Application Report

SLVA655–June 2014

Hardware Design Considerations for a Wireless LED-Based Display Design

Srivatsa Raghunath

ABSTRACT

Scoreboards have different purposes with various features. Scoreboards are typically used for displaying scores of a game played between two teams. These games can be a timed or an untimed event. The application described in this application report creates a multi-sport scoreboard that is cost efficient, portable, and easy to use. The display values on the scoreboard are transmitted wirelessly.

This application report describes the selection of the following: an appropriate LED display, a controller system, a communication system, and software for building the wireless LED-based scoreboard. Different techniques are included to drive the LED display from a microcontroller as well as some test results. This application report is only for displaying numerals on the scoreboard but the same concept can be applied to display alphabets. A similar concept can be extended to large LED-display modules with multiple 16 × 16 or 24 × 24 matrices.

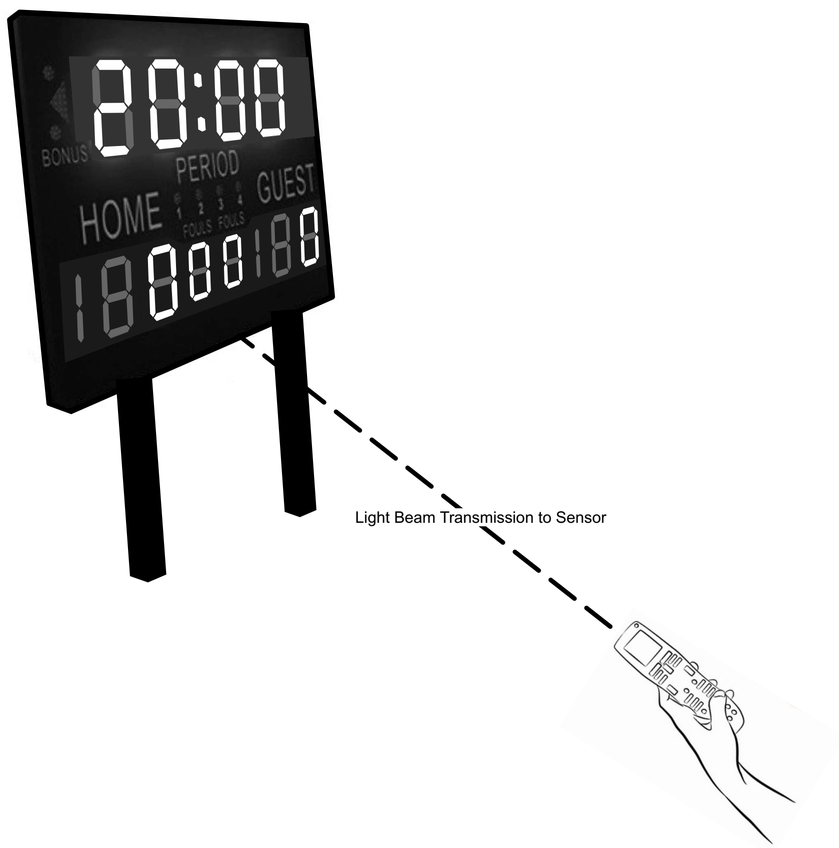


Figure 1. Wireless Scoreboard Concept

Contents

List of Figures

1 LED Display

There are several factors that must be considered when selecting an LED display. These factors include:

- Location of use; either inside or outside
- Type of message content to be displayed
- Communication method used
- Viewing distance of the audience
- Character size
- Pitch
- Viewing Angle
- Viewing Time

2 Selection Factors

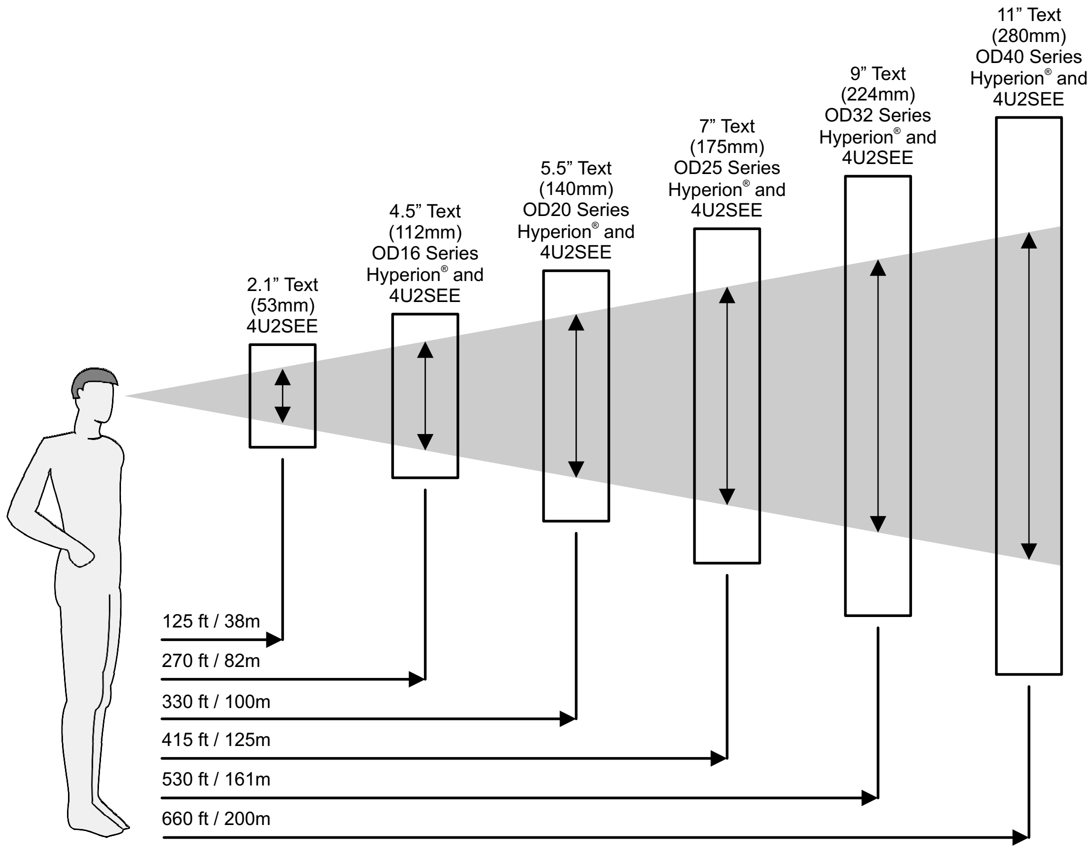
LED displays are specifically designed for either inside or outside use. Two main factors determine whether a display can be used inside or outside. The first factor is weather durability. Outside displays are designed to operate in extreme weather conditions while inside displays are not built to operate to the same environmental factors. Secondly, outside displays use different LEDs than inside displays. The LEDs used for outside displays are brighter to account for the varying lighting conditions. Inside displays do not have the same concerns and therefore are not as bright. A scoreboard that is used as an inside display for inside sporting events is selected for this application.

We need to evaluate how to use the sign by checking if we need only numeric or alpha numeric content to be displayed. Color display in the messages is required or monochrome is sufficient. Number of lines of messages, video messages, etc can also be looked into. These decisions also can increase or decrease the cost of the LED display. For example, a full color display is more expensive than a monochrome display. We've decided on monochrome, single line display of 7 x 24 LED matrix.

There are many different ways to communicate with the LED display. Understanding the requirements of the display in this regard will help in choosing the right technology for your application. Some installations may have specific requirements. We've chosen IR communication and the benefits of same are explained later in the note.

The distance between the sign and its viewers is the key factor in determining the best LED sign for any application. Considering two things about why it is crucial to choose a display based on where it will be located in relations to its audience. First, the distance from the viewer to the display will affect the minimum and maximum viewing distances. Second, the viewing distance also impacts the resolution or pitch

Figure 2 shows the approximate distance that specific sized text can be read.



The following viewing distances can be inferred for the listed text sizes:

- 6” text has an approximate viewing distance of 30 m.
- 8” text has an approximate viewing distance of 150 m
- 12” text as an approximate distance of 210 m

Figure 2. Distance Chart

The distance from the sign to the viewers is a factor of determining the actual physical size of the sign. The farther the viewer is from the sign, the larger the letters must be in order to be readable. Large letters can increase the size of the sign. The general rule of thumb in determining the appropriate size for the sign is 1 inch for every 50 feet of viewing distance. For example, if you were traveling 60 MPH on a highway, and the sign is 500 ft away, the text letters should be at least 10-in tall to be legible. Likewise, if you are standing 100 ft away from a sign at street level, the letters only need to be 2 in tall to be legible. For this application, the display board is 8 in to 10 in.

Longer distances require less resolution, and shorter distances require higher resolution. Resolution, also known as pitch, plays a key role in the cost of the sign. The higher the resolution, the higher the cost of the sign. Pitch is the distance (typically in millimeters, mm) between pixels. Pitch is always measured from the center of one diode to the center of another diode. The smaller the pitch number, the higher the resolution quality. Larger pitch numbers indicate a lower resolution. A pixel can be one single diode or can consist of a cluster of diodes running off the same circuit. Certain pitches are better for certain distances. For example, a sign next to a large road that is only viewed from far away only requires a low resolution because of the distance from which the sign is being viewed. A sign viewed from street level, such as one in front of a school, requires a higher resolution because the sign will be viewed from a shorter distance.

The typical viewing angle of electronic message centers is 90°. If the target audience is moving, a viewing time element is introduced. The size of characters, the viewing angle of the display, and the speed of the target audience all translate into one important element—time. The time it takes for the audience to read and comprehend a message before a new message is displayed must be considered. In general, 5 s is the length of time taken to read and comprehend ten characters of text.

The remaining two selection factors are for displays with moving audiences. Because this application is for a scoreboard, the audience is at rest and therefore typical viewing angles and viewing time are applied. A viewing angle of 150° (75° in each direction off of the center) means that as the target audience draws closer to the display, the display is still readable and increases viewing time.

3 LED Color Selection

Figure 3 shows the electromagnetic spectrum. A broad range of wavelengths with a visible-light spectrum at 380 nm to 750 nm and infrared light spectrum at 750 nm to 1 mm is used in this application.

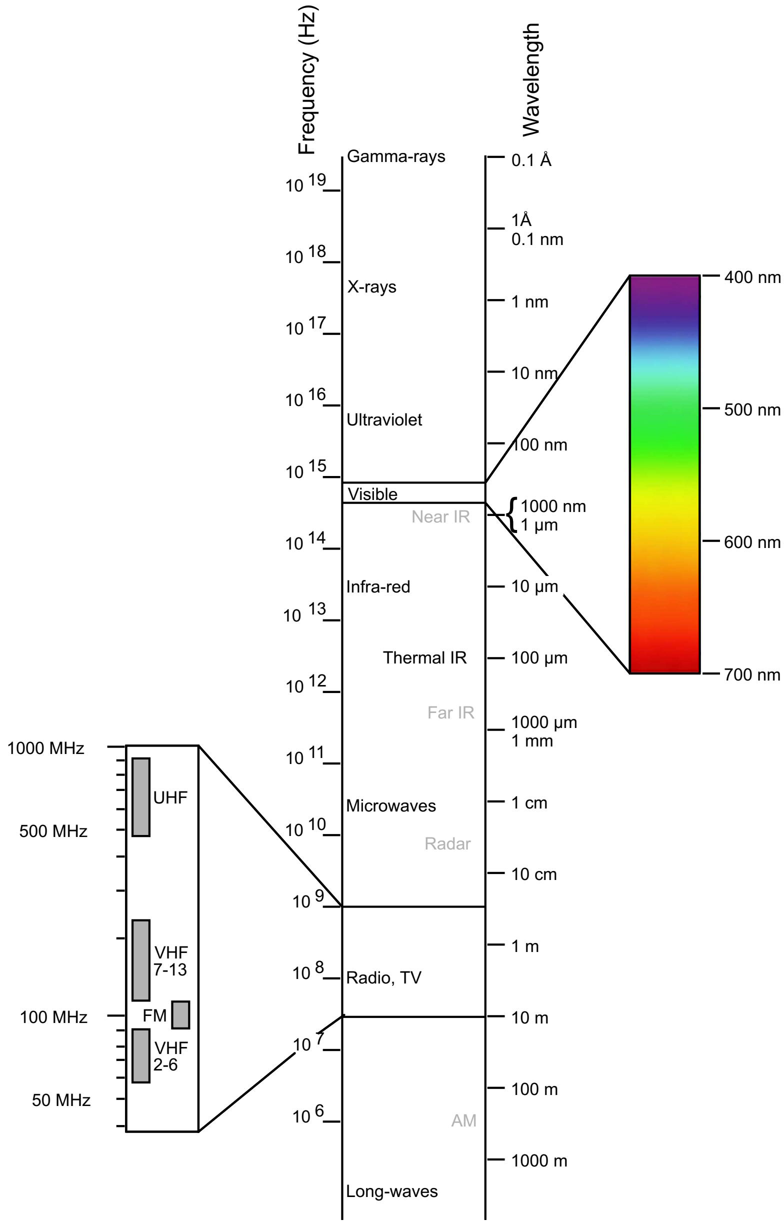


Figure 3. Electromagnetic Spectrum

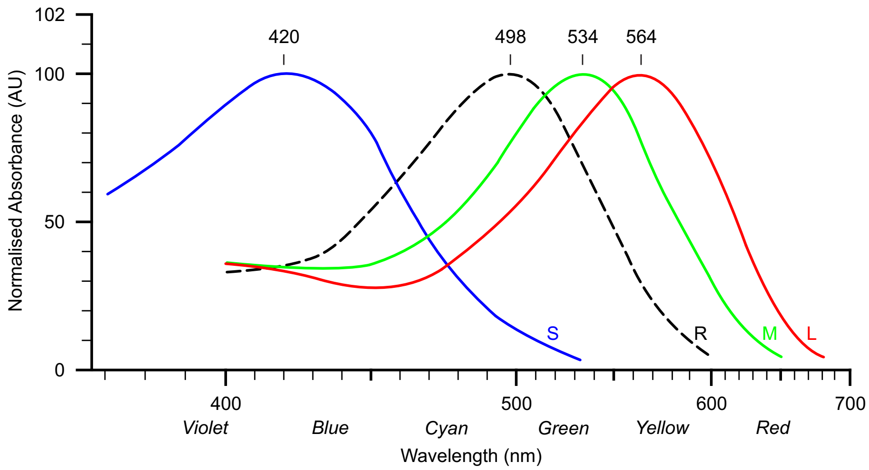


Figure 4. Absorbency

The color red has the broadest range of visible light (620 nm to 750 nm). The color yellow has the second broadest range of visible light. Human eyes see the colors red and yellow the best because of cones in the human retina. The scoreboard in this application displays yellow numbers and is clearly visible at a minimum of 30 ft.

4 Communication

Infrared (IR) light-based communication is selected because a computer is not used to program the sign. IR light-based communication is a simple, easy, and inexpensive method to program the sign. The distance between the remote and the display is less than 20 ft and the front of the sign are easily accessed. The message is changed infrequently and therefore storing all of the possible messages is not necessary.

For this application, an IR LED with a range of 850 nm to 950 nm and an acute beam angle was selected.

5 System Block Diagram

Figure 5 shows the complete system block diagram.

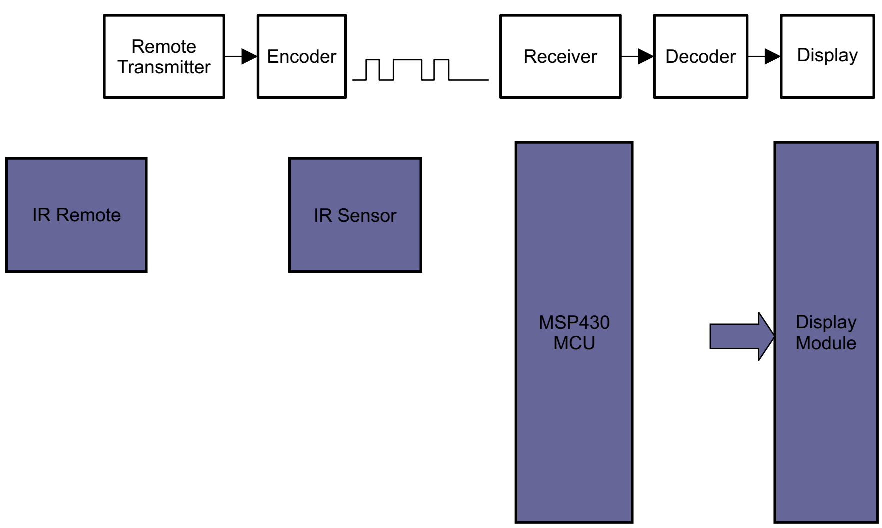


Figure 5. System Block Diagram

6 Hardware Design

6.1 Scoreboard Layout—Display Module

The board used in this application is a 7 × 24 LED display board. The LEDs are controlled by two SN74HC595 8-bit serial-in, serial or parallel-out shift register. This technique is one of the most basic techniques. Other techniques are discussed at the end of the application report.

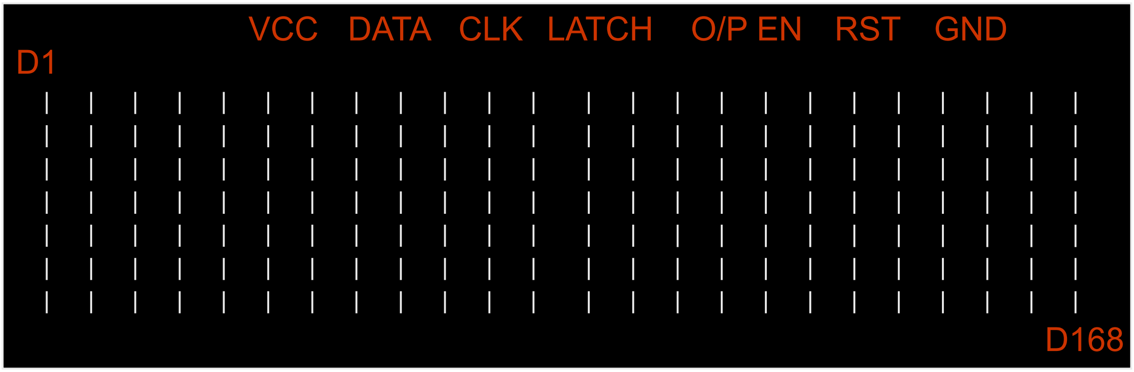


Figure 6. LED Display Board

6.2 IR Module

A standard TV remote is used as IR transmitter. The TSOP1738 device is used as a receiver for infrared remote control systems. Figure 7 shows the TSOP1738 block diagram.

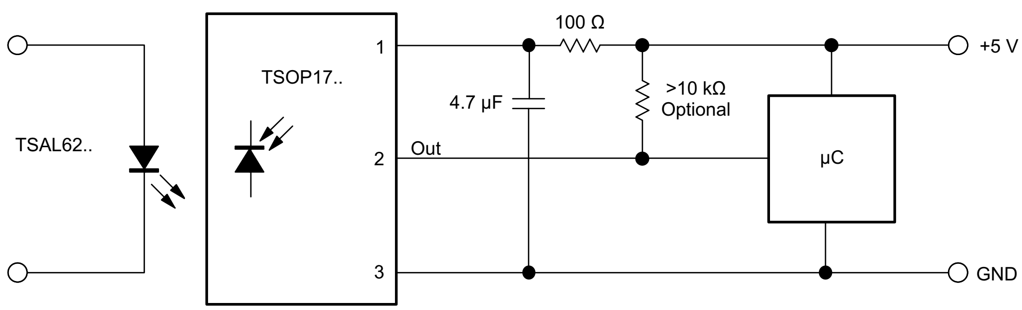


Figure 7. IR Module

Again, the IR module is a basic technique for wireless implementation. Other methods such as *Bluetooth*®, LPRF, and WiFi are discussed toward end of the application report.

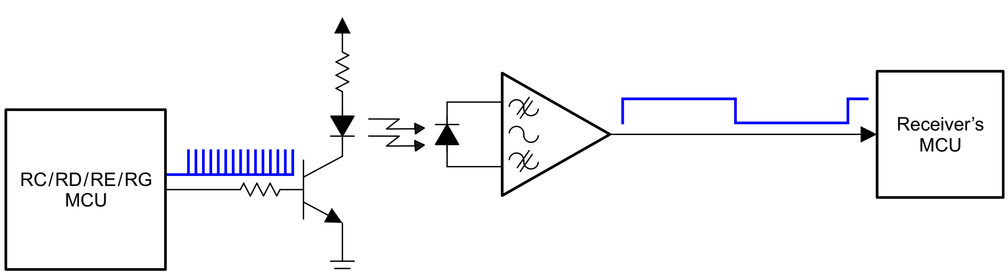


Figure 8. IR Modulation and Demodulation

Manchester encoding is used for transmission. Each bit consists of two half-bits that always have a different level. For example, a transition occurs from mark-to-space or space-to-mark. The polarity of the transition defines the logical level, for example mark-to-space denotes logic 1, space-to-mark denotes logic 0. See Figure 9.

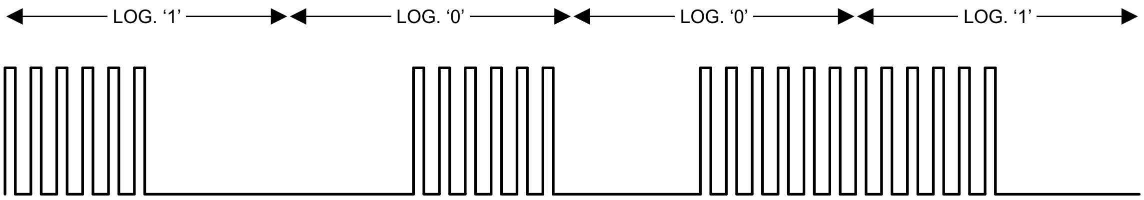


Figure 9. Manchester Encoding

Alternatively, see TI's CC2540 and CC2541 based BLE devices or the CC2530 and CC2500 based LPRF devices for remote control options. Complete reference designs are available for these remote control designs online at www.TI.com/TIDesigns.

6.3 Controller Module

The MSP430™ microcontroller (MCU) device is selected to interface between the IR communication module and the display module.Figure 10 shows the block diagram for the MSP430 device.

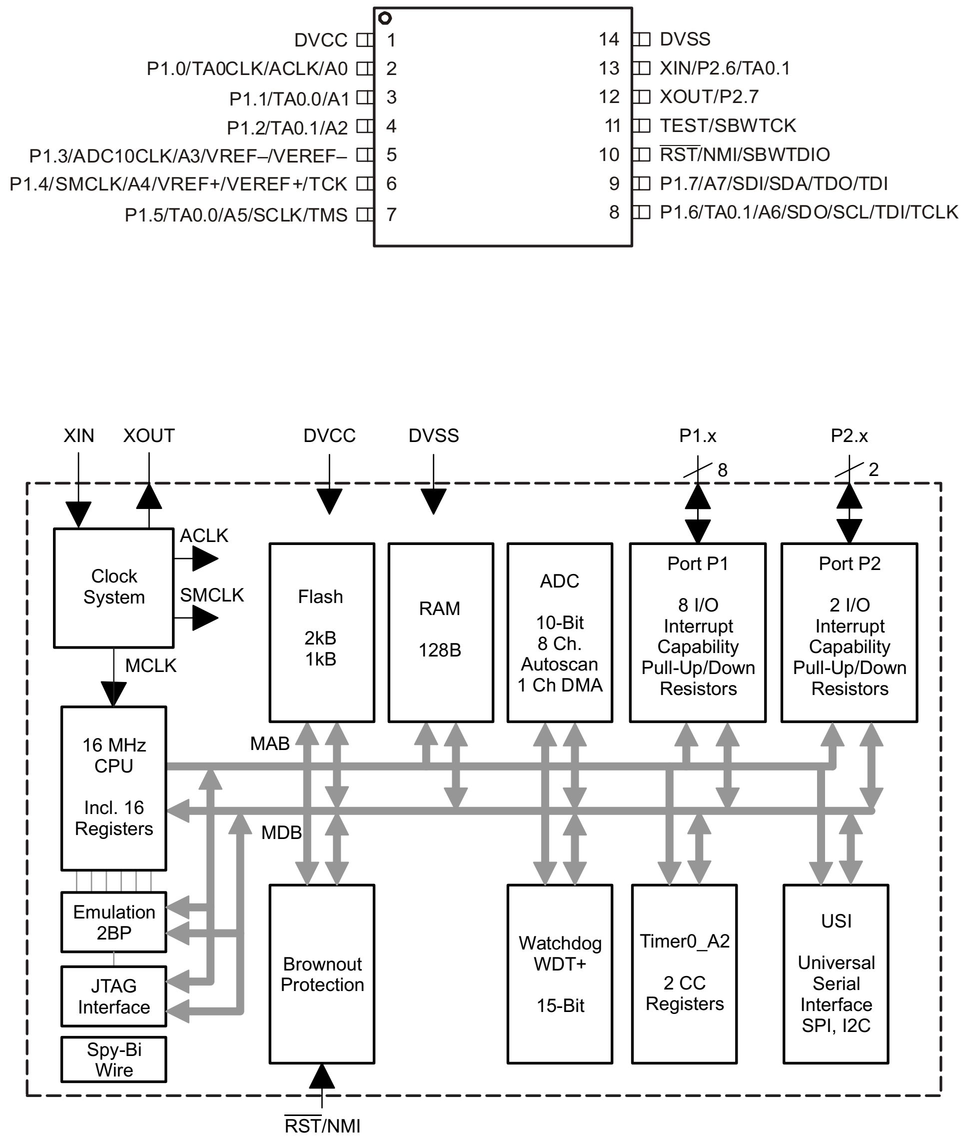


Figure 10. MSP430 Block Diagram and Pin Configuration

The MSP430 LaunchPad™ development tool is used for testing purposes.

Using a Shift Register

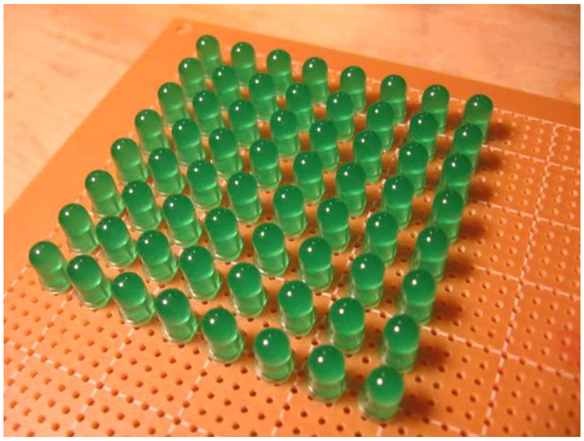


Figure 11. Typical LED Array

An LED array contains LEDs arranged in a grid of rows and columns. This matrix arrangement helps to control many LEDs with just a few output pins. The number of LEDs that can be controlled is the product of the number of columns and rows. The most common arrangement is an eight-by-eight matrix which has a total of 64 LEDs from 16 output pins.

The importance of the matrix configuration is that not all LEDs can be controlled at the same time. Only the LEDs located on one row or column can be turned on at a time . The wiring of the matrix determines if either a row or a column is controlled. Only one setting can be used at a time. However, the LEDs in each row or column can be turned on quickly enough so that all of the rows or columns appear to be on all at the same time. This quick turn on occurs through the persistence of vision of the eye and, if it occurs quickly enough, the matrix will not appear to flicker.

In this example, both ends of the LED matrix are under control. The LED turns on when the anode is high and the cathode is low. If the cathode on the two columns is low then two LEDs on the row light up.

NOTE: The top row of LEDs must be the source supply of the current for two LEDs where as the columns only have to connect to ground and sink only the current for one LED.

A matrix is created in two ways either by row scanning or column scanning. Data is supplied to one side of the matrix and a walking bit is applied to the adjacent side. A walking bit is a sequence where there is one different bit that changes position during each step of the sequence. Therefore a walking one is a line of bits that are all set to 0 except one. The position of the bit set to 1 changes at each step in the following example:

```
10000000
01000000
00100000
00010000
00001000
00000100
00000010
00000001
```

The data of this bit pattern defines which pattern the matrix shows when the LEDs are turned on as shown in Figure 12. The walking bit is a walking 1 on the rows and at each step the column is fed with data that defines the pattern of LEDs shown in Figure 12. In this example each row driver must supply the current for up to eight LEDs at any one time, where as the columns only have to sink the current for one LED.

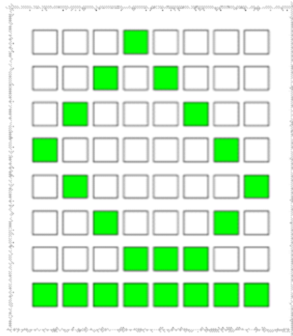


Figure 12. LED Array With Column Sink

The column scan is used in this example. The walking bit is a walking 0 and the data that defines the pattern is applied to the rows. In this example the row only sources the current for one LED at any one time where as the column must sink the current of up to eight LEDs.

Figure 13. LED Array With Row Sink

To drive a matrix with an MSP430 device, the application must include an external driver for the current-source or current-sink. The current through the LED must be limited. The simplest way of limiting the current is with a series resistor. However, the resistor must be located in the row that only takes the current for one LED (such as a column scan). If the resistor is not located in this row the brightness of the LEDs changes based on the number of LEDs that are on at any one time. The simplest arrangement is a column scan with an external current sink.

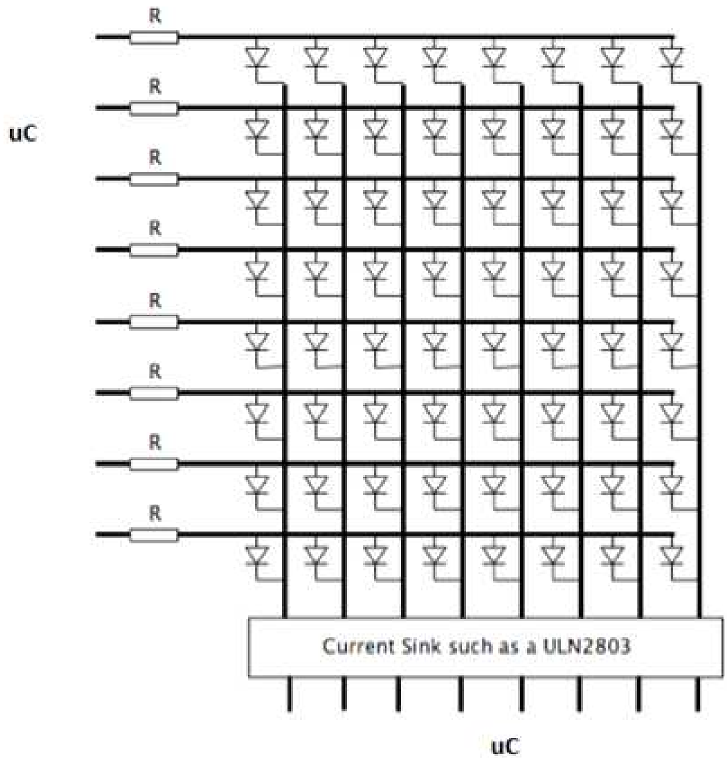


Figure 14. Column Scan

The column sink circuit uses TIs ULN2803 device which consists of eight Darlington drivers conveniently in one package. The collective current that this can be switched is about 650 mA at any one time which allows a maximum current of 80 mA per LED (see Equation 1). In practice, the LED current is limited by the amount of current can be sourced from the MSP430 row pins.

$$650 \text{ mA} / 8 = 80 \text{ mA}$$

where

- 650 mA is the collective current that the device can switch (1)

For row scan matrix an external current-source driver is used. The row-source circuit can use a high-sided driver such as TI's LMD18400 device.

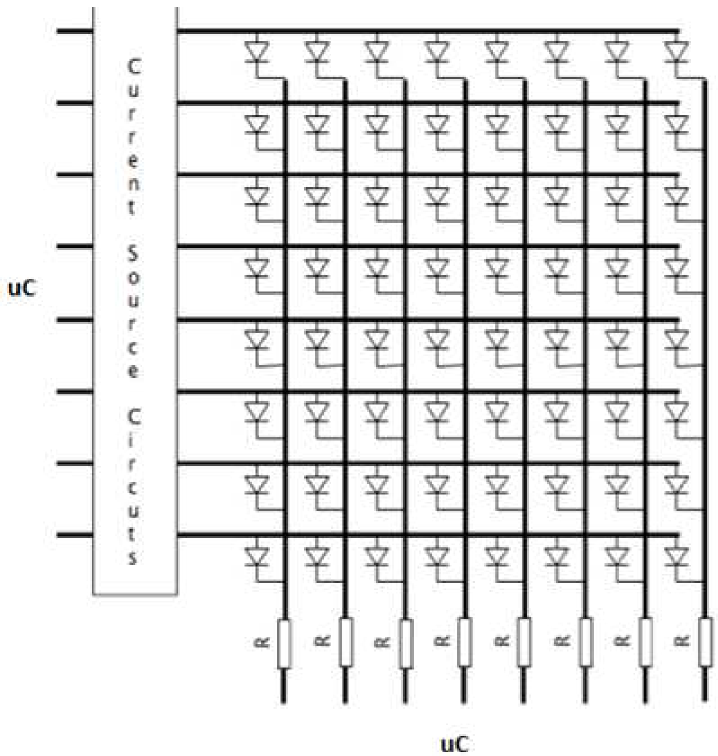
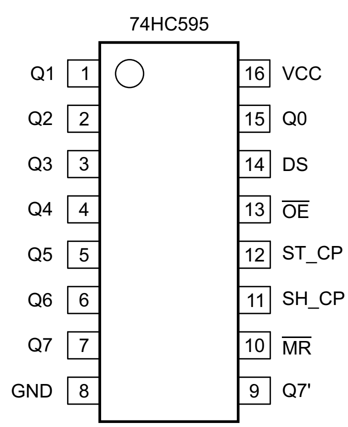


Figure 15. Row Scan

The availability of the pins in a microcontroller can be a constraint because of the variety of functions that the pins can perform. To lower the number of pins, a demultiplexer device such as the SN74LS42 decoder or the SN74LS145 decoder can be used. When using a demultiplexer device, only three rows are required to select which one, out of eight rows, goes low. If a low occurs on a row and one column then the row on each device can be common the enabled which reduces the number of rows used to control an eight-by-eight matrix down from sixteen rows to seven rows. Shift registers can also be used to drive the matrix which further reduces the pin count down to three rows if two rows are cascaded. This approach requires a longer time to refresh and typiclly a shift register can only source or sink enough current for one LED.

The SN74HC595 shift register has an 8-bit storage register and an 8-bit shift register. Data is written to the shift register serially and then latched onto the storage register. The storage register then controls eight output lines.



Pin 14 (DS) is the data pin. On some device data sheets, this pin is called SER.

Figure 16. Pin Configuration of the 74HC595

When pin 11 (SH_CP or SRCLK on some data sheets) goes from low to high the value of the DS pin is stored in the shift register and the existing values of the register are shifted to make room for the new bit.

Pin 12 (ST_CP or RCLK on some data sheets) is held low while data is written to the shift register. When the ST_CP pin goes high the values of the shift register are latched to the storage register which are then outputted to pins 1 through 7 (Q0 through Q7).

The timing diagram in Figure 17 shows how to set the Q0 through Q7 output pins to 11000011, assuming starting values of 00000000.

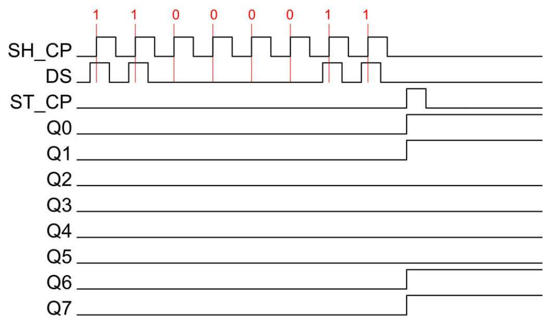


Figure 17. 74HC595 Timing Diagram

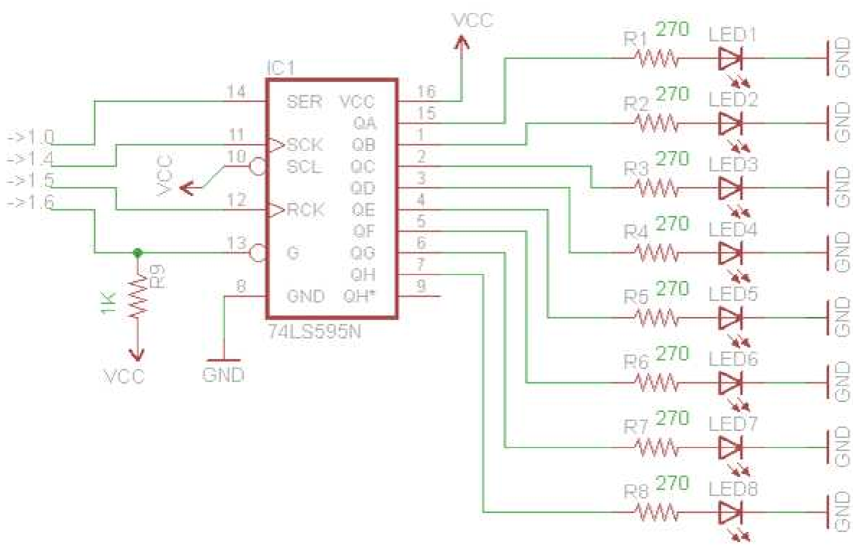


Figure 18. Shift Register Connections

Figure 19 shows the overall schematic of the application.

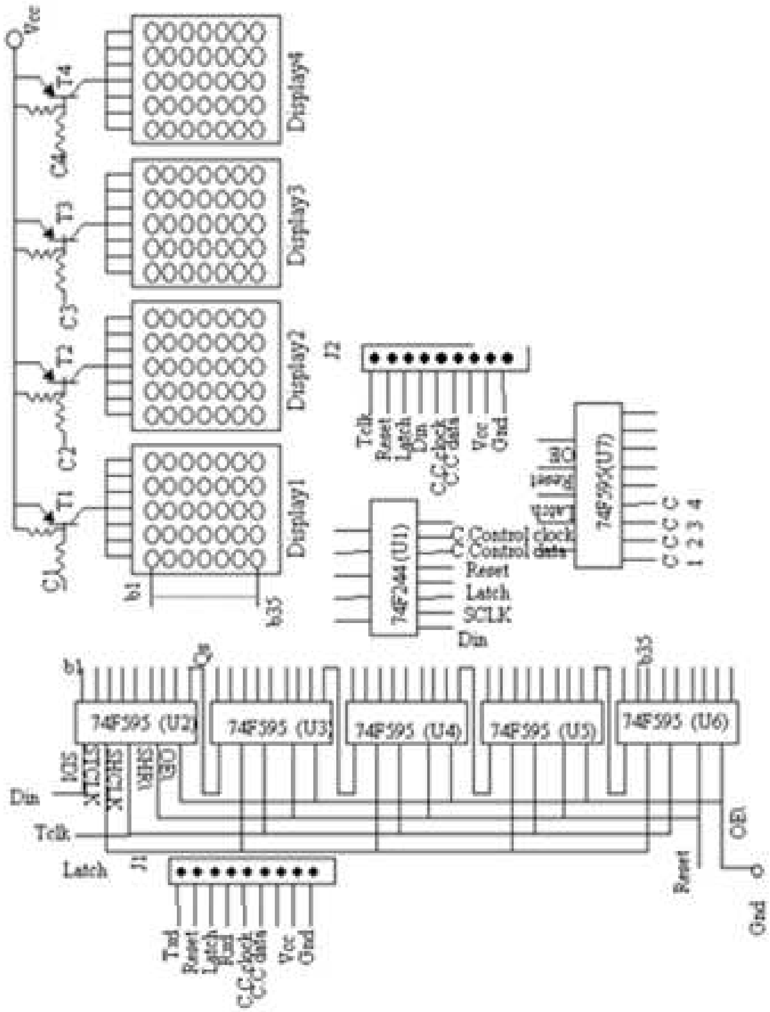


Figure 19. Schematic

The use of the TLC5940 (16 channel) device or TLC5947 (24 channel) device in the application can also simply the design.

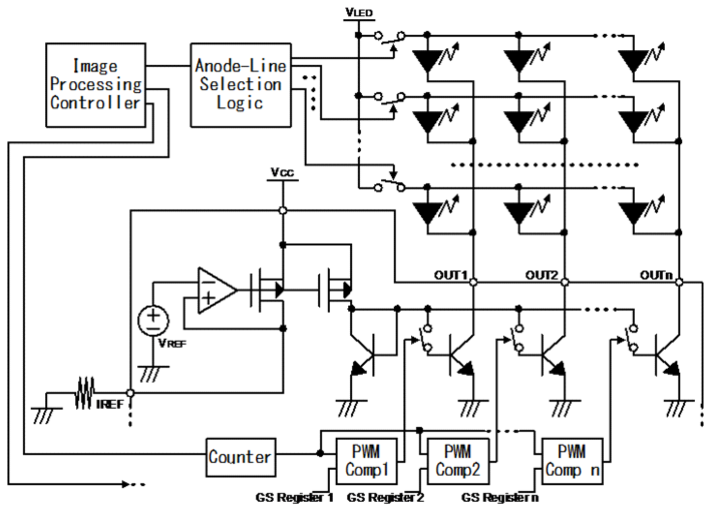


Figure 20. System Block Diagram with the TLC59xx

As shown in Figure 21, Figure 22, and Figure 23, the overall system is simplified further.

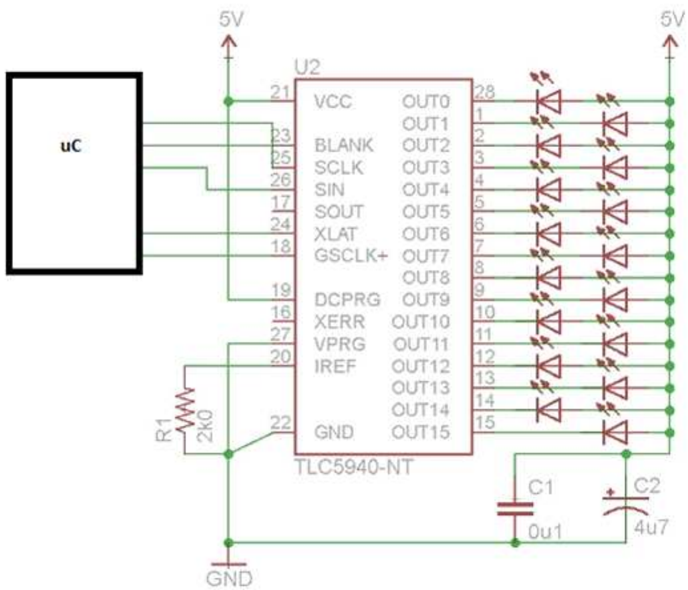


Figure 21. Single TLC5940 Interface

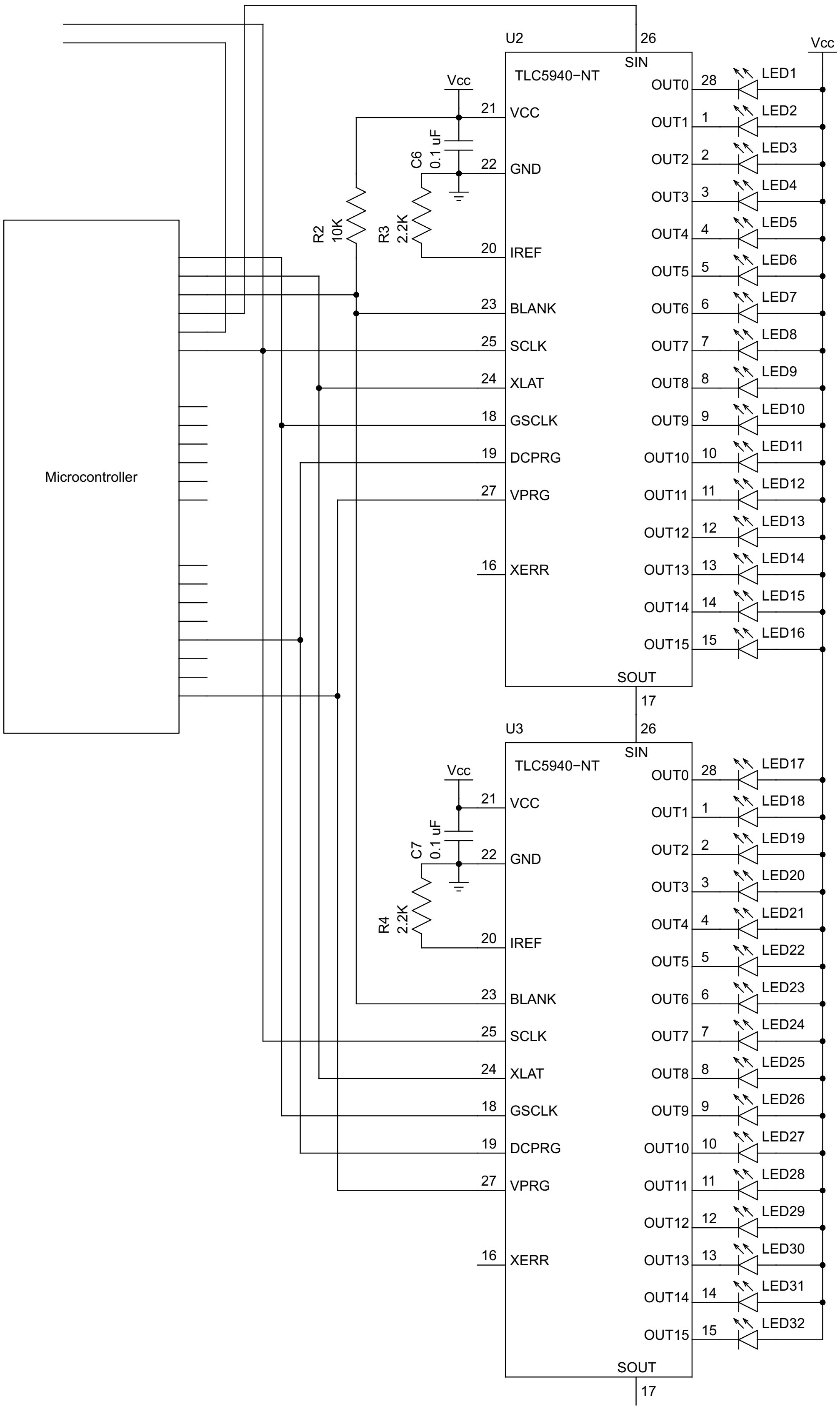


Figure 22. Multiple TLC5940 Interface

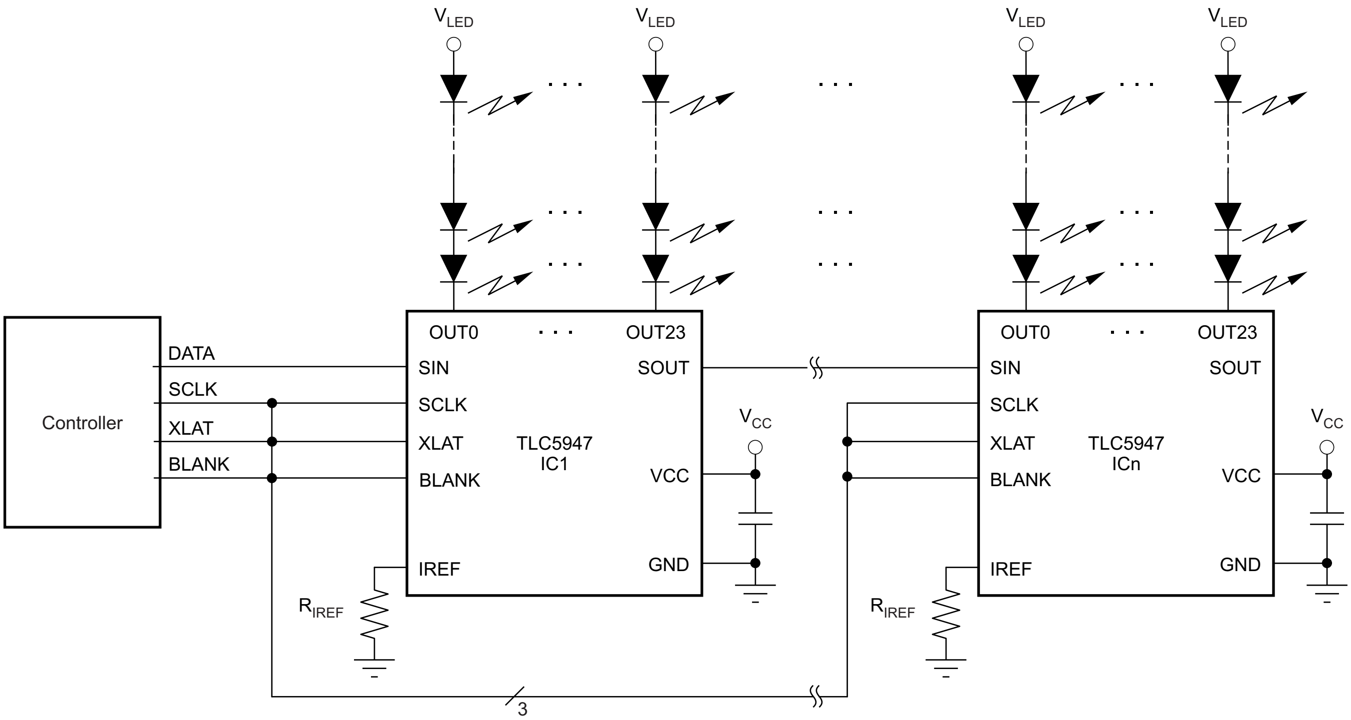


Figure 23. Further Simplification Using the TLC5947

7 Software

The development environment used in the application is Code Composer Studio 4 (CCS4). The complete sample program was written in assembly language.

After the assembly program is compiled and the build is complete, a .hex file is generated in the .exe folder in the project directory. Then the .hex is programmed to the MSP430 device.

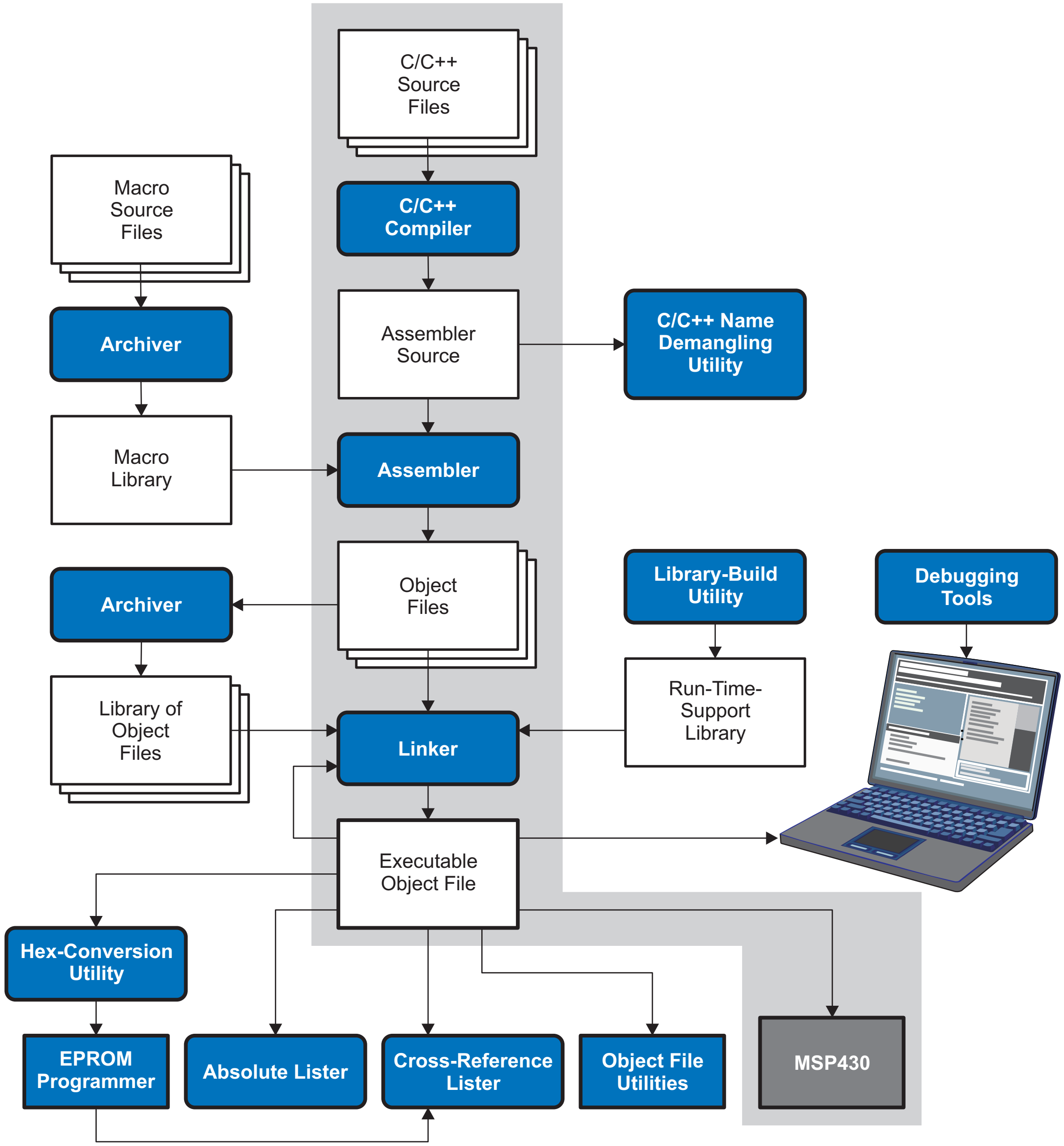


Figure 24. Overall Software-Development Flow

7.1 MSP430 Assembly program

The following code shows the complete MSP430 assembly program.

```
.include "msp430g2x31.inc"

.def  temp              = R16
.def  num_1             = R18
.def  num_2             = R19
.def  num_3             = R20
.def  num_4             = R21
.def coloum             = R22
.def row                = R23
.def temp_1             = R24
.def temp_2             = R25
.def temp_3             = R26
.def temp_4             = R27
.def data               = R28
.def buffer             = R15
.def bug                = R14
.def num_of_bits        = R13
.def threshold          = R12
.def code               = R11

//----using "Z" r31:30 memory pointer
.org 0
                jmp main
.org 0x0006
                jmp ir_interupt
//_________________________________________________________________________________________________

main:
ldi temp,LOW(RAMEND)
out spl,temp
ldi temp,high(RAMEND)
out sph,temp
//sei                    //enabling all interupts
ldi temp ,0x00                   //portb as input port
out ddrb ,temp
ldi temp,0x20                     //enabling INTR_2 intreupt
out gicr,temp
ldi temp,0x00                     //INTR_2 as -ve edge triggered interupt
out mcucsr,temp
ldi temp,0xff                     //portd as output port
out ddrd,temp
out ddra,temp                     //port a as output port for display
ldi num_1,0x00
ldi num_2,0x00
ldi num_3,0x00
ldi num_4,0x00
//_________________________________________________________________________________________________

again                     :cbi porta,3          //enable output buffer
                                 cbi porta,4            //reset all data
                                 call bit_delay
                                 sbi porta,4

                                 mov data,num_1
                                 call display_char
                                 //________________

                                 mov data,num_2
                                 call display_char
                                 //________________

                                 mov data,num_3
                                 call display_char
```

```
                                //________________

                                mov data,num_4
                                call display_char
                                //________________
                                ldi temp,0xff
                                out portd,temp
    wait_for_intr    :sbis pinb,2
                                rjmp outside_loop
                                rjmp wait_for_intr
//wait_for_intr    :rjmp wait_for_intr
    outside_loop     :call ir_interupt
                                jmp again


//__________________________________________________________________________________________________
_______


display_char    :ldi zh,high(d1)
                                ldi zl,low(d1)
                                lsl data                      //data X 2
                                lsl data                      //data X 4
                                lsl data                      //data X 8
                                add zl,data                   //calculating offset for lookup
table
                                ldi coloum,0x06
nxt_coloum              :
                                ldi row,0x08
                                lpm buffer,z+

nxt_bit                 :rol buffer
                                brcs send_1
                                cbi porta,0                  //data to be sent = 0
                                jmp latch_data
send_1                  :sbi porta,0                  //data to be sent = 1
latch_data              :
                                //________________________
                                cbi porta,1
                                call bit_delay
                                sbi porta,1
                                call bit_delay
                                cbi porta,1
                                //_________________________
                                cbi porta,2
                                call bit_delay
                                sbi porta,2
                                call bit_delay
                                cbi porta,2
                                //_________________________
                                dec row
                                brne nxt_bit
                                dec coloum
                                brne nxt_coloum
                                ret
//__________________________________________________________________________________________________
_________________
increment_team_1        :
                                ret
//__________________________________________________________________________________________________
_______
increment_team_2        :
                                ret
//__________________________________________________________________________________________________
_________________
```

```
ir_interupt                     :ldi temp,0x00            //disabling INTR_2 intreupt
                                out gicr,temp
                                ldi temp,4
                                out portd,temp
                                ldi temp,0x08
                                mov num_of_bits,temp
                                ldi temp,100
                                mov threshold,temp
                          l1    :sbis pinb,2
                                rjmp l1
                          l2    :sbic pinb,2
                                rjmp l2
                                ldi temp_1,0x00
                          l3    :sbic pinb,2
                                rjmp compare
                                inc temp_1
                                nop
                                nop
                                nop
                                nop
                                jmp l3
             compare            :
                                cp threshold,temp_1
                                brlo zero_detected         //branch if carry set
                                sec
                                jmp l_4
zero_detected       :clc
                          l_4    :rol code
                                dec num_of_bits
                                brne l2
//***********************************************
                                mov temp,code
                                cpi temp,0b11111110
                                brne check_2
                                //_____________key_1_detected
                                ldi temp,0x01
                                out portd,temp
//                   stop       :rjmp stop
                                cpi num_1,0x02
                                brne incr_1
                                cpi num_2,0x01
                                brne incr_1
                                vldi num_1,0x00
                                ldi num_2,0x00
                                ldi num_3,0x00
                                ldi num_4,0x00
                                jmp display
incr_1                          :cpi num_2,0x09
                                brne inc_num_2
                                ldi num_2,0x00
                                inc num_1
                                jmp display
inc_num_2                       :inc num_2
                                //__________________
                                jmp display
//############################################################################
check_2                         :cpi temp,0b01111110
                                brne check_3
                                //____________key_2_detected
                                ldi temp,0x02
                                out portd,temp
                                //__________________
                                cpi num_3,0x02
                                brne incr_2
                                cpi num_4,0x01
```

```
                                brne incr_2
                                ldi num_1,0x00
                                ldi num_2,0x00
                                ldi num_3,0x00
                                ldi num_4,0x00
                                jmp display
incr_2                          :cpi num_4,0x09
                                brne inc_num_4
                                ldi num_4,0x00
                                inc num_3
                                jmp display
inc_num_4                       :inc num_4
                                jmp display
//##############################################################################
check_3                         :cpi temp,0b10111110
                                brne display
                                //____________key_3_detected
                                ldi temp,0x03
                                out portd,temp
                                //__________________
                                ldi num_1,0x00
                                ldi num_2,0x00
                                ldi num_3,0x00
                                ldi num_4,0x00

         display:
                                call delay
                                ldi temp,0x05
                                out portd,temp
                                call delay
//                        l6    :jmp l6
//                              ldi temp,0x20               //enabling INTR_2 intreupt
//                              out gicr,temp
                                ret
//________________________________________________________________________________________________
________________
bit_delay                       :ldi temp_2,0x02
//bit_delay_1                   :ldi temp_3,50
//bit_delay_2                   :ldi temp_4,25
//bit_delay_3                   :dec temp_4
//                              brne bit_delay_3
//                              dec temp_3
//                              brne bit_delay_2
//                              dec temp_2
//                              brne bit_delay_1
                                ret
//________________________________________________________________________________________________
________________
delay                           :ldi temp_2,0x02
delay_1                         :ldi temp_3,0xff
delay_2                         :ldi temp_4,0xff
delay_3                         :dec temp_4
                                brne delay_3
                                dec temp_3
                                brne delay_2
                                dec temp_2
                                brne delay_1
                                ret
.org 0x200
d:          .db
0x82,0x7c,0x7c,0x7c,0x82,0xff,0xff,0xff,0xdc,0xbc,0x00,0xfc,0xfc,0xff,0xff,0xff,0x7c,0x78,0x74,0x6
c,0x1c,0xff,0xff,0xff,0x6c,0x6c,0x6c,0x6c,0x00,0xff,0xff,0xff,0x0e,0xee,0xee,0xee,0x00,0xff,0xff,0
xff,0x0c,0x6c,0x6c,0x6c,0x60,0xff,0xff,0xff,0x00,0x6c,0x6c,0x6c,0x60,0xff,0xff,0xff,0x6c,0x6a,0x66
,0x6e,0x0e,0xff,0xff,0xff,0x00,0x6c,0x6c,0x6c,0x00,0xff,0xff,0xff,0x0c,0x6c,0x6c,0x6c,0x00,0xff,0x
ff,0xff
//d:     .db
```

```
0x82,0x7c,0x7c,0x7c,0x82,0xff,0xff,0xff,0x7f,0x7e,0x00,0x7a,0x7f,0xff,0xff,0xff,0x72,0x6c,0x5c,0x3
c,0x7a,0xff,0xff,0xff,0x82,0x8c,0x8c,0x8c,0xaa,0xff,0xff,0xff,0xee,0xee,0x00,0xea,0xe6,0xff,0xff,0
xff,0x9a,0x6c,0x6c,0x6c,0xb2,0xff,0xff,0xff,0xf2,0xec,0x1c,0xfc,0xfa,0xff,0xff,0xff,0x9a,0x6c,0x6c
,0x6c,0x92,0xff,0xff,0xff,0x92,0x8c,0x8c,0x8c,0x92,0xff,0xff,0xff,0x82,0x8c,0x8c,0x8c,0xb2,0xff,0x
ff,0xff
```

The following code in C, which uses the TLC5940 device, is similar to the previous code.

```
/*
 * TLC5940 Demo for MSP430
 * Only PWM functionality is demonstrated
 * Dot correction function is not used
 * Rotates a pattern of decreasing light intensity on LEDs - OUT0..OUT15
 * To pause, press and hold the S2 button on the launchpad
 * Launchpad LED1 stays on, LeED2 flashes on every PWM cycle
 */
#include <msp430g2231.h>
// Pin mapping
#define GSCLK_PIN    BIT7    // to TLC5940 pin 18
#define SIN_PIN      BIT1    // to TLC5940 pin 26
#define SCLK_PIN     BIT2    // to TLC5940 pin 25
#define BLANK_PIN    BIT4    // to TLC5940 pin 23
#define XLAT_PIN     BIT5    // to TLC5940 pin 24
#define LED1_PIN     BIT0    // Launchpad LED1 (Red)
#define LED2_PIN     BIT6    // Launchpad LED2 (Green)
#define S2_PIN       BIT3    // Launchpad S2 button
// Number of daisy-chained TLC5940 ICs
#define TLC5940_N    1
// Useful macros
#define setHigh(n)   (P1OUT |= n)
#define setLow(n     (P1OUT &= ~n)
#define pulse(n){ setHigh(n); setLow(n); }
// PWM cycles to wait before rotating gsData
#define DELAY 3
// Grayscale values for each channel in reverse order
unsigned int gsData[] = {
    0,          // Channel 15
    0,          // Channel 14
    0,          // Channel 13
    0,          // Channel 12
    BIT1,       // Channel 11
    BIT2,       // Channel 10
    BIT3,       // Channel 9
    BIT4,       // Channel 8
    BIT5,       // Channel 7
    BIT6,       // Channel 6
    BIT7,       // Channel 5
    BIT8,       // Channel 4
    BIT9,       // Channel 3
    BITA,       // Channel 2
    BITB,       // Channel 1
    BITB | BITA | BIT9 | BIT8 | BIT7 | BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0 // Channel 0
} ;
// Update gs during PWM cycle?
unsigned char gsUpdateFlag=1;
void init(void)
{
    P1DIR |= GSCLK_PIN | SIN_PIN | SCLK_PIN | BLANK_PIN | XLAT_PIN;               // Set output

    // Init output pins
    setLow(GSCLK_PIN);
    setLow(SCLK_PIN);
    setLow(XLAT_PIN);
    setLow(BLANK_PIN);
}
void GS(void)
```

```
{
    unsigned int Data_Counter = 0, GSCLK_Counter;
    setLow(BLANK_PIN);
    for (GSCLK_Counter = 0; GSCLK_Counter < 4096; GSCLK_Counter++)
    {
        if (gsUpdateFlag && (! (Data_Counter > TLC5940_N * 192 -1)))
        {
            // Check and load next data bit, MSB first, 12 bits per value
            if (gsData[Data_Counter / 12] & (BITB » ( Data_Counter % 12)))
                    setHigh(SIN_PIN);
            else
                    setLow(SIN_PIN);
            pulse(SCLK_PIN);
            Data_Counter++;
            P1OUT |= LED2_PIN;         // LED2 on
        }
         else
            P1OUT &= ~LED2_PIN;        // LED2 off
         pulse(GSCLK_PIN);
    }
    setHigh(BLANK_PIN);
    pulse(XLAT_PIN);
    gsUpdateFlag = 0;                  // PWM update finished, reset gsUpdate flag
}
void main(void)
{
    int i, t;

    WDTCTL = WDTPW + WDTHOLD;                                      // Disable watchdog timer
    // Set DCO to ~15.25 MHz as per datasheet
    // Does not use calibration values
    BCSCTL1 = RSEL0 | RSEL1 | RSEL2 | RSEL3;                       // RSELx = 15
    DCOCTL = DCO0 | DCO1;                                          // DCOx = 3, MODx = 0
    init();
    P1DIR |= LED1_PIN | LED2_PIN;                                  // Enable Launchpad LEDs for
output
    P1OUT |= LED1_PIN;                                             // LED1 on
    P1OUT &= ~LED2_PIN;                                            // LED2 off
 for (;<imgsrc='http://forum.43oh.com/public/style_emoticons/<#EMO_DIR#>/#>/icon_e_wink.gif'
class='bbc_emoticon' alt=';)' /> // Loop forever
    {
        for (i=0; i < DELAY; i++)
                            GS();
        if (P1IN & S2_PIN)                                          // If S2 not pressed
        {
            // Rotate gsData
            t = gsData[0];
            for (i = 0; i < 15; i++)
                            gsData[i] = gsData[i+1];
            gsData[15] = t;
            gsUpdateFlag = 1;                                        // Update gs on next PWM
cycle
        }
    }
}
```

7.1.1 Design Results

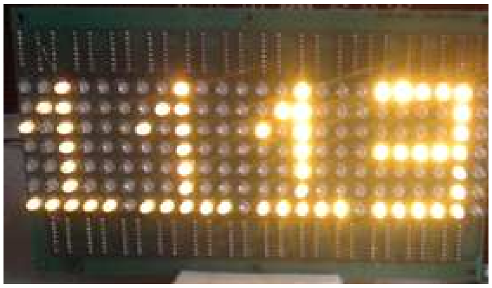


Figure 25. Scorboard Display Results

8 Conclusion

A 7-×-24 LED matrix display and an IR receiver were successfully interfaced to the MSP430 device. The resulting scoreboard design is a low-cost wireless application.

9 References

1. Texas Instruments Power Management website for LED Drivers and LED Lighting Design, www.ti.com/led
2. *LED Display System Module Using Cascading TLC5940 Devices*, SLVA242
3. *LED Driver – Paralleled Outputs Provide High-Current Outputs*, SLVA253
4. *TLC5940 One-Wire Control – Eliminating Microprocessor Control for Integrated LED Driver*, SLVA259
5. MSP430 Assembly Language Tools v 4.3 User's Guide, SLAU131

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265