ECE 241 **LED Cube Interface**

To interface to an LED cube we will be driving 64 separate LED's, 4x4x4. This means that we will be unable to hook each LED to a pin on the micro, thus we need to work on how we can turn on each of the LED's, at least for a short while, using as few pins as possible. First we set the LED's up as a grid, as shown here.

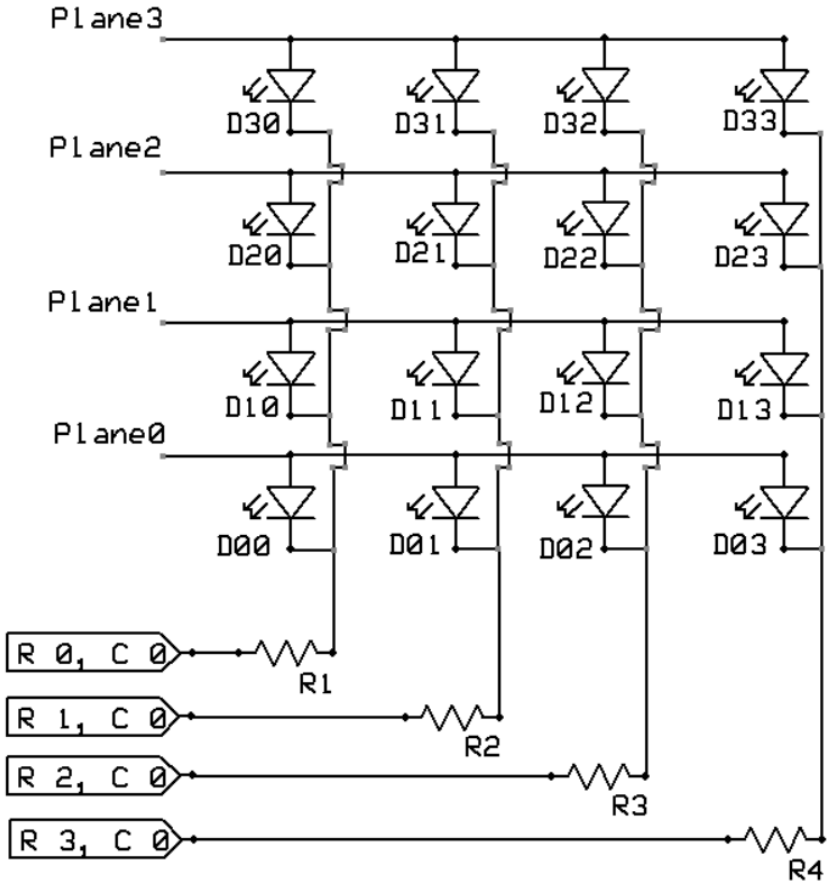

Figure 1. LED's in a Grid, Column 0.

In this grid, we see that if we were to only activate (set high) one of the “Planes”, then the lines along the bottom would be able to go low and turn on the LED's in a plane. For example if Plane3 was set high (5 volts) and R0, R1, R2 and R3 were in the pattern (5v, 0v, 0v, 5v), then there will be voltage across LED's D31 and D32 and only these two LED's will light up.

To make each LED appear to be on, if we want it on, we will need to work our way through each plane. In other words we will have a timer that fires at 4 milliseconds and it will 1) Turn off the planes (all set to 0 v), 2) Set R0, R1, R2, and R3 to the next desired values, then 3) turn on the next plane. But if we were to do a 4x4x4 cube, we would have 16 rows and columns and four planes or 20 pins. This number of pins is still more than we have on our Arduino nano.

Thus we will need to use another approach to allow us to get the data out to the rows and columns. For this a good way to do this is to use the SPI interface and connect it to a shift register. The circuit for which is shown in Figure 2. The square units inside the shift register are called flip-flops (official name

is a bistable multivibrator, but no one calls them that) and they basically hold an single bit.

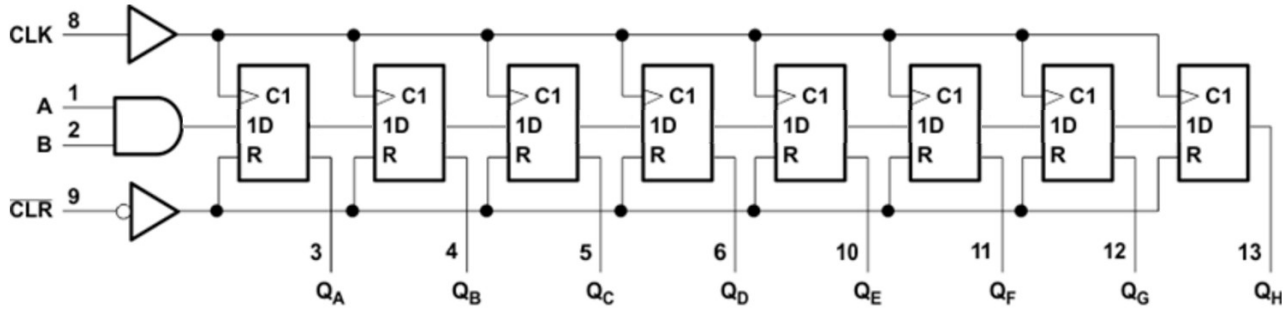

Figure 2. Shift Register Circuit (74164)

Often the best way to understand a system of this type is to look at the timing diagram, Figure 3. In this we can see that at the rising edge of CLK, the value of A and B will be set into the first FF (Qa) and the valu of Qa will move into Qb.

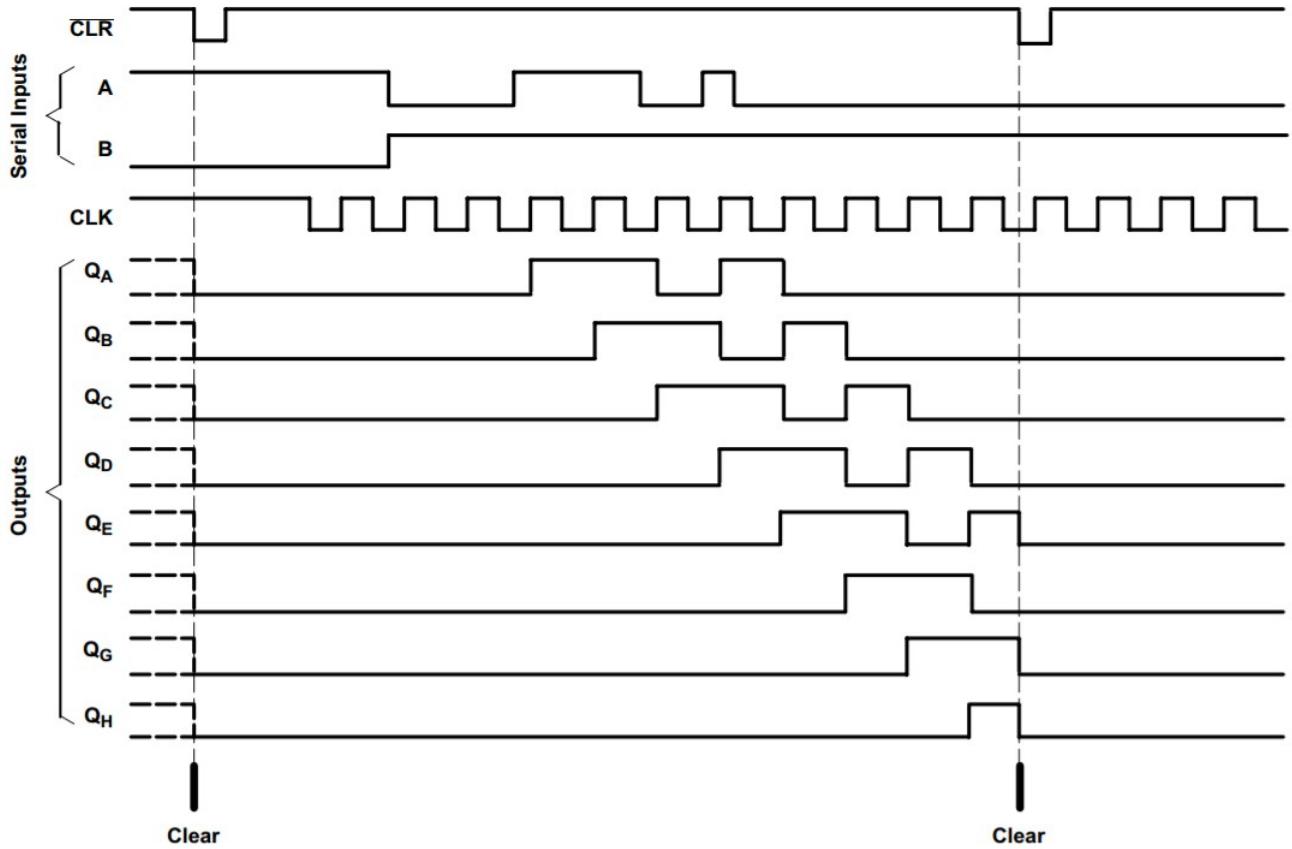

Figure 3. SN74HC164 Example Timing Diagram

The circuit to be used to hold the 16 values for the rows and columns of the LED Cube is shown in

Figure 4. In this schematic we can see that two of the 74164 shift registers is used in series, thus holding the 16 bits.

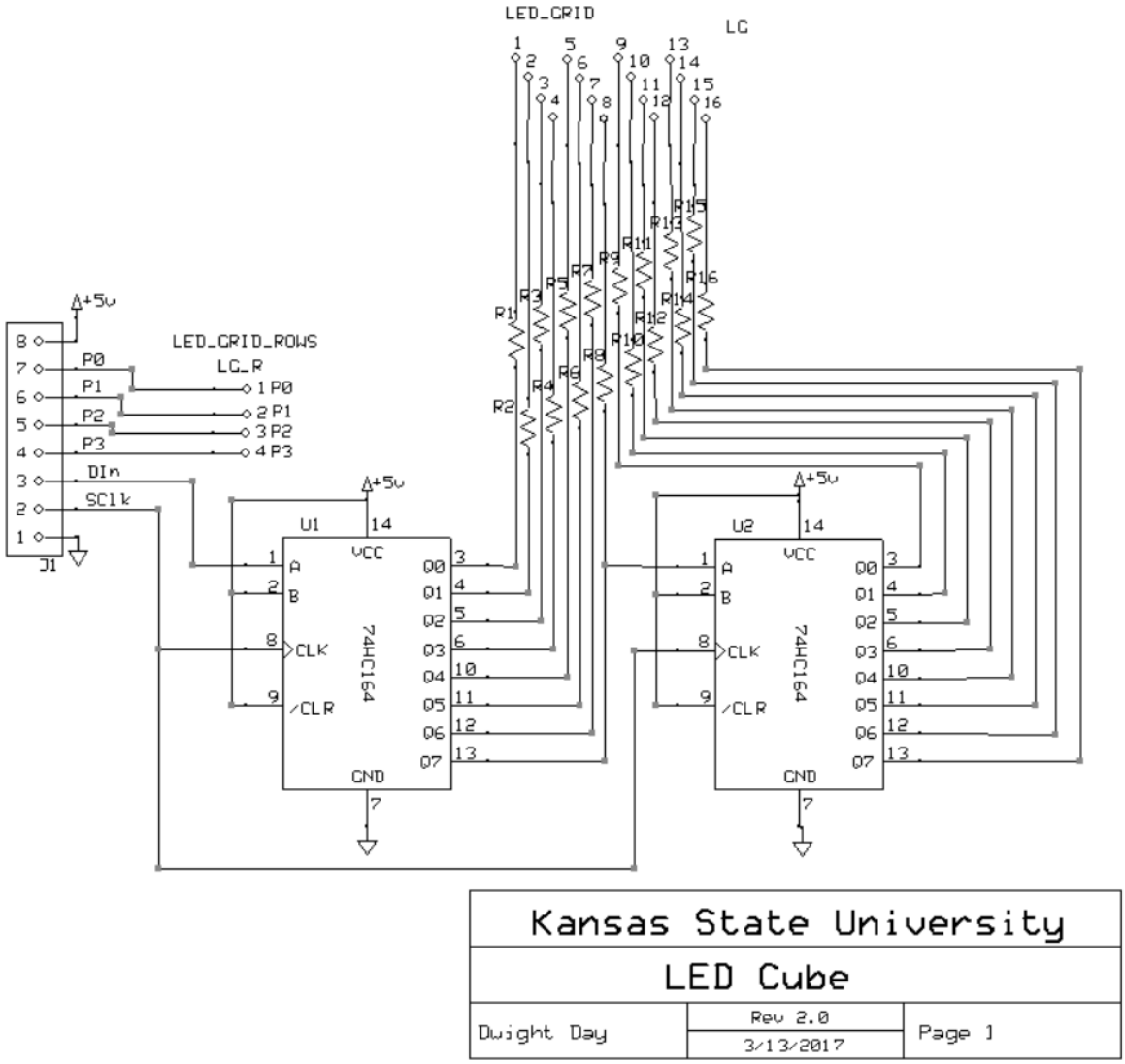

Figure 4. Schematic for LED Cube Base Board.

Appendix A: Code to Support LED Cube.

```
#ifndef LedCubeData_H
#define LedCubeData_H

// Establish constants for LedCube
#define LEDCUBE_NUMBER_PLANES  4
#define LEDCUBE_NUMBER_ROWS    4
#define LEDCUBE_NUMBER_COLUMNS 4

// Establish data for LedCube
// Variable representing led's state (on/off => 0/1)
// for each plane.
int LedCube_Planes[LEDCUBE_NUMBER_PLANES] = { 0 };

// variable representing plane that is currently on.
int LedCube_CurrentPlane = 0;

// Support routine to move to next plane, returning
// the bit pattern for the next plane to be displayed.
int LedCube_NextPlane()
{
        // Move to next plane.
        LedCube_CurrentPlane++;
        // Check for wrap.
        if (LedCube_CurrentPlane >= LEDCUBE_NUMBER_PLANES)
                LedCube_CurrentPlane = 0;

        // Send back the current bit pattern.
        return LedCube_Planes[LedCube_CurrentPlane];

} // End of LedCube_NextPlane

// Initialization routine, sets all leds to off.
void LedCube_ClearData()
{
        // Loop through the planes.
        for (int k = 0; k < LEDCUBE_NUMBER_PLANES; k++)
                LedCube_Planes[k] = 0xffff;
        // Note a bit being 1 means off.

} // End of LedCube_ClearData

// This function turns on the led at r,c,p
// (row, column, plane.
void LedCube_SetLed(int r, int c, int p)
{
        // make copy of r,c,p
        int plane = p, row = r, col = c;

        // Error Check r,c,p
        if (plane < 0) plane = 0;
        if (plane >= LEDCUBE_NUMBER_PLANES)
                plane = LEDCUBE_NUMBER_PLANES - 1;
        if (row < 0) row = 0;
        if (row >= LEDCUBE_NUMBER_ROWS)
                row = LEDCUBE_NUMBER_ROWS - 1;
        if (col < 0) col = 0;
        if (col >= LEDCUBE_NUMBER_COLUMNS)
                col = LEDCUBE_NUMBER_COLUMNS - 1;

        // force the bit low in LedCube_Planes, turning that led on.
        LedCube_Planes[plane]
                &= ~(1 << (row*LEDCUBE_NUMBER_COLUMNS + col));

} // End of LedCube_SetLed
```

```
// This function will turn off the led at r,c,p
void LedCube_ClearLed(int r, int c, int p)
{
        // Make a copy of r,c,p
        int plane = p, row = r, col = c;
        // Error check r,c,p
        if (plane < 0) plane = 0;
        if (plane >= LEDCUBE_NUMBER_PLANES)
             plane = LEDCUBE_NUMBER_PLANES - 1;
        if (row < 0) row = 0;
        if (row >= LEDCUBE_NUMBER_ROWS)
             row = LEDCUBE_NUMBER_ROWS - 1;
        if (col < 0) col = 0;
        if (col >= LEDCUBE_NUMBER_COLUMNS)
             col = LEDCUBE_NUMBER_COLUMNS - 1;
        // force bit high at r,c,p turning it off
        LedCube_Planes[plane]
               |= (1 << (row*LEDCUBE_NUMBER_COLUMNS + col));

} // End of LedCube_ClearLed

#endif
```

Appendix B: Arduino Code for Testing LED Cube.

```
#include <SPI.h>
#include <MsTimer2.h>
#include "LedCubeData.h"

void NextDisplay()
{
	PORTC &= 0xf0; // set bottom bit low, turning off display.
	// Send next pattern to Shift Registers.
	SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));
	SPI.transfer16(LedCube_NextPlane());
	SPI.endTransaction();
	// Turn on the current plane.
	PORTC |= (1 << LedCube_CurrentPlane);
} // End of NextDisplay

// As a test, each plane is given a location (row,column)
// and that plane is moved through all its locations.
int ZeroRow = 0, ZeroColumn = 0;
void MoveZero()
{
	ZeroColumn++;
	if (ZeroColumn >= 4)
	{
		ZeroColumn = 0;
		ZeroRow++;
		if (ZeroRow >= 4)
			ZeroRow = 0;
	}
} // End of MoveZero

int OneRow = 1, OneColumn = 1;
void MoveOne()
{
	OneColumn++;
	if (OneColumn >= 4)
	{
		OneColumn = 0;
		OneRow++;
		if (OneRow >= 4)
			OneRow = 0;
	}
} // End of MoveOne

int TwoRow = 2, TwoColumn = 2;
void MoveTwo()
{
	TwoColumn++;
	if (TwoColumn >= 4)
	{
		TwoColumn = 0;
		TwoRow++;
		if (TwoRow >= 4)
			TwoRow = 0;
	}
} // End of MoveTwo

int ThreeRow = 3, ThreeColumn = 3;
void MoveThree()
{
	ThreeColumn++;
	if (ThreeColumn >= 4)
	{
		ThreeColumn = 0;
		ThreeRow++;
		if (ThreeRow >= 4)
			ThreeRow = 0;
	}
}  // End of MoveThree
```

```
unsigned long Timer = 0;

// setup code, run once:
void setup()
{
        MsTimer2::set(4, NextDisplay); // 4ms period
        MsTimer2::start();

        // A3-A0 to outputs.
        DDRC |= 0x0f;
        // Set up display data.
        LedCube_ClearData();
        // Start up the SPI
        SPI.begin();

        // Timer for moving the ON led's
        Timer = millis();

} // End of setup

// main code, run repeatedly:
void loop()
{
        // 200 millisecond timer to update display
        if (millis() - Timer >= 500)
        {
                // Clear currently on LED.
                LedCube_ClearLed(ZeroRow, ZeroColumn, 0);
                // Move to next location.
                MoveZero();
                // Set next ON led.
                LedCube_SetLed(ZeroRow, ZeroColumn, 0);

                // Repeat for plane one
                LedCube_ClearLed(OneRow, OneColumn, 1);
                MoveOne();
                LedCube_SetLed(OneRow, OneColumn, 1);

                // Plane two
                LedCube_ClearLed(TwoRow, TwoColumn, 2);
                MoveTwo();
                LedCube_SetLed(TwoRow, TwoColumn, 2);

                // Plane three
                LedCube_ClearLed(ThreeRow, ThreeColumn, 3);
                MoveThree();
                LedCube_SetLed(ThreeRow, ThreeColumn, 3);

                Timer += 500; // Update timer

        } // End of timer if.

} // End of loop.
```