**EGRE 364**

**Lab #8 Report**

**Date:**

12/5/2019

**Team Members:**

Dakota Bernacki

Jonathan O’Dell

## Introduction

Lab 8 is focused on creating a device that uses four digital IR reflectance sensors and an analog distance sensor to generate accurate readings and display their results on the STM32L4 Discovery board’s LCD. These sensors need to be calibrated to produce accurate readings and the LCD and ADC need to be properly setup to respectively receive and display the data.

## Design

The code associated with this project has been zipped and uploaded to Blackboard.

The file name is: Lab8\_DakotaBernacki\_JonathanO\_Dell.zip

Primary files:

* main.c
* setupHeaders.h
* LCD\_structures.h
* LCD\_structures.c
* LCDinit.c
* ADCinit.c
* IRinit.c
* Sys\_clk\_init.c
* startup\_stm32l476xx.s
* Pin\_Connection.txt

See **Appendix A** for code snippets. [startup\_stm32l476xx.s and Pin\_Connection.txt excluded as they were provided as part of the project]

## Functionality and Correctness

**Correctness**

The STM32L4 Discovery board was correctly configured to receive data from four IR reflectance sensors and an analog distance sensor. These inputs were then displayed on the board’s LCD display for the user to read.

**Functionality**

**LCD:**

In order to display information to the user, we used the default LCD panel that comes with the STM32L4 Discovery demonstration board. This panel has six 14-segment digits — the first four of these positions additionally containing a colon and decimal point — and four *bar* positions stacked vertically in the rightmost position of the panel. Upon configuration of the LCD driver for a 1/4 Duty, 1/3 Bias display and of the GPIO pins to their proper alternate functions, all we have to do is modify the LCD driver’s display memory according to our particular configuration (*see below*). One drawback to using the LCD for sensor readout is due to its image persistence. If the display is changed too quickly, two characters in the same position may merge together, making it difficult to read.

In our code, we’ve produced two files just for this purpose: *LCD\_structures.h* and *LCD\_structures.c*. These introduce a new function, *char2mem*, a character structure, *encoding*, and three character-sets. The function *char2mem* accepts a position parameter, which can be either one of six digit-positions or the bar position, and an *encoding* character structure. Its purpose is to map the character-to-display into LCD RAM depending on the desired position. In *main.c*, we define a display buffer of seven character-structures, each corresponding with a particular LCD position. The *char2mem* function is then called once for each available position in an *LCD\_Update* function, which is useful for limiting the screen refresh rate.

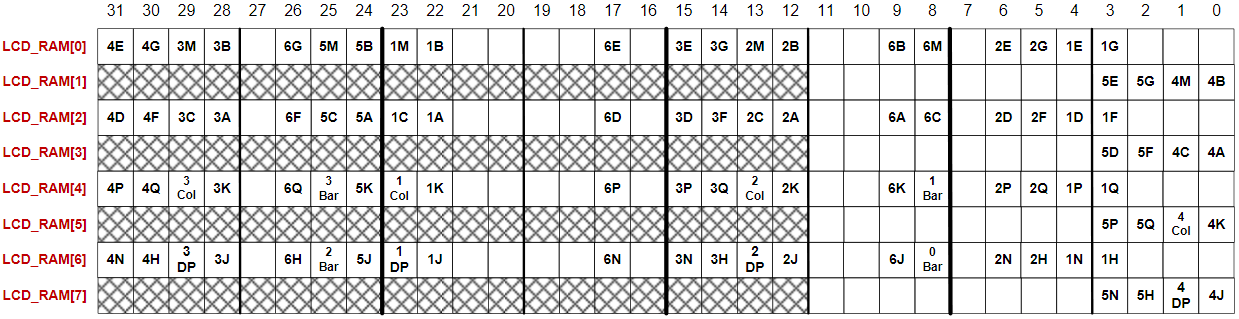


Figure : LCD Memory Bitmap

**IR Sensor:**

The QTR-1RC IR reflectance sensors produce a frequency-modulated digital output which characterizes the IR-reflectance of an object placed in front of it. Increasing the IR reflectivity toward the sensor will decrease the period of the sensor’s output. In order to initiate a reflectance reading, we first had to drive the output of the sensor high. Doing so charges a capacitor in the sensor whose purpose is to maintain a logic high. The capacitor is then discharged through an infrared-sensitive phototransistor that is proportional to the amount of infrared being reflected into it. In order to read the output, we stop driving the line and set it to a high-impedance input in order to mitigate current leakage and ensure a reliable result. The time it takes for the output to become logic low after the line is no longer externally driven is the *discharge period*.

In this project, we use four sensors, the outputs of which are attached to GPIO pins E12-15, one pin for each sensor. In order to distinguish between high and low reflectance (“white” and “black” materials) we determined a threshold for the discharge period experimentally. This threshold is defined as the amount of time to wait after de-asserting the line. After this time, if the sensor’s output is a logic low, then we interpret it as a high reflectance material. Otherwise, it’s interpreted as a low reflectance material. The threshold value which worked best for us is ~10 milliseconds. Using the LCD, we display the readings of these sensors in the four leftmost digit positions, one for each of the four sensors. A high-reflectance reading is displayed as “W” and a low-reflectance reading is displayed as “B”.

**Distance Sensor:**

The Sharp GP2Y0A21YK0F distance sensor is analog, consequently the onboard ADC had to be utilized to read its output. The onboard ADC was setup on PA1 with a 10bit resolution, single conversion, using the 80MHz clock, with the conversion initiated via software.

Once the onboard ADC was properly initialized, several additional steps needed to be taken. First the value generated by the ADC needed to be converted from binary into a voltage. Then this value needed to be converted to a distance to be sent to the LCD. To accomplish the 1st step, the ADC output conversion formula from the textbook [pg496] was used:

Where is the number of result bits and is 3.0V when using the internal voltage reference.

This was then combined with the linear formula [see Plot #1] we generated based off the sensor’s datasheet. This resulted in the following equation:

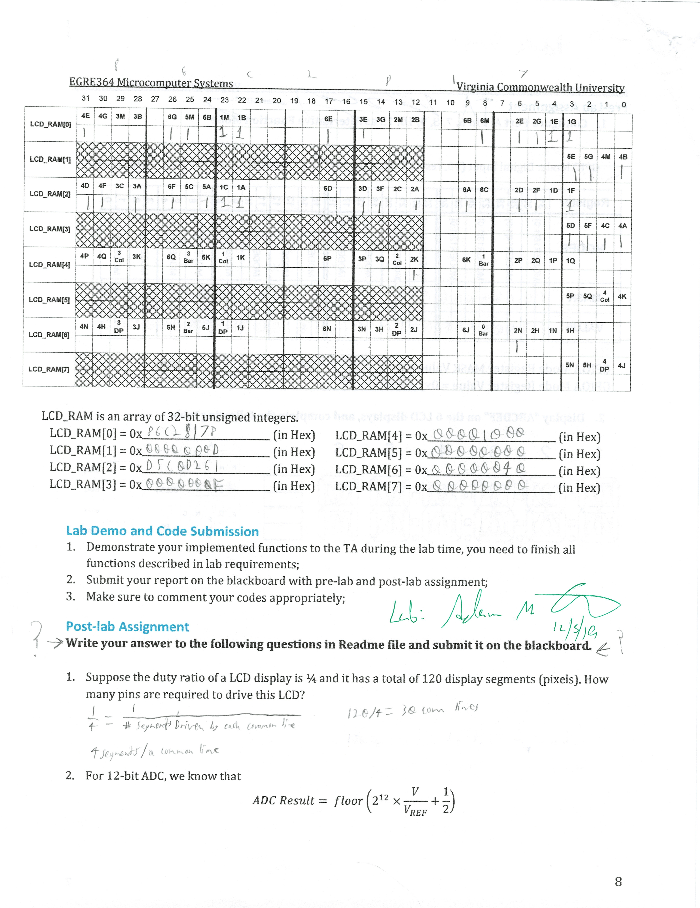
Several linearizations of the sensor’s curve were experimented with but using the 30-80cm range to generate the equation got us the most accurate readings; once we tweaked it a little via experimentation.

Once the correct distance was calculated, it was then divided and modulus by 10 to separate it into its 10’s and 1’s place. Additionally, the distance was put into an if/else to determine what percentage of the sensor’s effective range was utilized and display it on the LCD’s bars. [100% = 0cm, 4bars lit: 0% = 80cm, 0 bars lit]

**Plot #1**

## Lab Demonstration

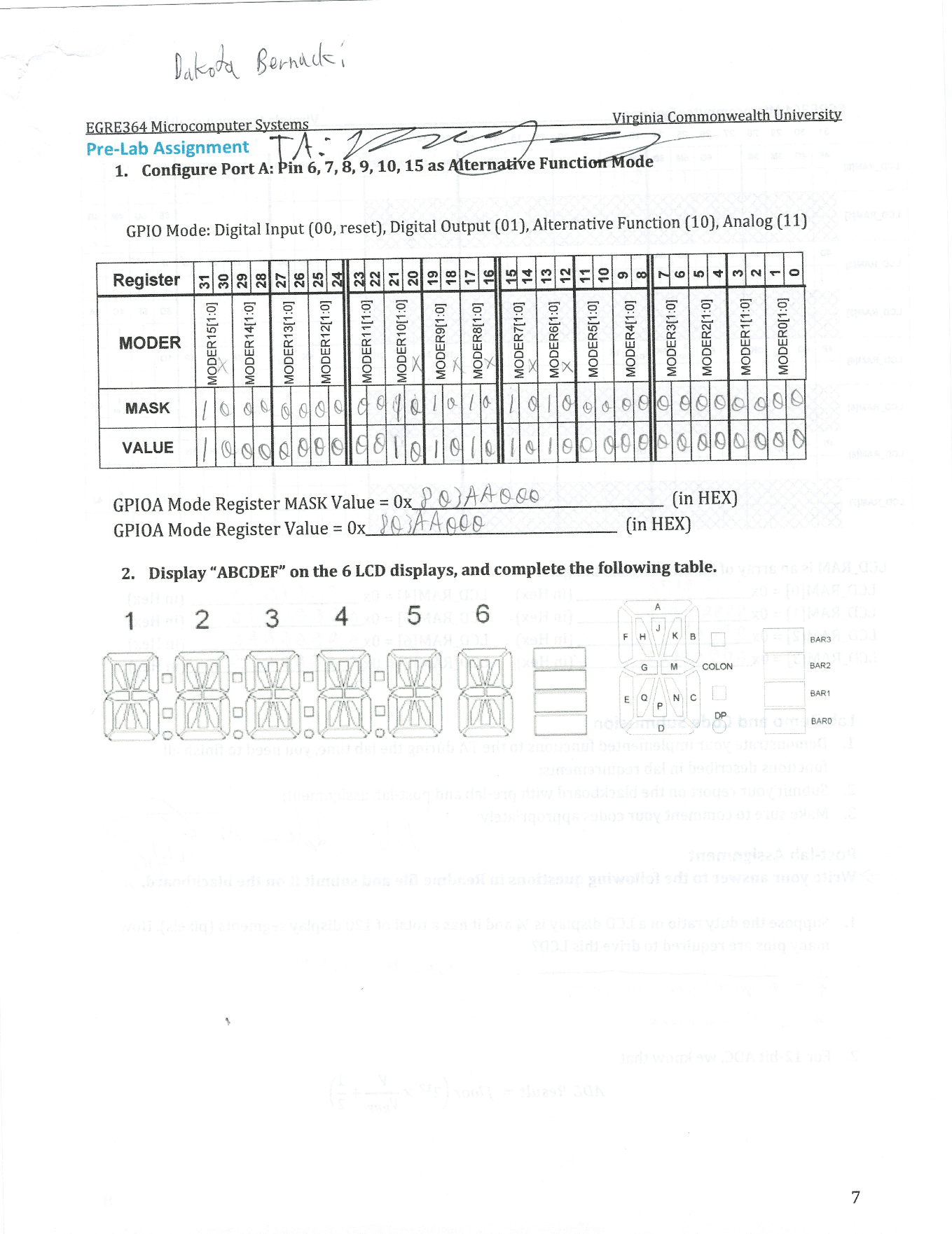
The aforementioned correct behavior was demonstrated to a TA, and he signed off on it at approximately 1320hrs 12/5/2019:

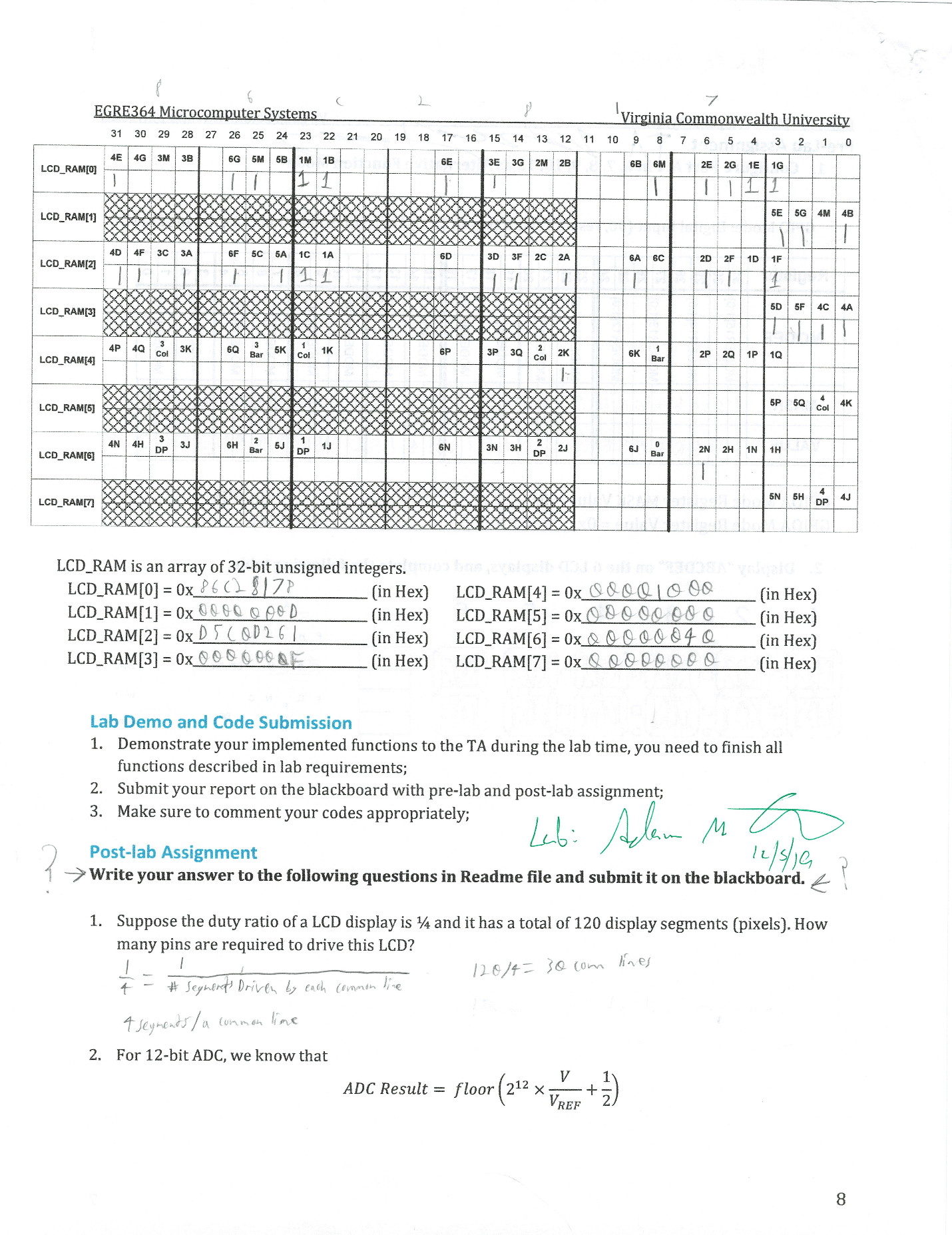


## Pre-lab assignment

Scanned pre-lab assignments:

Bernacki:





O’Dell:

## Post-lab assignment

1. Suppose the duty ratio of an LCD display is ¼ and has a total of 120 display segments. Howa many pins are required to drive this LCD:

**34pins are needed to drive the LCD**

1. For a 12bit ADC, the margin of error is found via:

Design and experiment to find an unknown Vref:

With this derived equation, a series of known voltages can be applied to the ADC input. For each of these voltages the returned ADC result can be plugged into the equation returning an approximation of the reference voltage.

At least five different voltages should be supplied the ADC and the resulting should be averaged to determine the actual .

1. What is the maximum distance your Distance Sensor can reliably detect an object:

This is HEAVLY dependent upon the size of the object and, to a lesser extent, said object’s reflectivity.

The sensor can reliably detect large objects up to distances around 90-100cm and, if the object is very reflective [bright white paper] sometimes beyond. However, a smaller object, like a finger, becomes hard for the sensor to pickup one it gets 60-70cm out. This is likely due, at least in part, to the difficulty of keeping said finger in front of the sensor’s aperture at such a range.

## Conclusions

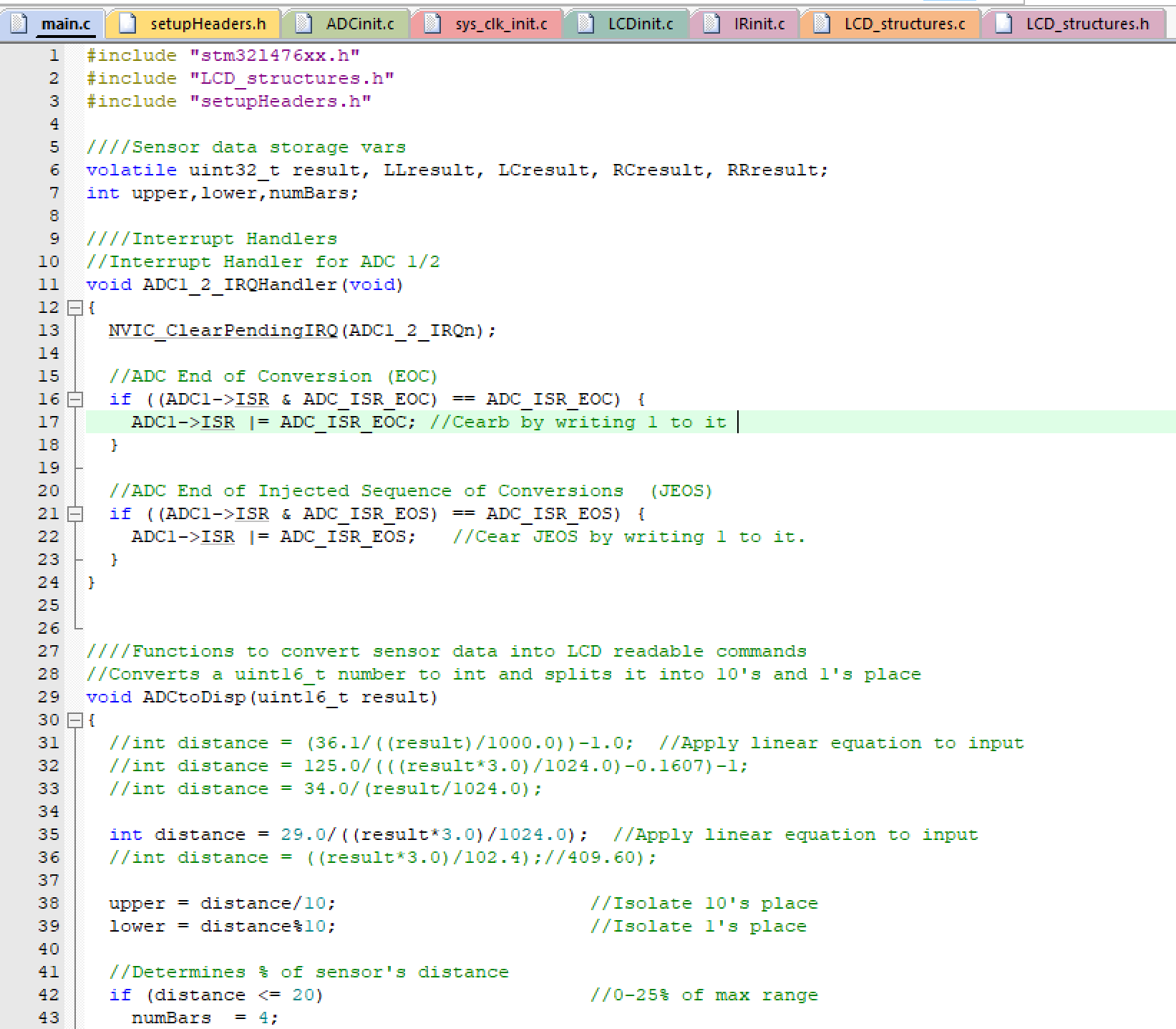
All lab objectives were successfully achieved. The data from the IR reflectance sensors and the distance sensor was correctly calibrated and displayed on the STM32L4’s LCD.

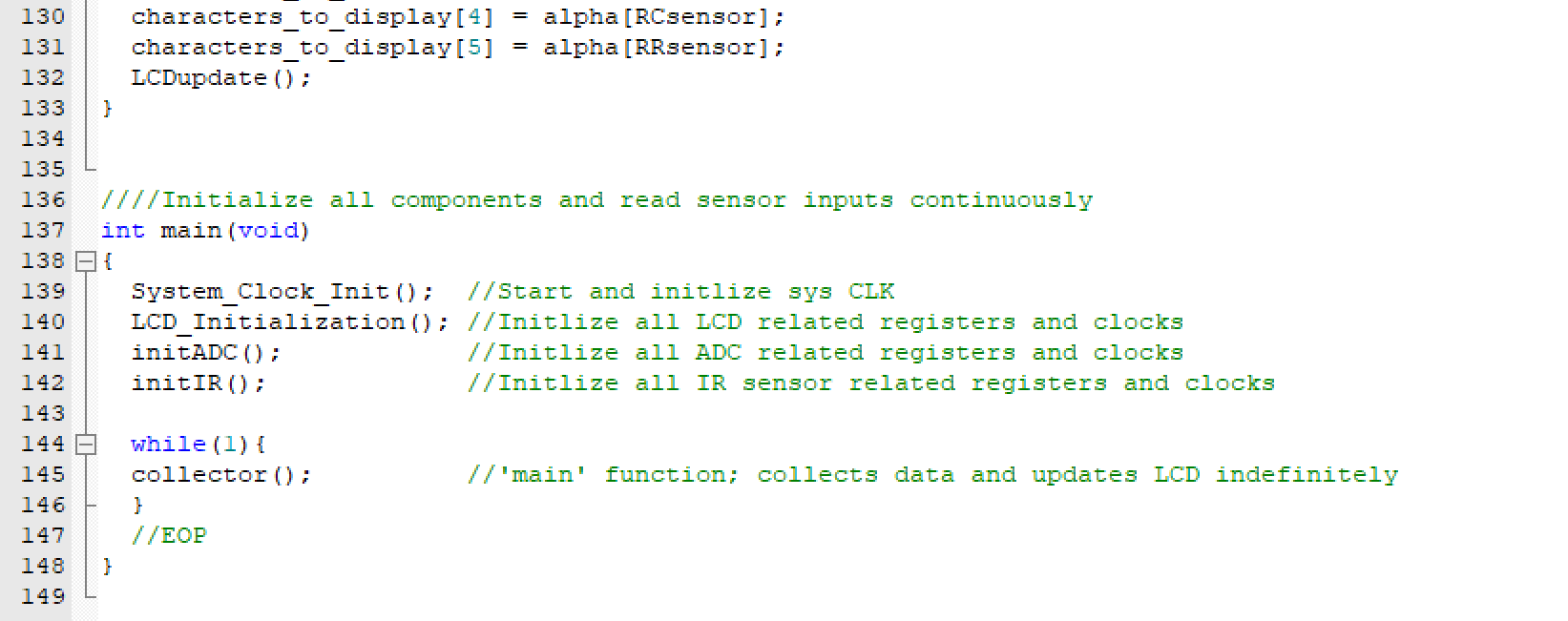
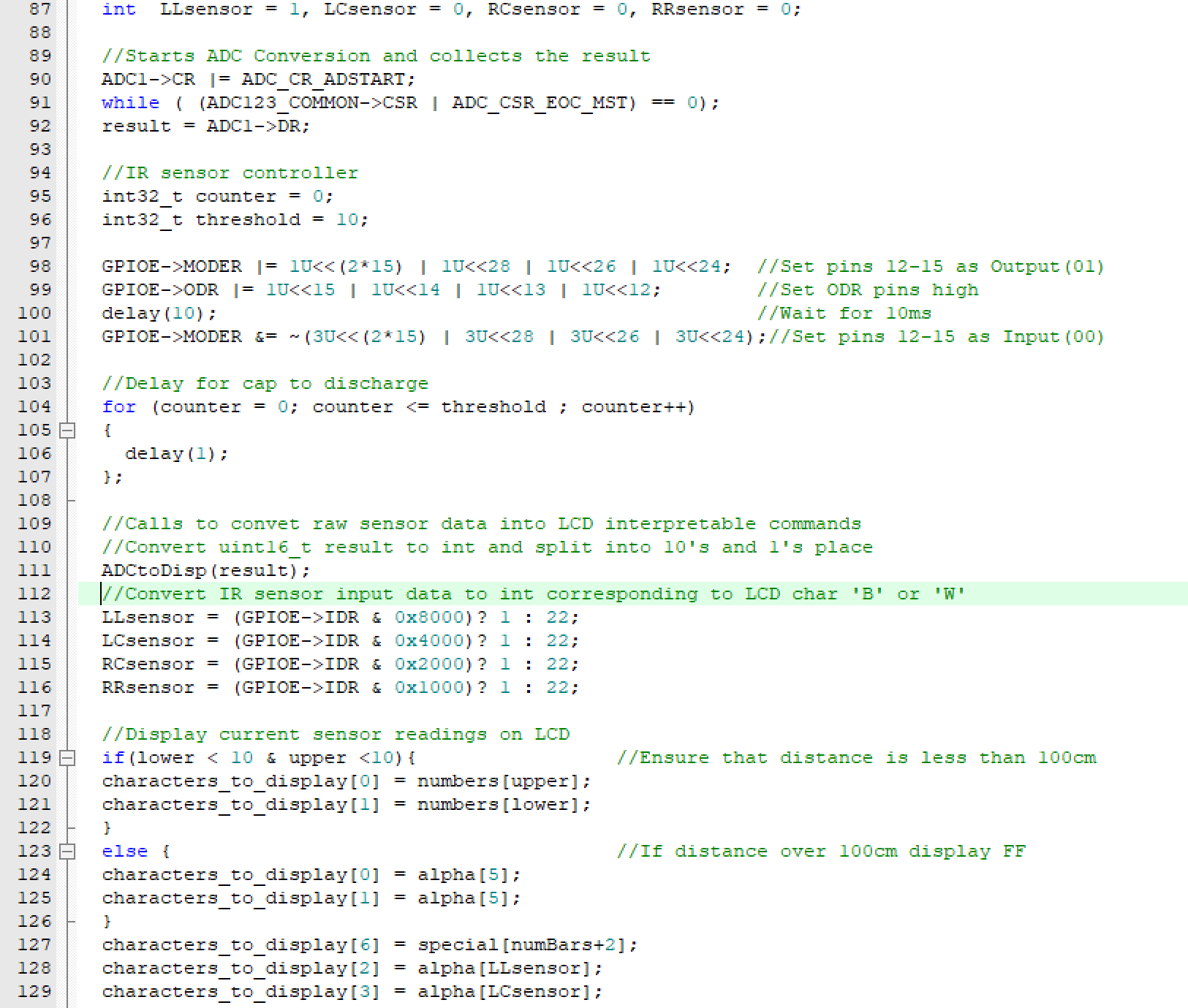
Initially there was significant difficulty in getting the custom LCD bitmask to display the correct characters. After troubleshooting, it was discovered that the bitmask was written in reverse order and that the LCD RAM requires that all 32 bits, of a RRAM[x] section, to be written each time; we were only writing 4bit areas at a time.

There was also an error with the ADC initialization. Its resolution was mistakenly set to 12bits. This resulted in some very convoluted attempts to get the sensor reading accurately. Once the mistake was discovered, and the ADC set to use 10bits, our calculation to calibrate the sensor became much simpler and more accurate.

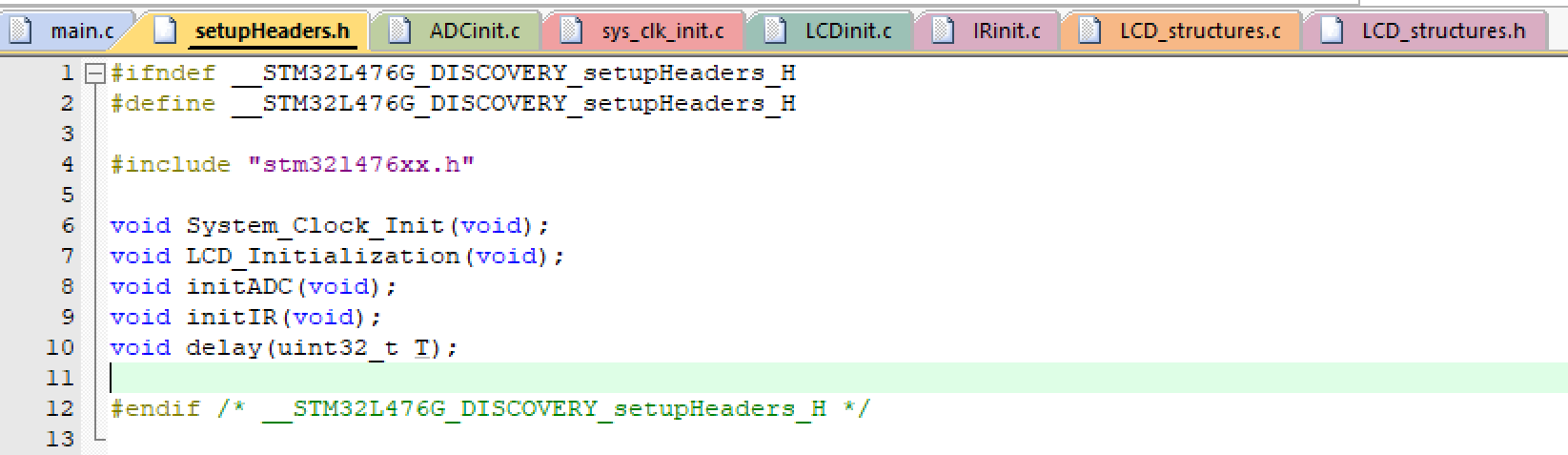
## Appendix A

**main.c:**

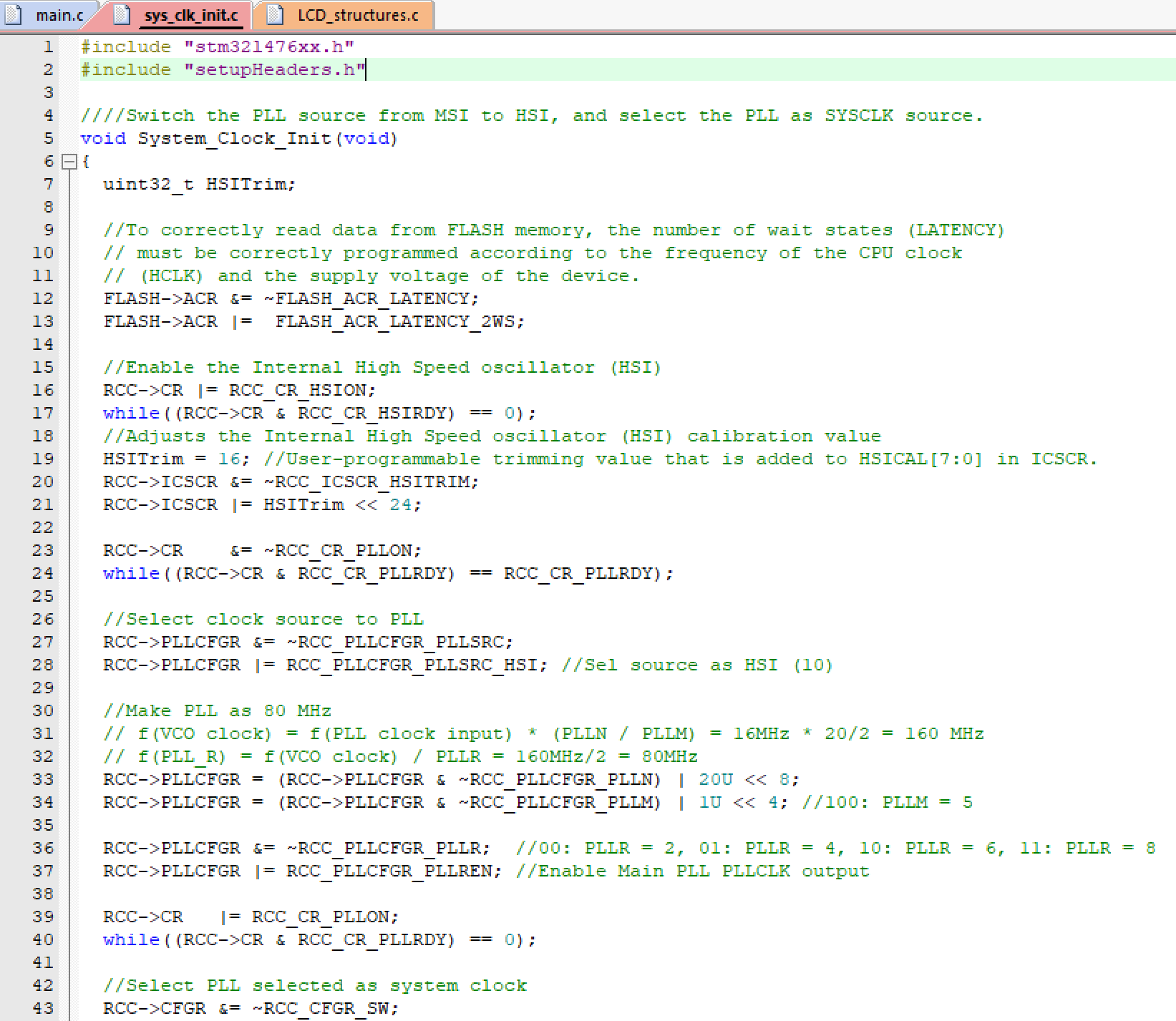
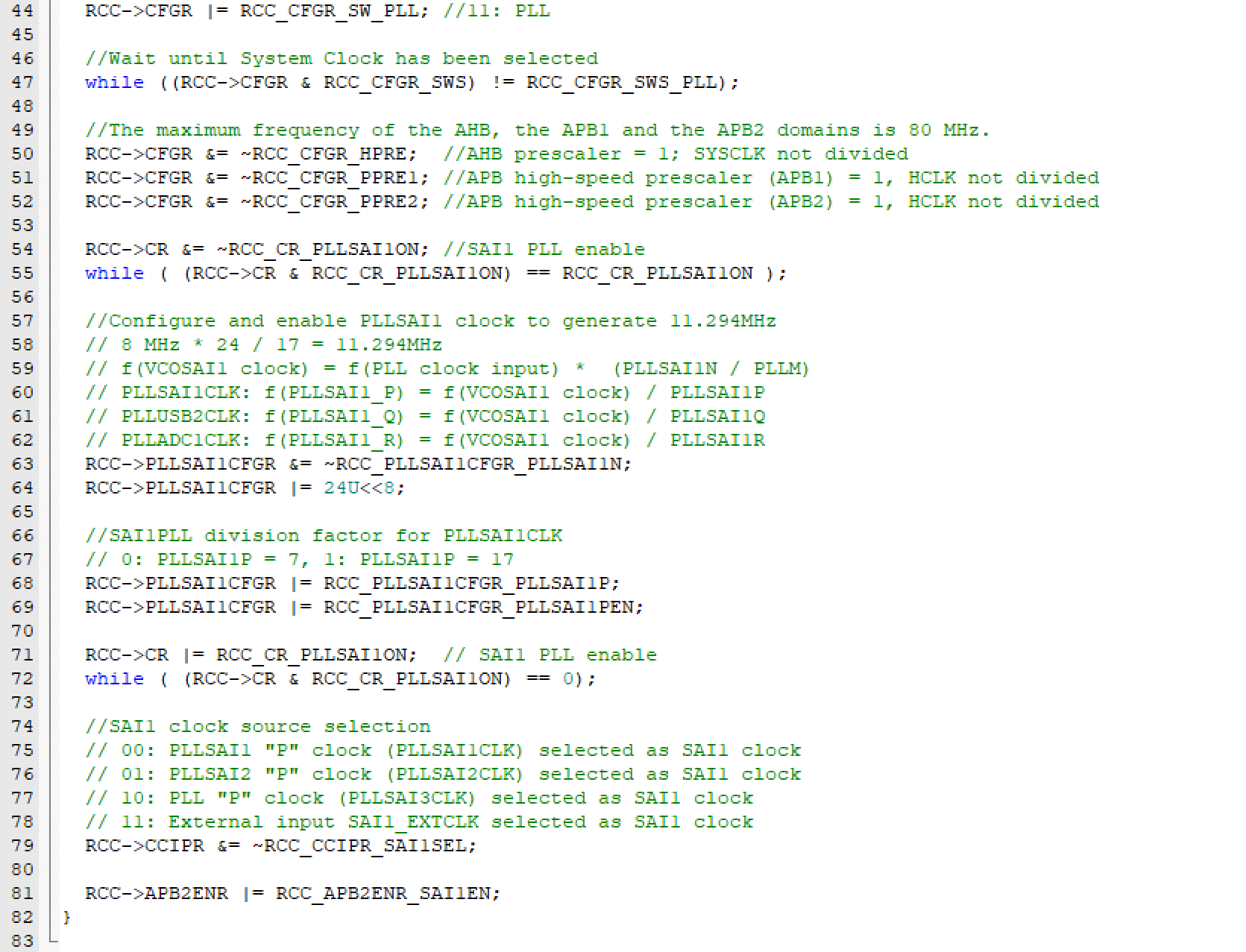




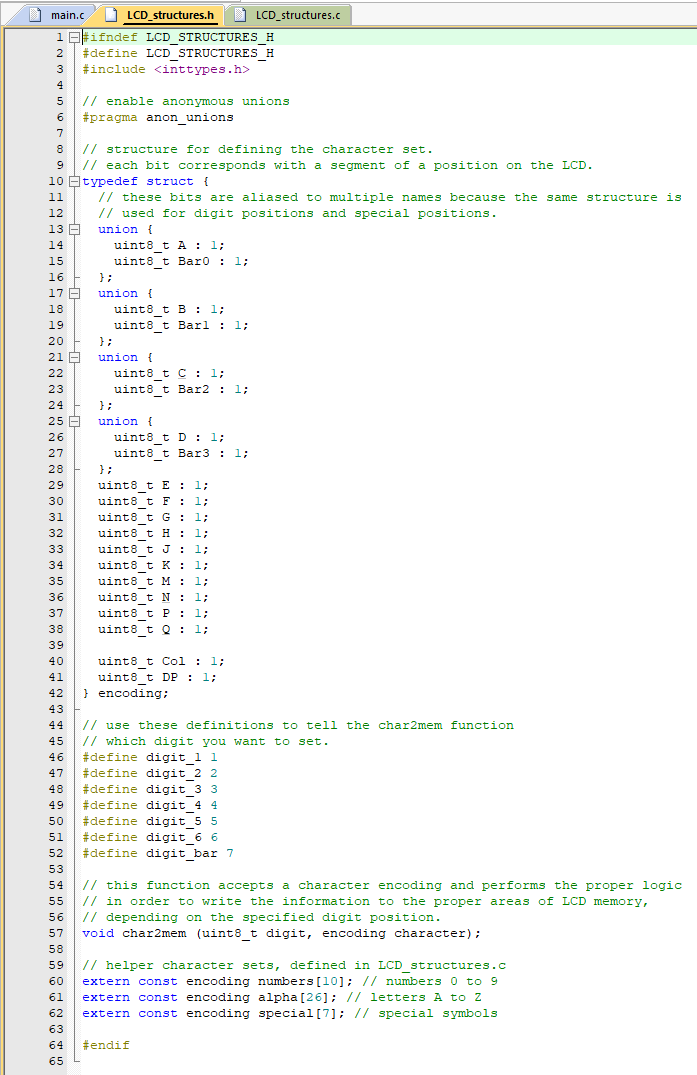
**setupHeaders.h:**



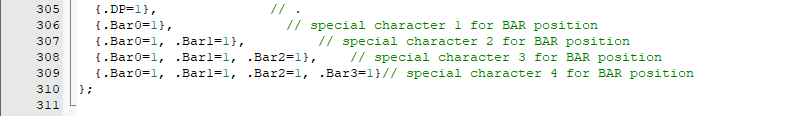
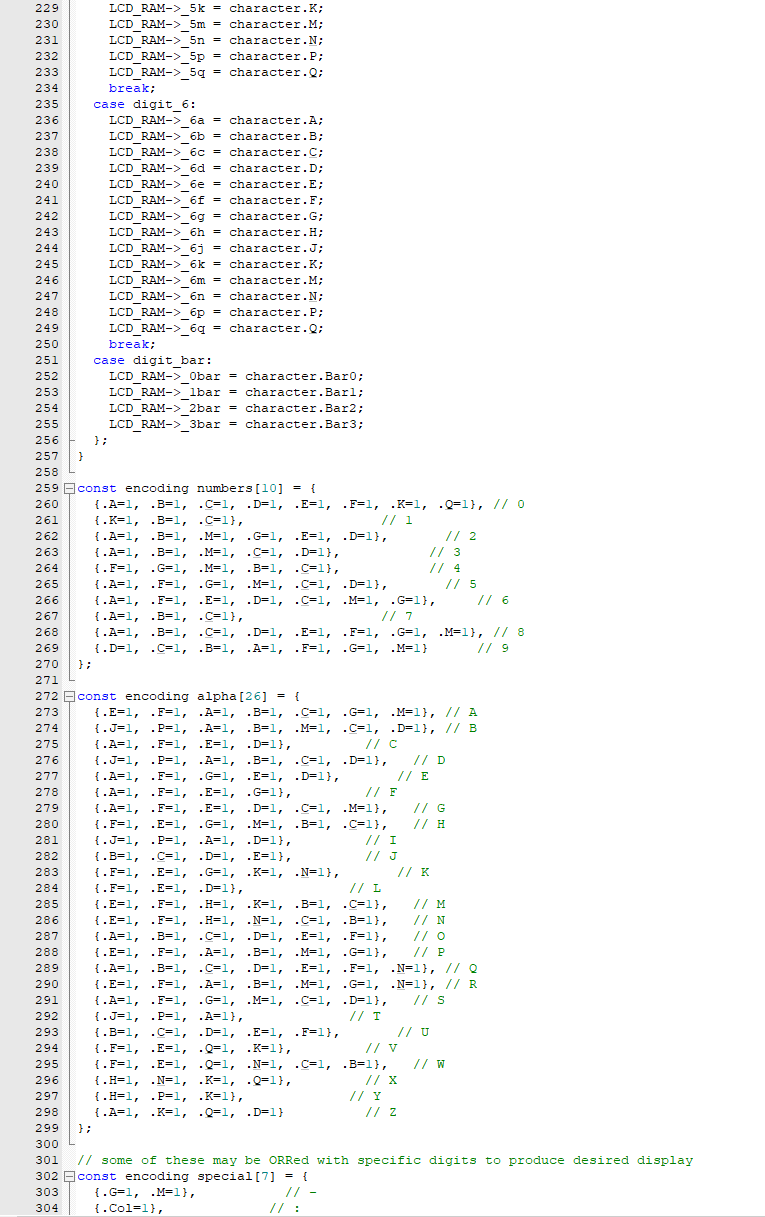
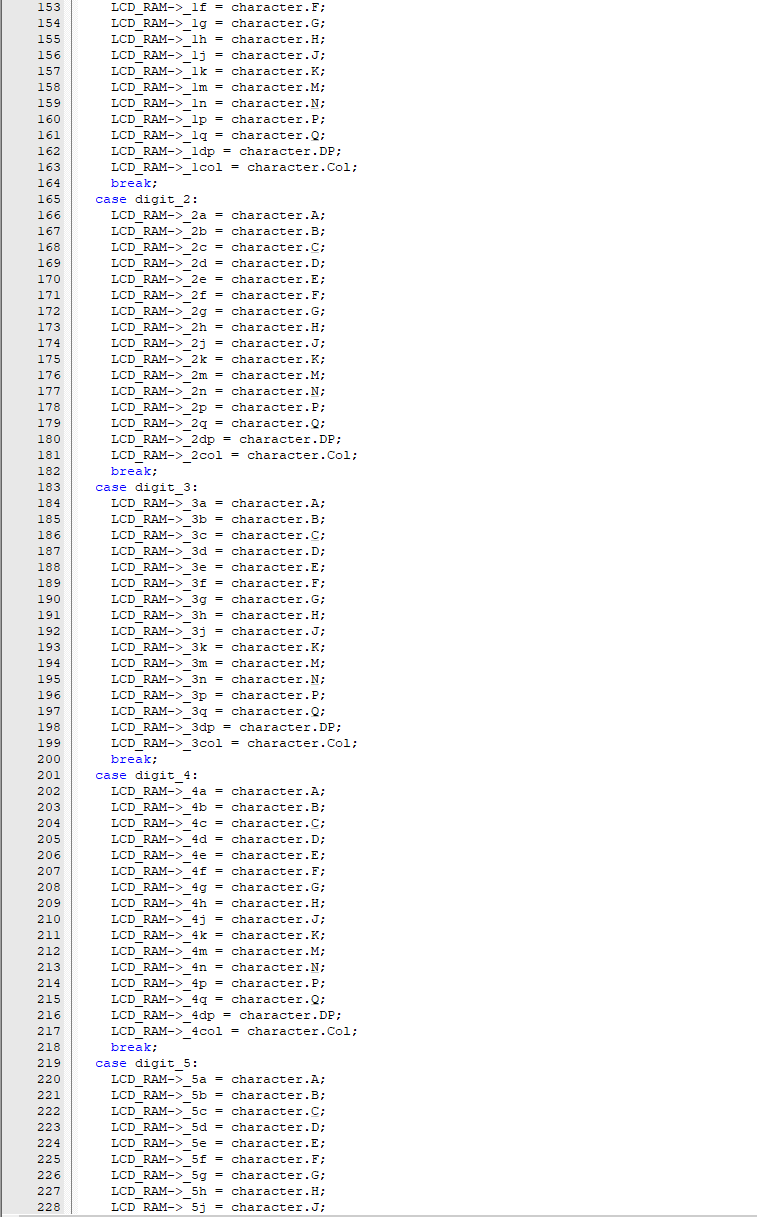
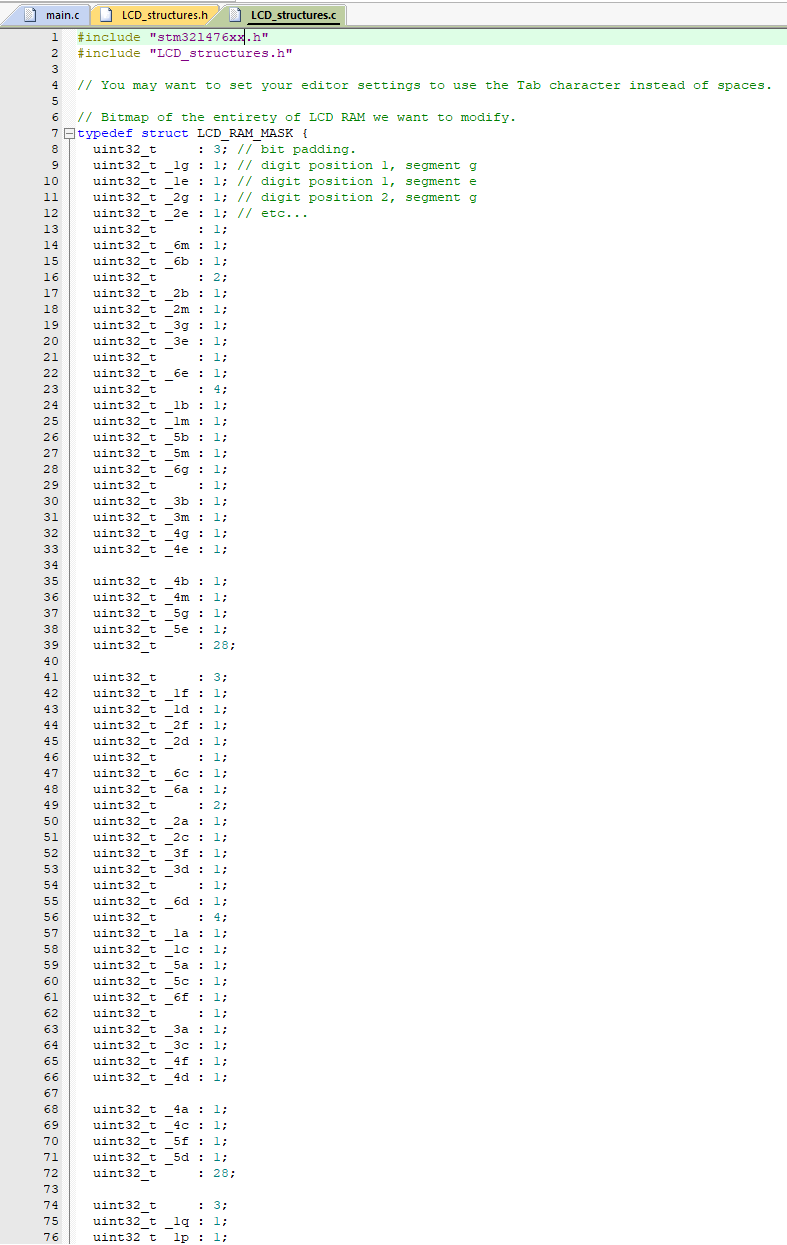
**Sys\_clk\_init.c:**

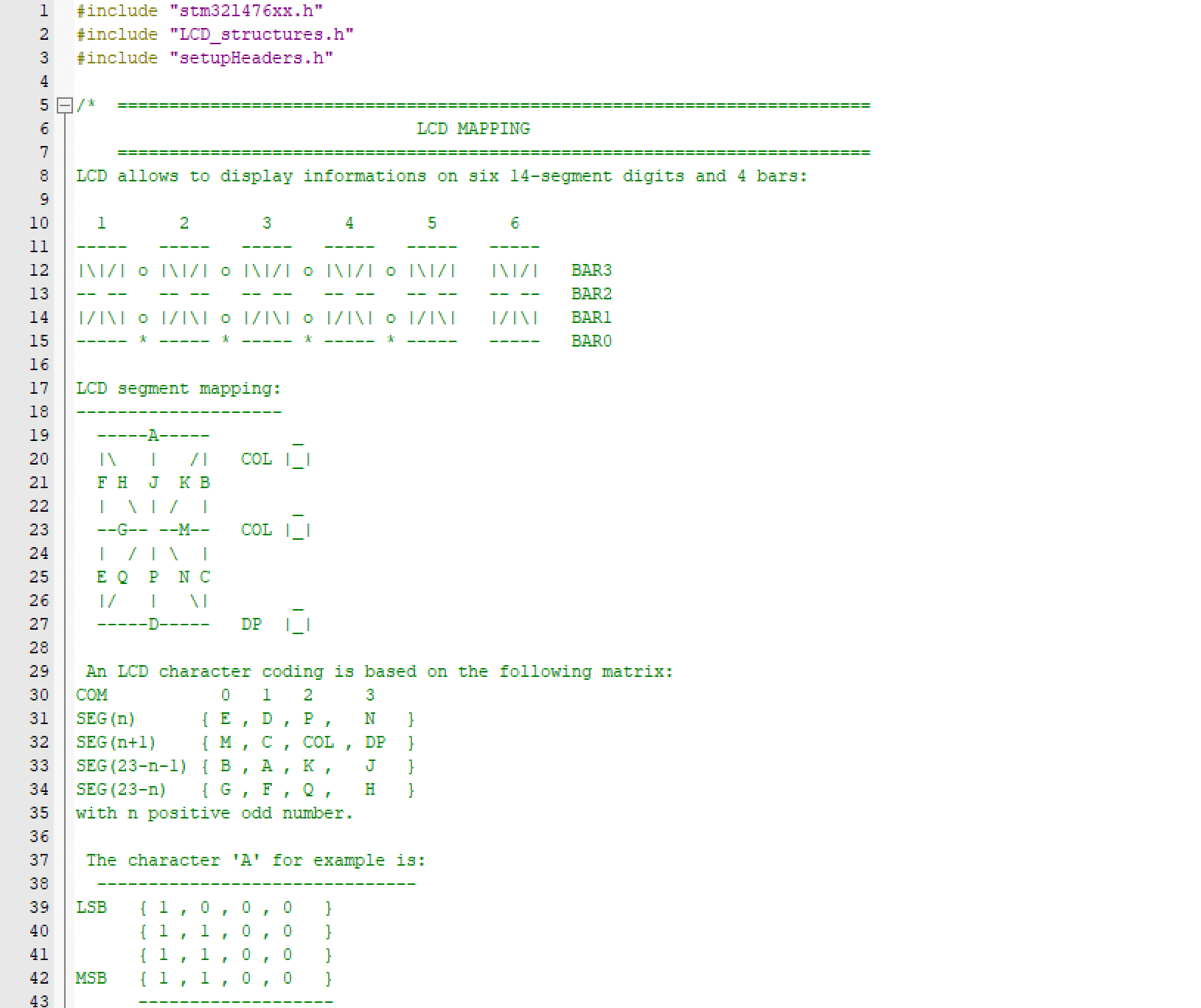
**LCD\_structures.h:**

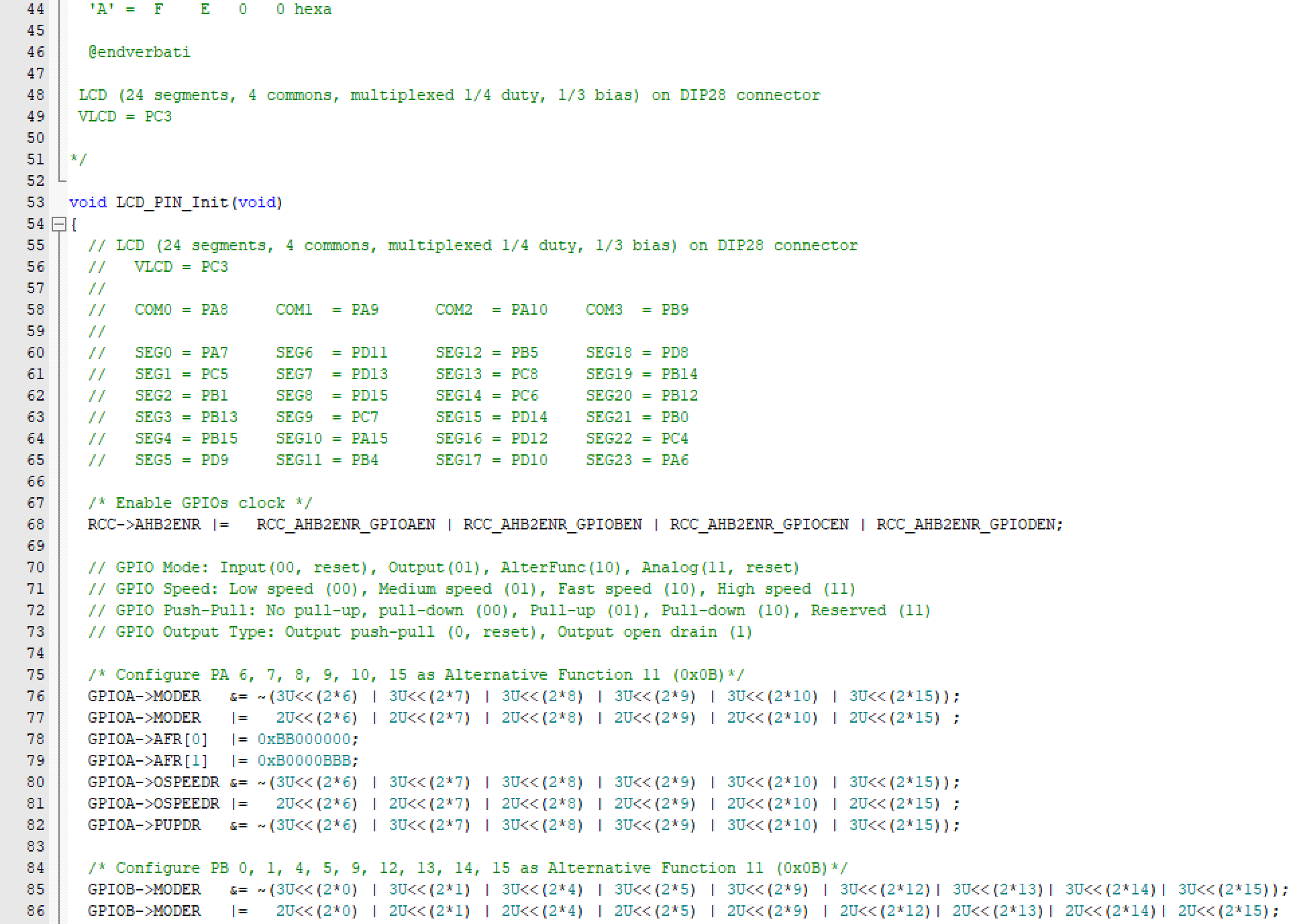
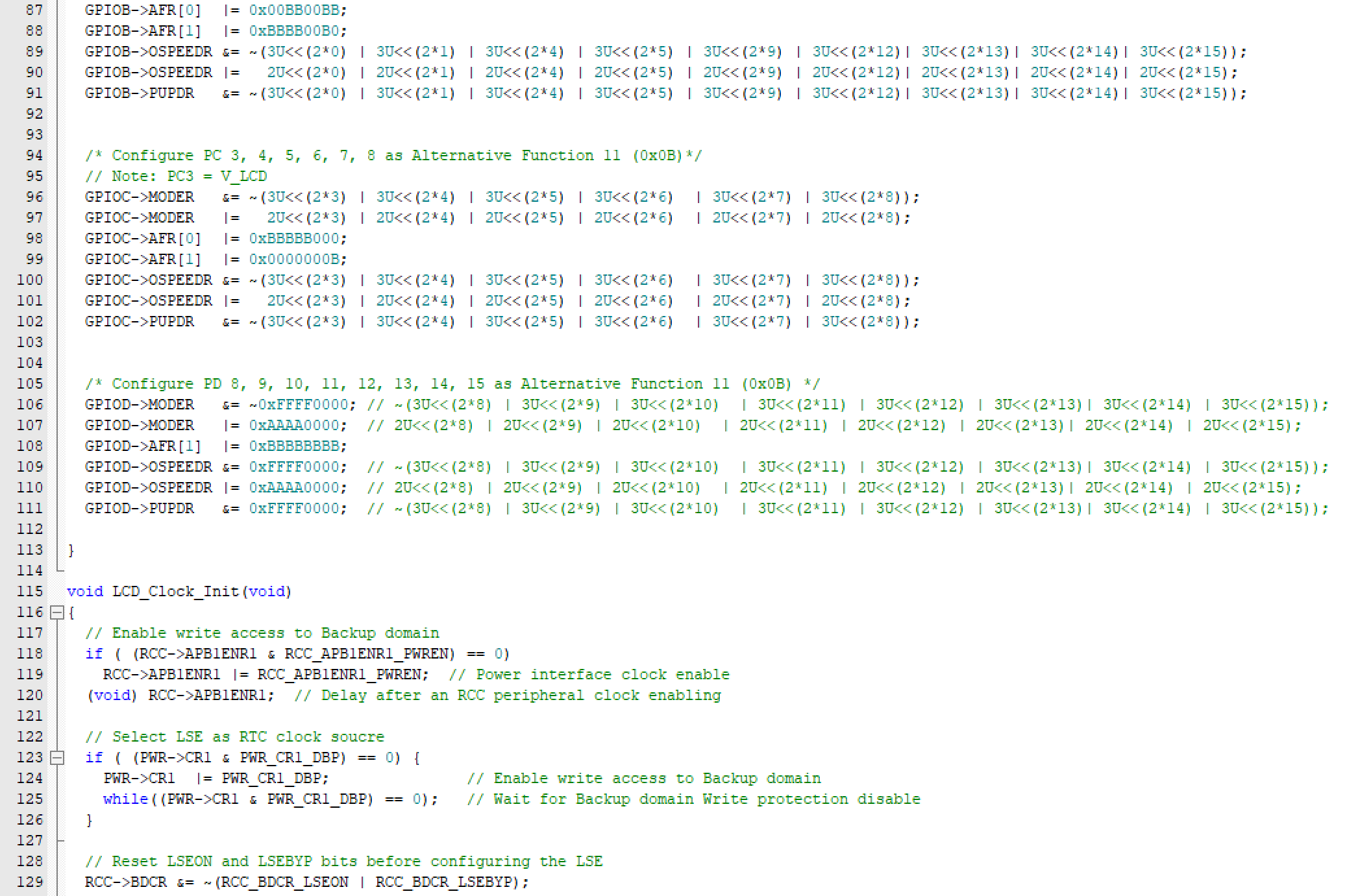
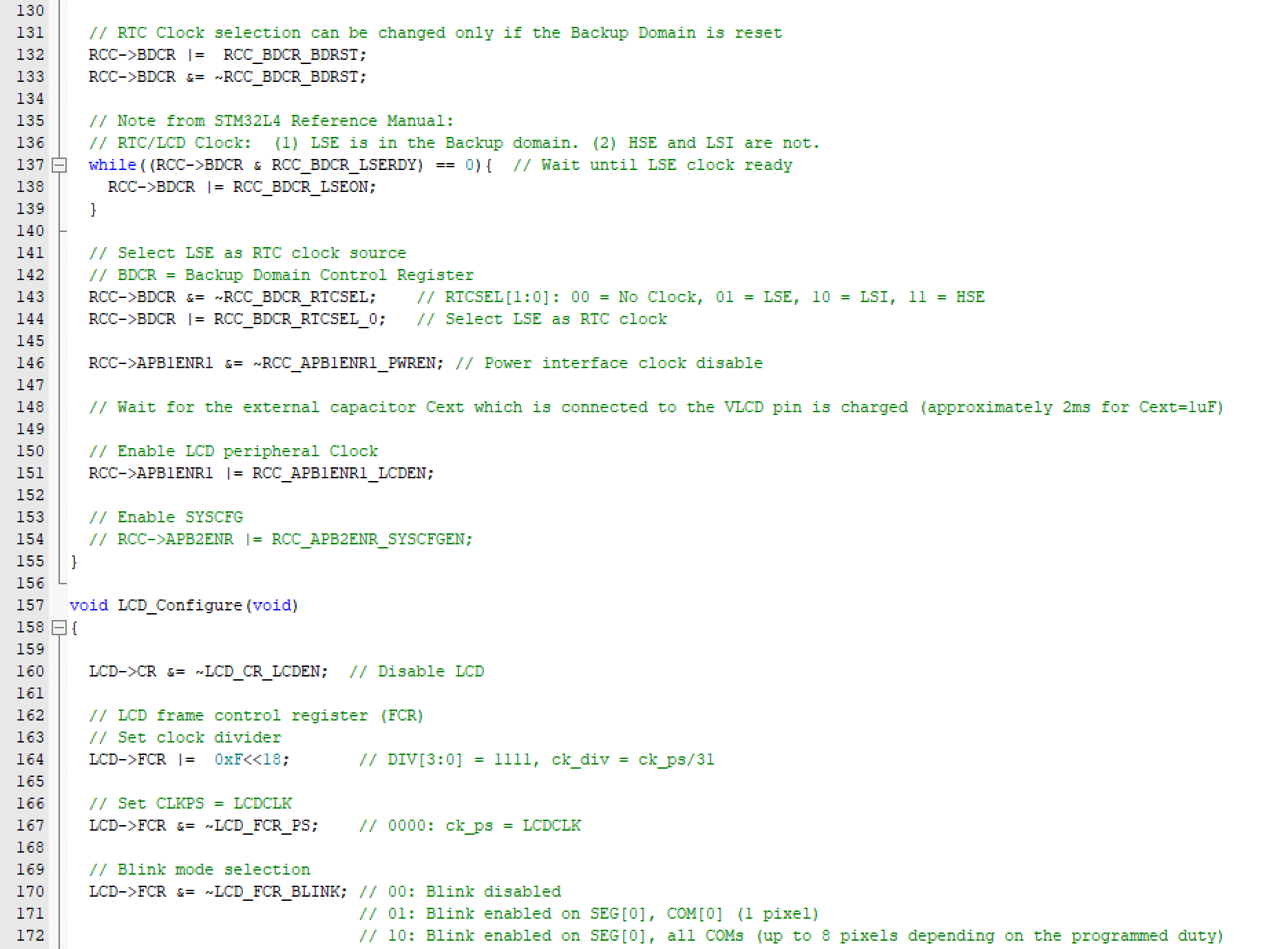
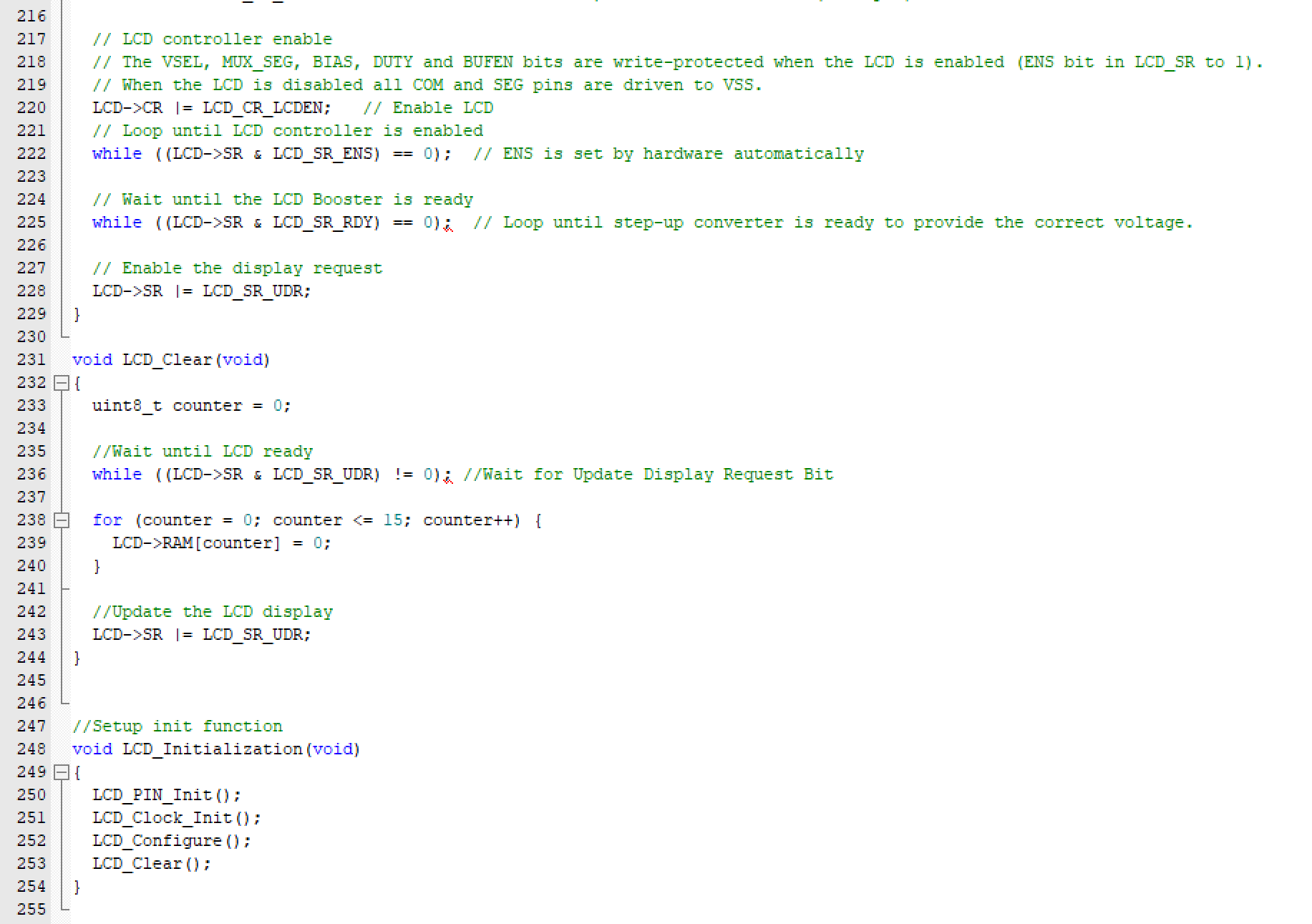


**LCD\_structures.c:**

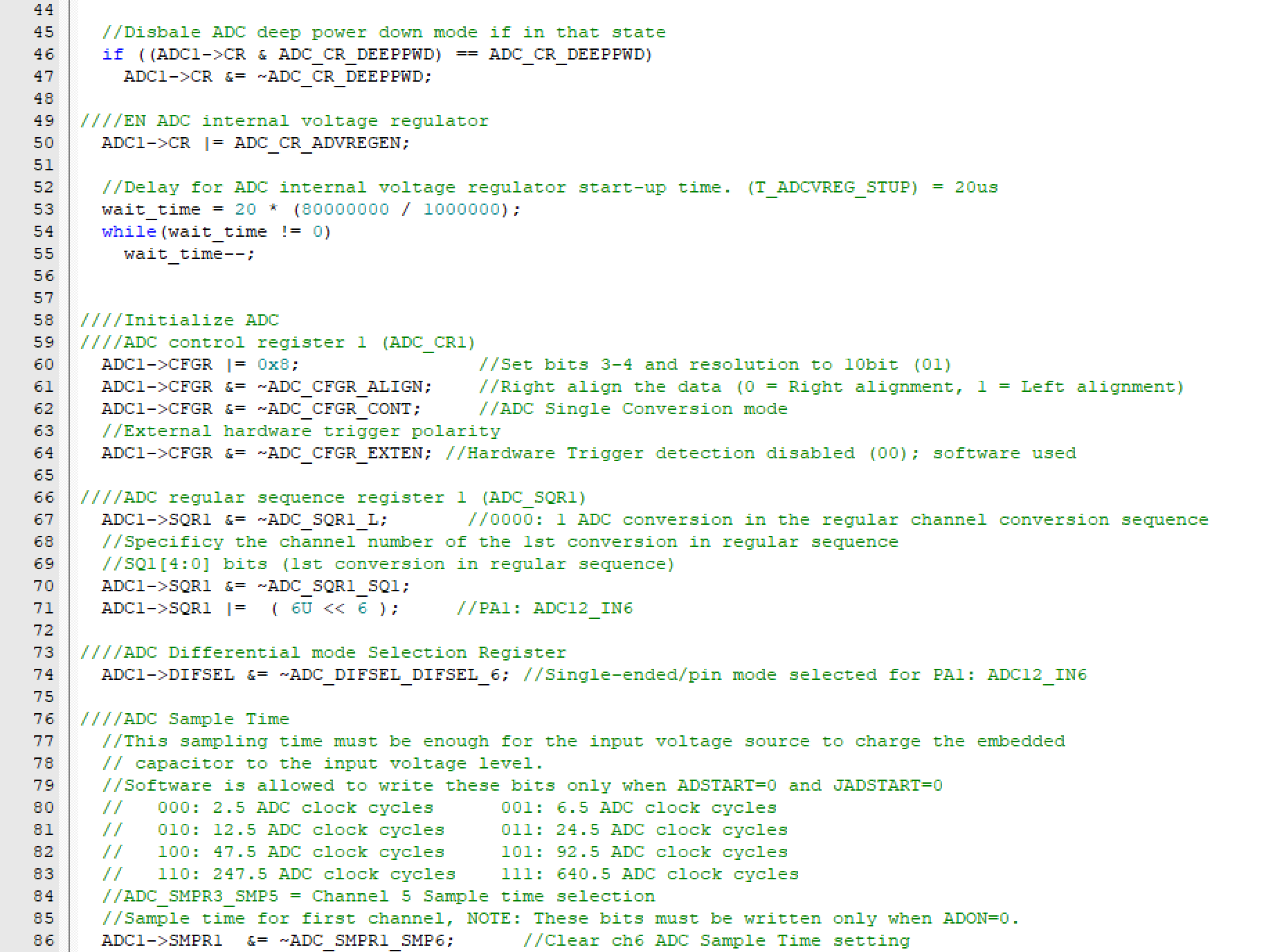
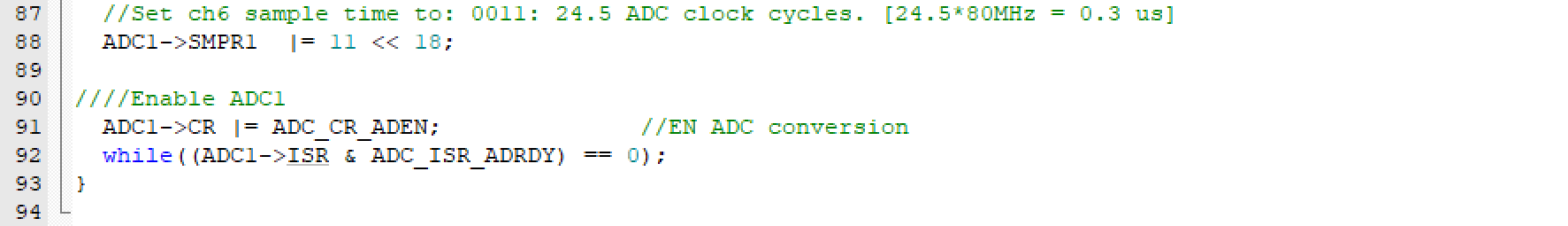


**LCDinit.c:**



**ADCinit.c:**

**IRinit.c:**

