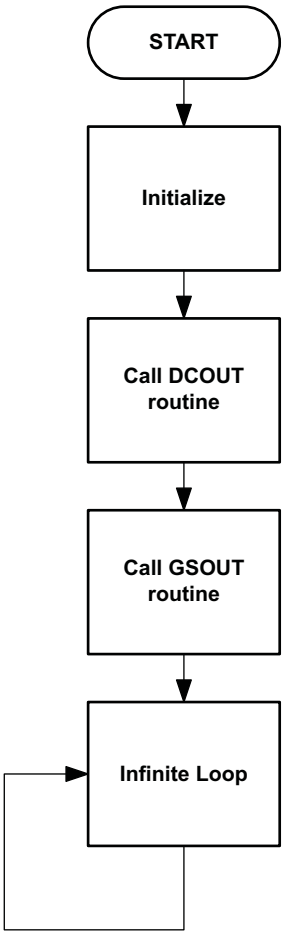

Figure 2. Main Loop Flow Chart

Copyright © 2007–2018, Texas Instruments Incorporated

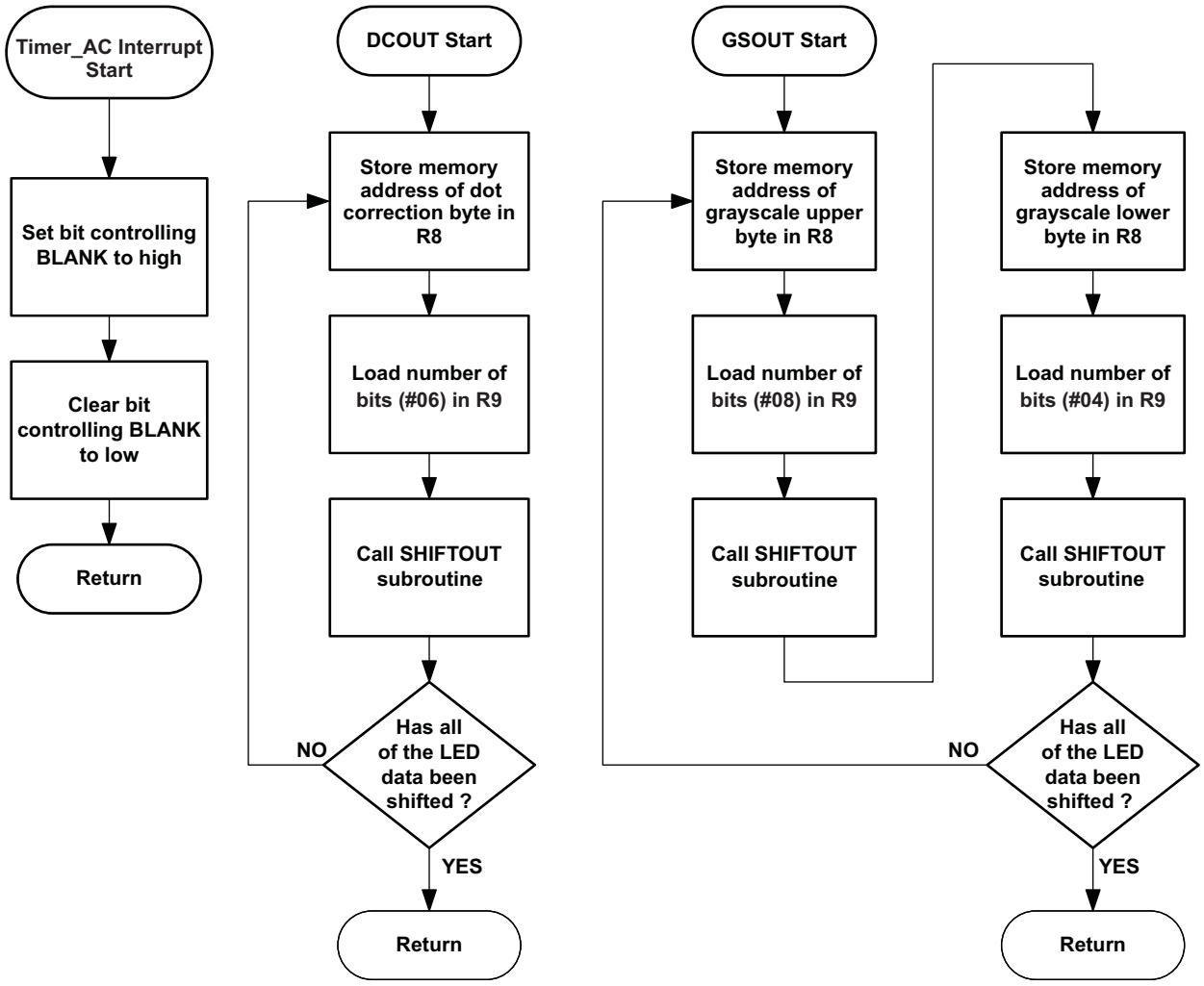

Figure 3. Interrupt and Subroutine Flow Charts

Copyright © 2007–2018, Texas Instruments Incorporated

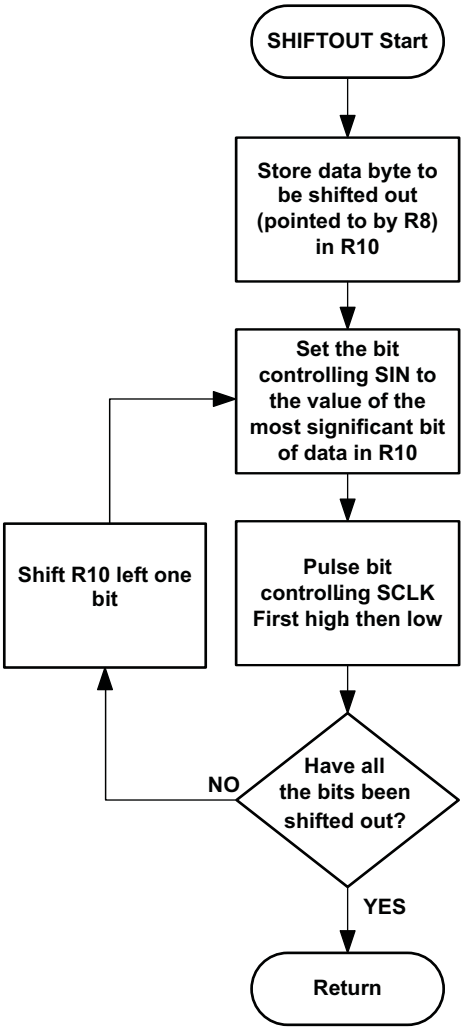

Figure 4. SHIFTOUT Flow Chart

Copyright © 2007–2018, Texas Instruments Incorporated

3 TLC5941 Driver Firmware Code

```
TLC5941 COMMUNICATIONS FIRMWARE
;
;Scot Lester
;Texas Instruments Incorporated
;February 2008
;
;This firmware was written for the MSP430F427
;
;The hardware is configured to have one general
;purpose I/O port (GPIO) configured as all bits being
;outputs. For this example, 6 bits of port one are used to
;drive the TLC5941. Each of the six used GPIO pins of the MSP430
;are connected as follows:
;
; PIN NAME   TLC5941 FUNCTION
;
; P1.0    BLANK
; P1.1    GSCLK=2MHz
; P1.2    SCLK
; P1.3    XLATCH
; P1.4    MODE
; P1.5    SIN
; P1.6    not used
; P1.7    not used
;
#include "msp430x42x.h"
;-------------------------------------------------------------------------------
;    Definition of named constants
;-------------------------------------------------------------------------------
#define BLANK (0x01)
#define SCLK (0x04)
#define XLAT (0x08)
#define MODE (0x10)
#define SIN  (0x20)

;P1OUT is equal to the memory address of output port.

;-------------------------------------------------------------------------------
;    RESERVED RAM MEMORY FOR VARIABLES
;-------------------------------------------------------------------------------

; Dot Correction values are stored sequentially in 8 bit bytes.
; The TLC5941 only needs 6 bits of dot correction data so the
; dot correction data is stored left justified in the 8 bit byte
; the two least significant bits of each 8 bit byte are set
; to zero in this example but the firmware ignores these
; two bits so they can be any value.
; For example, a binary dot correction value of 101010b = 0x2A
; is stored as 10101000b = 0xA8
ORG    0200h     ; Start of RAM
       EVEN      ; Align data on even boundary
Dot
    DB    0xFC   ;CH0
    DB    0xFC   ;CH1
    DB    0xF4   ;CH2
    DB    0xF8   ;CH3
    DB    0xF8   ;CH4
    DB    0xFC   ;CH5
    DB    0xF4   ;CH6
```


Copyright © 2007–2018, Texas Instruments Incorporated

```
    DB    0xF0   ;CH7
    DB    0xF4   ;CH8
    DB    0xFC   ;CH9
    DB    0xFC   ;CH10
    DB    0xFC   ;CH11
    DB    0xF8   ;CH12
    DB    0xF4   ;CH13
    DB    0xF4   ;CH14
    DB    0xF8   ;CH15
;
;
; Grayscale values are stored sequentially in 16 bit words.
; The TLC5941 only needs 12 bits of grayscale so the
; grayscale data is stored left justified in the 16 bit word
; the four least significant bits of each 16 bit word are set
; to zero in this example but the firmware ignores these
; four bits so they can be any value.
; For example, a hex grayscale value of  0xFFF
; is stored as 0xFFF0
;
Grayscale
    DW    0xAFF0   ;CH0
    DW    0xF000   ;CH1
    DW    0xF000   ;CH2
    DW    0xC000   ;CH3
    DW    0xD400   ;CH4
    DW    0xE000   ;CH5
    DW    0xB000   ;CH6
    DW    0xF000   ;CH7
    DW    0xFFF0   ;CH8
    DW    0x0000   ;CH9
    DW    0x0000   ;CH10
    DW    0x13F0   ;CH11
    DW    0x2380   ;CH12
    DW    0xE980   ;CH13
    DW    0x0450   ;CH14
    DW    0xBA30   ;CH15

;-----------------------------------------------------------------
    ORG   0C000h   ; Starting Address of Program Space
;-----------------------------------------------------------------

RESET    ;reset jump vector jumps to here to start execution.

;
;Program execution begins here after power up and reset
;Typical start up routines should be located here.
;Routines to initialize the stack pointer, watch dog timers, parallel I/O
;ports etc. etc.
;the General Purpose I/O port (P1) needs to be configured with bits 0:5
;set to outputs. At initialization, all output bits should be cleared to zero.
;this is not shown since it is processor specific.
;Bit P1.1 has a special configuration.  The output bit is driven by the
;MSP's internal oscillator which is set to 2MHz.  This drives the
;grayscale clock of the TLC5941.  Since the output signal is simply the
;system clock, there is no need for the firmware to manipulate this bit.
;The grayscale clock will run continuously.

    MOV.W   #600h,SP     ;Initialize stack pointer to location 600 hex
    MOV.W   #WDTPW+WDTHOLD,&WDTCTL ; Stop Watchdog Timer
```


Copyright © 2007–2018, Texas Instruments Incorporated

```
    BIS.B   #0xFF,&P1DIR    ;make all port one GPIO pins outputs
    BIS.B   #0x02,&P1SEL    ;p1.1 = peripheral module output for GSCLK signal.
    MOV.B   #0x00,&P1OUT    ;Initialize all outputs to zero
    BIC     #GIE,SR     ;Turn all interrupts off

;The following commands configure several clocks that are specific to the
;MSP430F427.  These commands configure an internal phase locked loop (PLL)
;to generate a 2MHz stem clock  from a 32.768KHz external crystal.

    MOV.B   #63,&SCFQCTL     ;set MCLK=64*ACLK or 2MHz
    MOV.B   #FN_2,&SCFI0     ;set DCO range
    BIS.B   #02,&FLL_CTL1    ;set ACLK/4
    BIC.B   #01,&FLL_CTL1    ;set ACLK/4
    BIS.B   #XCAP18PF,&FLL_CTL0  ;Set load capacitance for xtal

;The following is a delay loop.  This delay is needed to wait until the
;crystal oscillator is stable before continuing to execute code.  This
;step is MSP430 specific.

    MOV.W   #10000,R15   ;Initial value for a delay loop
Xtal_Wait
    DEC     R15          ;Delay for 32 kHz crystal to
    JNZ     Xtal_Wait    ;stabilize

;**********************************
;SET UP TIMER TO MAKE BLANK SIGNAL
;**********************************
;
;This section sets up a 16 bit timer with interrupt capability.  The timer
;is configured to count up to the decimal number 4097.  The timer clock signal
;is supplied by the system clock which is also the GSCLK.  The timer will count 4097
;GSCLKS and then initiate an interrupt.  The timer automatically clears to zero
;then starts to count again to 4097.
;The interrupt routine simply toggles the
;Parallel I/O pin that is connected to the BLANK signal of the TLC5941.
;The TLC5941 will receive 4096 GSCLKS and then get a pulse on the BLANK
;pin to reset the internal counters of the TLC5941
;
    MOV.W   #OUTMOD_3+CCIE,&CCTL0    ;CCR1 toggle/set
    MOV.W   #4097,&CCR0    ;load timer value
    MOV.W   #TASSEL_2+MC_1,&TACTL    ;SMCLK, up mode
    BIS.W   #GIE,SR     ;enable timer interrupt for BLANK signal
;since the grayscale clock run continuously, this timer is used to automatically
;send out BLANK signals to start new display frames.
;This method makes the grayscale clocking transparent to the rest of the
;firmware.

;**********************************
;SEND DC AND GS DATA TO TLC5941
;**********************************

    CALL     #DCOUT   ;call routine to move dot correction
                      ;data to TLC5941
    CALL     #GSOUT   ;call routine to move grayscale
                      ;data to TLC5941

mainloop
    NOP
    JMP     mainloop   ;infinite loop when done
```


Copyright © 2007–2018, Texas Instruments Incorporated

```
;-------------------------------------------------------------------------
;    SUBROUTINES
;-------------------------------------------------------------------------

;*************************************
;SEND GRAYSCALE DATA TO TLC5941
;*************************************
;
;shift out Grayscale data stored in RAM to TLC5941
;clocks out as MSB of channel 15 first then works down through bytes in memory
;shifts of 12 bits per channel for a total of 192 bits.
;
GSOUT

    MOV.W #16,R12     ;Register 12 is a loop counter.
                      ;Loop through 16 LEDs.
    MOV.W #Grayscale,R13 ;load register 13 with the starting address of the
                      ;grayscale data stored in RAM. R13 will point to the
                      ;byte to shift out.
    ADD.W #32,R13     ;Add 32 to the pointer so that R13 points to the byte after
                      ;the last byte in the grayscale table. The first instruction
                      ;in the following loop is a decrement instruction that will
                      ;make R13 point at the last byte in the table.
ltagsdt
    DEC.W R13       ;decrease the address pointer by one byte
    MOV.B #08,R9    ;Load register R9 with the number of bits to shift out
                    ;this value will be passed to the SHIFTOUT subroutine
                    ;R13 points to the MSB first in memory so there are 8
                    ;bits to shift out.
    MOV.W R13,R8    ;Copy address pointer to R8.The SHIFTOUT
                    ;routine uses R8 to point to the byte to shift out.
    CALL #SHIFTOUT  ;call subroutine to shift out data
    DEC.W R13       ;decrement pointer one byte to point to LSB byte
    MOV.B #0x04,R9  ;load R9 with the number of bits in the second byte
                    ;to shift out. Only four bits remain to be shifted
    MOV.W R13,R8    ;Copy address pointer to R8. The SHIFTOUT
                    ;routine uses R8 to point to the byte to shift out.

    CALL #SHIFTOUT  ;call subroutine to shift out data
    DEC.W R12       ;decrement loop counter by one
    JNZ ltadcdt     ;jump if not zero to continue looping

    BIS.B #XLAT,&P1OUT  ;set the I/O pin for XLATCH high to latch
                        ;serial data into the TLC5941
    BIC.B #XLAT,&P1OUT  ;set XLATCH back to zero

    BIS.B #SCLK,&P1OUT  ;set the I/O pin for SCLK high to
                        ;give SCLK one extra pulse after XLATCH
                        ;this is only required if the previous dat sent
                        ;to the TLC5941 was dot correction information
    BIS.B #SCLK,&P1OUT  ;set I/O pin for SCLK back to zero

    RET     ;return from subroutine

;*****************************************
;SEND DOT CORRECTION DATA TO TLC5941
;*****************************************
;
;
;shift out dot correction data stored in ram to TLC5941
```


Copyright © 2007–2018, Texas Instruments Incorporated

```
;clocks out as MSB of channel 15 first then works down through bytes in memory
;shifts 6 bits per LED channel or 96 bits total.
;This routine is very similar to the Grayscale routine.  The DCOUT and GSOUT
;routines could be combined together for some memory space savings by
;using some flags and conditional statements.
DCOUT

    BIS.B #MODE,&P1OUT  ;set I/O line that is tied to the MODE pin of
                        ;the TLC5941 pin to one to enter DC mode
    MOV.W #16,R12       ;Register 12 is a loop counter.  Loop through 16 LEDs. MOV.W
#Dot,R13
                        ;load register 13 with the starting address of the
                        ;dot correction data stored in RAM. R13 will point to the
                        ;byte to shift out
    ADD.W #16,R13       ;Add 16 to the pointer so that R13 points to the byte after
                        ;the last byte in the dot correction table.  The first
                        ;instruction in the following loop is a decrement instruction
                        ;that will make R13 point at the last byte in the table.
ltadcdt
    DEC.W R13           ;decrease the address pointer by one byte
    MOV.B #06,R9        ;Load register R9 with the number of bits to shift out
                        ;this value will be passed to the SHIFTOUT subroutine
                        ;there are 6 bits to shift out
    MOV.W R13,R8        ;Copy address pointer to R8.The SHIFTOUT
                        ;routine uses R8 to point to the byte to shift out.
    CALL #SHIFTOUT      ;call subroutine to shift out data
    DEC.W R12           ;decrement loop counter by one
    JNZ ltadcdt         ;jump if not zero to continue looping
    BIS.B #XLAT,&P1OUT  ;set the I/O pin for XLATCH high to latch
                        ;serial data into the TLC5941
    BIC.B #XLAT,&P1OUT  ;set XLATCH back to zero
    BIC.B #10h,&P1OUT   ;set the I/O line that is tied to the MODE pin of
                        ;the TLC5941 pin to zero to set back to
                        ;grayscale mode

    RET                 ;return from subroutine

;************************************
;    SHIFT DATA OUT ROUTINE
;************************************
;
;clock variable number of bits to TLC5941.
;clocks out as most significant bit first then works down through byte
;r8 = address of byte containing data to shift (word length)
;r9 = number of bits 1-8 to shift out (byte length)
;r10 used for temporary storage for shifting
;
;This routine shifts out a variable number of bits from a byte
;stored in the location pointed to by R8.  The data to be sent
;is stored left aligned to the most significant bit.
;For example, a 6 bit value of 0x3F would be stored as 0xFC
;with the two least significant bits set to zero.
;
;Registers R8 and R9 are used to pass parameters to this routine.
;Prior to calling this routine, R8 should contain the address of the
;byte of data to be shifted out.  R9 should contain the number of bits
;that need to be shifted out.
;R9 will be used as a loop counter in this routine.  The contents of R9
;will not be preserved when returning form this routine.
SHIFTOUT
```


Copyright © 2007–2018, Texas Instruments Incorporated

```
    MOV.B @r8,r10        ;move the byte stored in the memory location
                         ;pointed to by register 8 into register 10
ctbit
    BIC.B #SIN,&P1OUT    ;clear the bit controlling SIN to zero
    BIT.B #80h,R10       ;test state of the most significant bit
                         ;of the data remaining to be shifted out
                         ;since we shift the MSB first, the software
                         ;looks at the MSB of R10 to decide what data
                         ;to shift out.
    JZ sdateq0           ;if the bit to shift out is a zero then skip ahead
                         ;because SIN was previously set to zero
    BIS.B #SIN,&P1OUT    ;The data to shift out is a one so set the bit
                         ;controlling SIN to a one
sdateq0
    BIS.B #SCLK,&P1OUT   ;set the bit controlling SCLK to a one
    BIC.B #SCLK,&P1OUT   ;clear the bit controlling SCLK to a zero
                         ;the last two commands pulse the SCLK
                         ;signal which latches the data on SIN into
                         ;the shift register of the TLC5941
    RLA.B R10            ;shift the data stored in register 10
                         ;one bit left.The LSB of R10 will have a zero
                         ;shifted in.
    DEC.B R9             ;decrement number of bits to send
    JNZ ctbit            ;jump if not zero. continue until all bits are sent

    RET                  ;return from subroutine

;-------------------------------------------------------------
;    INTERRUPT ROUTINES
;-------------------------------------------------------------

;-------------------------------------------------------------
;ISR:   Interrupt Service Routine for Timer
;-------------------------------------------------------------
;
;timer interrupt routine
;when the timer count reaches 4097 it will issue an
;interrupt. The processor jumps to here to service the interrupt.
;This interrupt routine toggles the bit controlling the BLANK
;pin of the TLC5941. When the BLANK line of the TLC5941,
;it resets the internal grayscale counter and starts a new
;grayscale frame.
;
    BIS.B  #BLANK,&P1OUT   ;Toggle bit controlling BLANK
    BIC.B  #BLANK,&P1OUT   ;signal. First high then low.

    RETI                   ;Return from ISR

;-------------------------------------------------------------
;     Interrupt Vectors
;-------------------------------------------------------------
;
;this segment is used to define the interrupt vectors in memory
;
    ORG     0FFFEh    ; RESET Vector
    DW      RESET     ;

    ORG     0FFECh    ; Timer Interrupt Vector
    DW      ISR       ;
END
```


Copyright © 2007–2018, Texas Instruments Incorporated

4 Modified Blank Signal

The example firmware and flow charts rely on the BLANK signal being manually toggled during a software interrupt. Using an interrupt allows the firmware to know when the BLANK signal is toggled and the end of a frame has been reached. This is useful when the microprocessor and software need to synchronize data transfers to correspond with the end of a display frame. If frame synchronization is not required, then the interrupt can be removed from the code which will free up the interrupt routine for other software functions. The capture and compare functions of the timer can be used to generate the BLANK signal without the need for the interrupt routine. This requires the timer to be configured slightly different than what is shown in the example firmware. The following four lines of code replace the four lines of code listed under the "SET UP TIMER TO MAKE BLANK SIGNAL" header in the code listing.

```
MOV.W   #OUTMOD_3+CCIE,&CCTL0   ; CCR1 toggle/set
MOV.W   #4098,CCR0
MOV.W   #4097,CCR1
MOV.W   #TASSEL_2+MC_1,&TACTL   ; turn on PWM - SMCLK, up mode
```

This code configures the counter timer to automatically toggle the BLANK line once the counter reaches 4097. With this code, the interrupts can be left disabled if desired.

5 Driving Additional LEDs

The firmware is configured to drive a single TLC5941 with 16 LEDs. However, several TCL5941s can be cascaded or daisy-chained together to drive a large number of LEDs. The TLC5941 data sheet shows how to daisy-chain several TLC5941s together. The loop counters and RAM address calculations of the GSOUT and DCOUT subroutines can be changed to accommodate any number of daisy- chained TLC5941s. The only limitation is the speed of the serial communications versus the desired frame rate of the display and the amount of RAM to store the dot correction and grayscale data.

The firmware example requires little EEPROM space in the MSP430F427. The dot correction and grayscale data tables require 48 bytes of RAM per TLC5941. The MSP430F427 was selected for this example implementation because it has 1KB of RAM available to store the dot correction and grayscale data. The 1KB of RAM is enough to store data for up to 21 TLC5941s or 336 LEDs.

The example implementation uses a 2-MHz system clock. This frequency is generated by the MSP430F427 from an external 32.768-kHz quartz crystal. The selection of a 32.768-kHz crystal and a 2-MHz system clock allow for the possibility of the MSP430F427 to drive an optional LCD display. If an LCD display is not required, then the system clock speed could be increased to 8 MHz which is the highest capability of the MSP430F427. Increasing the clock speed would speed the firmware execution and thus speed the communications between the MSP430F427 and the TLC5941 and thus increase the frame rate.

Copyright © 2007–2018, Texas Instruments Incorporated

6 Schematics

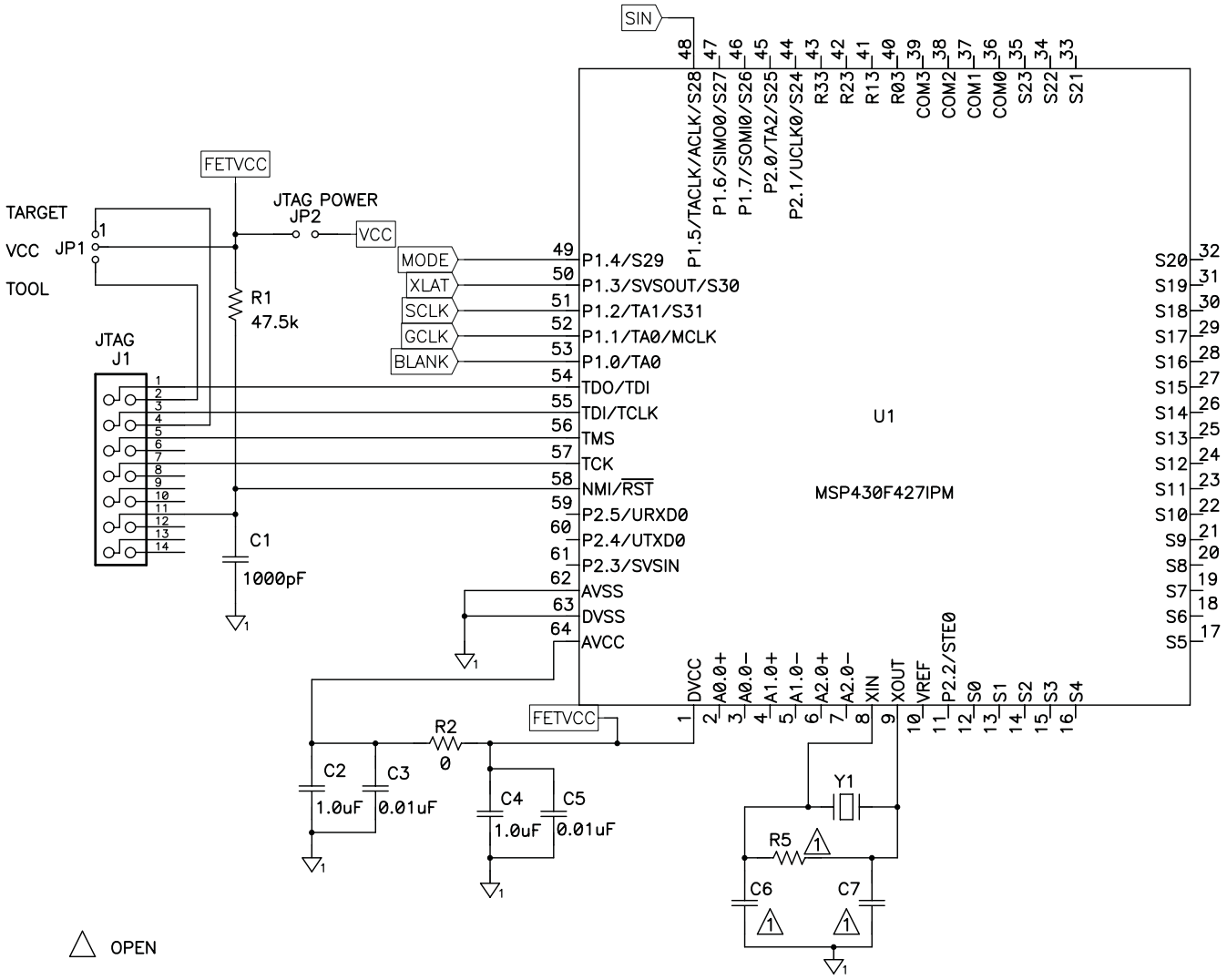

Figure 5. MSP430F427 Schematic

Copyright © 2007–2018, Texas Instruments Incorporated

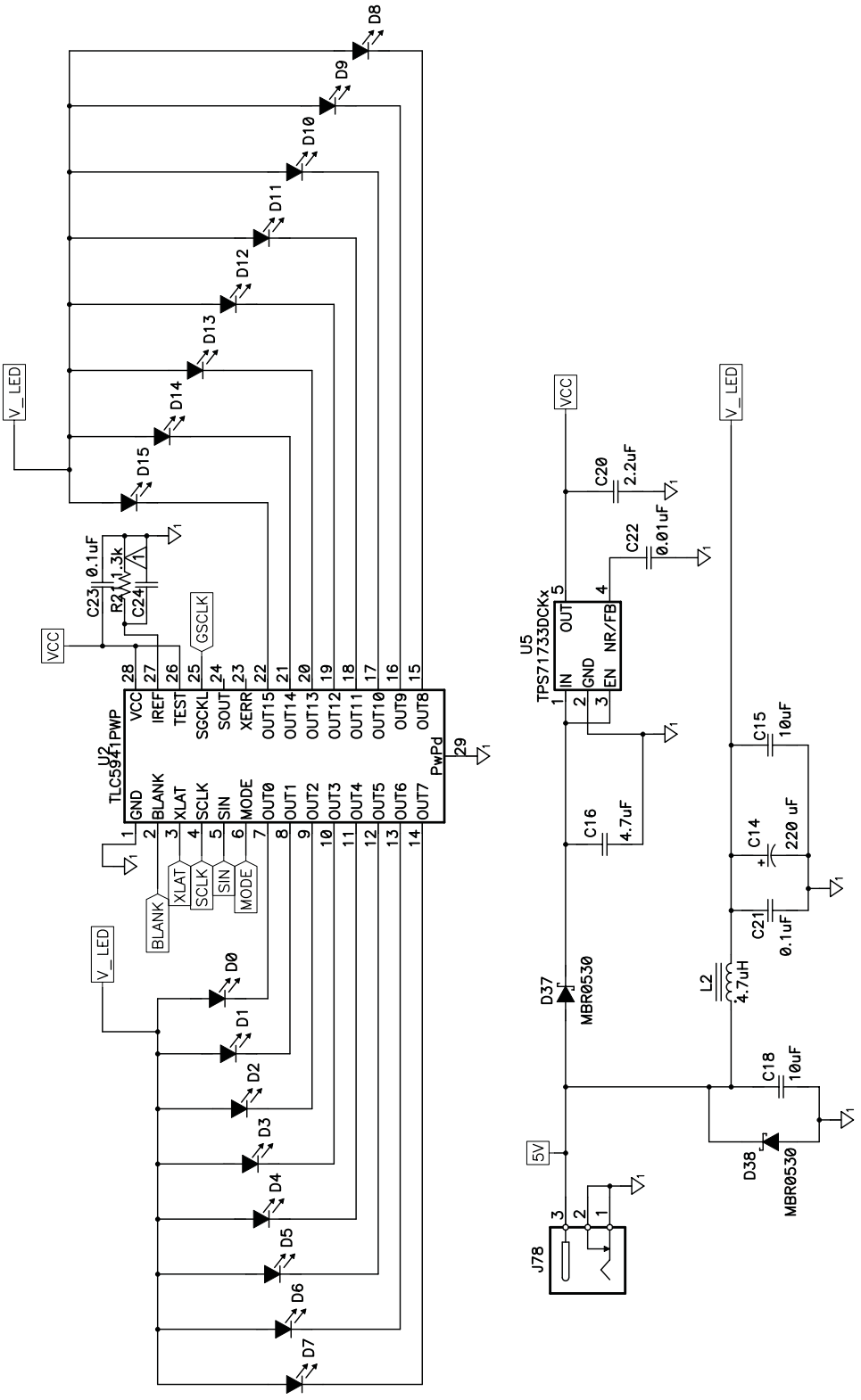

Figure 6. TLC5941 and Power Schematic

Copyright © 2007–2018, Texas Instruments Incorporated

7 Conclusion

The TLC5941 has a simple serial interface to accept grayscale and dot correction data for driving up to 16 LEDs. The simple serial interface can be driven by virtually any microcontroller, FPGA, or digital signal processor. This application report shows one hardware and firmware implementation to drive a display comprised of 16 LEDs.

8 References

1. *TLC5941 16 Channel LED Driver w/DOT Correction and Grayscale PWM Control* data sheet
2. *MSP430F42x Mixed-Signal Controllers* data sheet

Copyright © 2007–2018, Texas Instruments Incorporated

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from March 5, 2008 to September 27, 2018	Page
• Formatting and editorial changes throughout document	1

Copyright © 2007–2018, Texas Instruments Incorporated

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2018, Texas Instruments Incorporated