

Lab 8 – Serial Communication and the 7-Segement Display

# Preparation

* Read Sections 6.2, 6.4, 8.2, and 11.3
* Go through Lectures 23 and 24 again
* See the datasheets for the 7-segment display and the shift register
* Download and load into CCS “EGCP\_450\_Lab\_8\_Part\_1.zip”
* [Watch the related video(s) here](https://www.youtube.com/playlist?list=PLxYT_0WQMg35FiE5EDnCwW-1p7-E3gNtJ)

# Purpose

## Part 1: 7-Segment Device Driver

This part of the lab has these major objectives: 1) to interface a 7-segment display that can be used to display information on the embedded system, 2) using port interrupts, and 3) use SysTick to debounce a switch.

## Part 2: 4-Digit 7-Segment Interface

This part of the lab has these major objectives: 1) to interface a 4-digit 7-segment display, 2) use the SN74HC595N 8-bit shift register IC, 3) learn about the time-multiplexing technique, and 4) create an interface in CCS.

# System Requirements

## Part 1: 7-Segment Device Driver

The objective of this part of the lab is to develop a device driver for the 7-segment display. In particular, there are three components of a device driver. The first component is the description of the driver. Place the function prototypes for the public functions in the header file **SSEG.h**. It is during the design phase of a project that this information is specified. The second component of a device driver is the implementation of the functions that perform the I/O. Place the implementations in the corresponding code file (e.g., **SSEG.c**). In addition to public functions, a device driver can also have private functions. This interface might require some private functions (private functions should not include **SSEG\_** in their names). In this lab, you are required to develop and test the developed public functions (public functions must include **SSEG\_** in their names). The third component is a main program that can used to test these functions.

In the **SSEG.c** file, you will implement and test the functions to communicate directly with the 7-segment display. You will write the initialization ritual (i.e., **SSEG\_Init**) which must be a public function in the **SSEG.c** file. You will also write a function that will output to the 7-segment display. Your **SSEG\_Out** function will be used to output a number, between 0 and 15, to the 7-segment display which will be displayed on a single digit as a HEX value.

/\*

\* SSEG\_Init Function

\* Initialize 7-segment display

\* Inputs: none

\* Outputs: none

\*/

void SSEG\_Init**();**

/\*

\* SSEG\_Out Function

\* Output a number to a single digit of the 7-segment display

\* Inputs: a number between 0 and 15

\* Outputs: none

\*/

void SSEG\_Out**(**uint8\_t num**);**

You will also create an interrupt handler for Port 5 (**PORT5\_IRQHandler**). This handler will be executed when the external switches connected to the port are pressed. Use this to change the number that will be displayed on the 7-segment display digit. One button will decrease the number and one button will increase the number. The number must be between 0 and 15 (). So, when you reach the limits, wrap around. For example, if the current number is 15 and the button to increase is pressed, the number will wrap to 0. Also, use SysTick to debounce the switches. You can do this by waiting for 20 ms at the end of the handler.

## Part 2: 4-Digit 7-Segment Interface

### Task A

In this part of the lab, you will extend the driver from Part 1 to interface a 4-digit 7-segment display to the MSP432 using a TI SN74HC595N 8-bit shift register (Figure 1). To do so, you will need to change the existing public functions and add the following public functions:

/\*

\* SSEG\_Init Function

\* Initialize 7-segment display

\* Inputs: none

\* Outputs: none

\*/

void SSEG\_Init();

/\*

\* SSEG\_Out Function

\* Output a 4-digit number to the display

\* Inputs: none

\* Outputs: none

\*/

void SSEG\_Out();

/\*

\* SSEG\_Shift\_Out Function

\* Shifts data out serially

\* Inputs: 8-bit data

\* Outputs: none

\*/

void SSEG\_Shift\_Out(char);

/\*

\* SSEG\_Disp\_Num Function

\* Seperate the input number into 4 single digit

\* Inputs: num between 0 and 9999

\* Outputs: none

\*/

void SSEG\_Disp\_Num(int);

/\*

\* SSEG\_Off Function

\* Turns off all 7-seg digits

\* Inputs: none

\* Outputs: none

\*/

void SSEG\_Off();

This lab will use time-multiplexing. Multiplexed displays only show one digit at a time, but by cycling through all digits repetitively and cycling very fast, a multi-digit display is perceived. To drive a particular digit, its common cathode is connected to common ground and the LEDs are driven as needed for the desired number in that place. The cycling through the digits must be done quickly enough for flicker fusion to occur and this is accomplished through the system’s software.

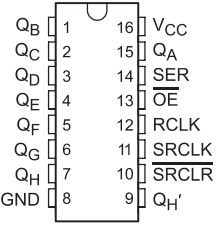


Figure 1: TI SN74HC595N 8-bit shift register.

As you can see in Figure 2, there are a total of 34 LEDs. However, there are only 14 pins. As shown in the circuit diagram, there are 5 common cathodes. Also, anodes of the same LEDs are connected together. So, if we connect all the LED pins to VDD and common pins to ground, you can see all the LEDs of all four digits are turned on. By switching between the four common cathodes we can choose which display to turn on. However, we can’t display four different digits at the same time in this display as anodes are the same for all four digits.

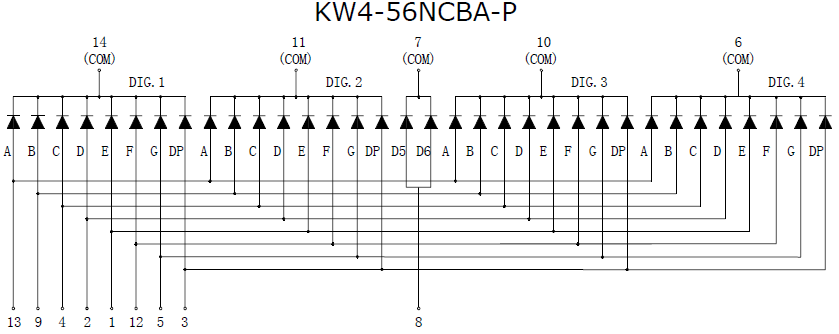


Figure 2: Diagram of common cathode 7-segment display (Lucky Light KW4-56NCBA-P).

A time-multiplexing method will be used to display a 4-digit number on this display. To do this, you will show one digit at a time on a display unit and switch between units fast enough such that it is perceived that all the digits are on simultaneously. For example, if we want to display the number “1234”, we need to do the following steps:

1. Turn the LEDs needed to display a “1”, and enable the 1st (leftmost) digit.
2. Delay ~1 second.
3. Turn off the 1st digit, turn the LEDs needed to display a “2”, and enable the 2nd digit (the digit to the right of the 1st digit).
4. Delay ~1 second.
5. Turn off the 2nd digit, turn the LEDs needed to display a “3”, and enable the 3rd digit (the digit to the right of the 2nd digit).
6. Delay ~1 second.
7. Turn off the 3rd digit, turn the LEDs needed to display a “4”, and enable the 4th (rightmost) digit.
8. Delay ~1 second.
9. Go back to step 1.

As you can see in Figure 3, we will use the 74HC595N IC. This IC is a shift register and it converts serial data to parallel data which can be used to reduce the number of digital I/O pins required to control the 7-segment display. As can be seen, we need only 3 I/O pins (connected to the shift register IC) to control 8 LED lines of the 4-digit 7-segment display. Here are the names of the pins and their functions:

* QA to QH – Parallel data output.
* SRCLK – Shift Register Clock. A clock pulse should be sent to this pin from the MSP432 MCU. If not, the shift register can’t identify the starting and end points of the data.
* RCLK – Storage Clock (a.k.a., Latch Pin). This pin is used to set the mode of the shift register. When latch pin is set to LOW, the shift register reads input values.
* – Output Enable. This pin should be connected to ground for the shift register to output values.
* SRCLR – Shift Register Clear. When this pin is connected to Vcc, all the data stored in the shift register will be cleared.
* SER – Serial Data Input.

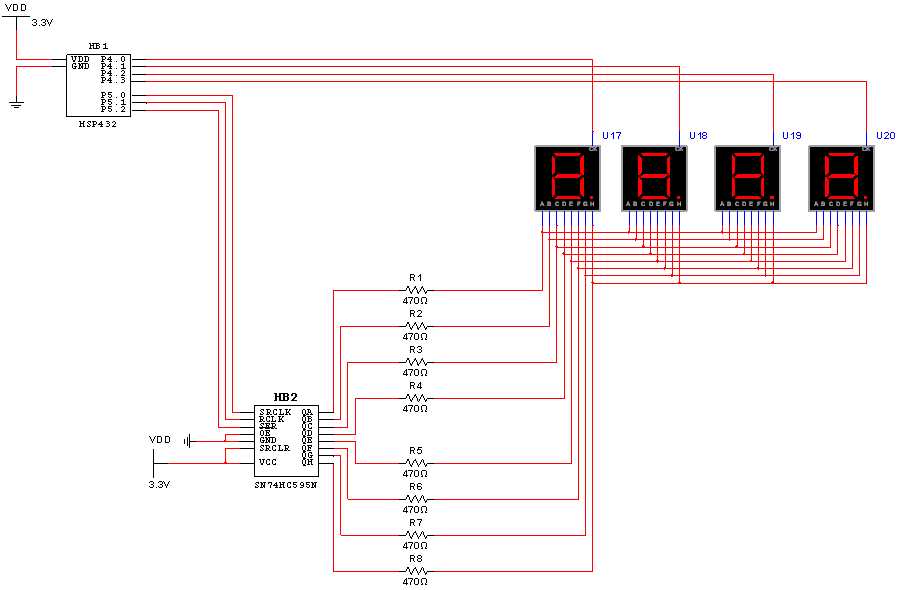


Figure : 7-segment display interface for Part 2.

All the LEDs of the 7-segment display must be connected to the parallel data output pins of the shift register through 470Ω resistors. The clock, latch, and serial data pins of the shift register must be connected to the MSP432 through port 5. Finally, each of the 4 common cathode pins must be connected the MSP432 through port 4.

### Task B

Now, in the main function, print to the console “Please Enter Number Between 0 - 9999” using printf and read in the input from the user (see functions “scanf” and “atol” in “stdlib.h”). If the number is not in this range, request a number again. Otherwise display the number. See the [video on the playlist](https://www.youtube.com/playlist?list=PLxYT_0WQMg35FiE5EDnCwW-1p7-E3gNtJ) to see how it should function.

# Demonstration

You will also be required to demonstrate the proper operation on the actual microcontroller. The instructor may ask you questions on 7-segment display interfacing and/or programming. The instructor will ask to see your implementation of the 7-segment display driver.

# Lab Report

The lab report is how I will grade you on your labs. Usually, the report is due 1 week following the completion of the lab (see TITANium for due dates). However, do to unforeseen circumstances, due dates may change. I will try my best to keep everyone informed of any changes. With this said, it is your responsibility to turn the report in during the scheduled due date. **There is a 10% penalty for late reports**. **Your report must be in MS Word Doc format. Submitting the report in another format will result in a 10% penalty.** Your lab report must include the cover sheet from the lab report template available on TITANium. **Not including the cover sheet will result in a 10% penalty.** The template contains instructions for the report and the rubric used for grading the labs. Please, read it thoroughly. **Don’t forget to include pertinent information such as code, flowcharts, waveform output, etc. in your report**. Please, no “spaghetti” code, keep your code clean and use comments. Remember, your code will affect your lab grade. If I can’t understand it, then I will assume it’s incorrect.