UCD School of Electrical and Electronic Engineering

EEEN40280

Digital and Embedded Systems

Microcontroller & Assembly Language Report

Name: **Student Number:**

Working with:

Code submitted by:

I certify that ALL of the following are true:

1. I have read the *UCD Plagiarism Policy* and the *College of Engineering and Architecture Plagiarism Protocol*. (These documents are available on Brightspace.)
2. I understand fully the definition of plagiarism and the consequences of plagiarism as discussed in the documents above.
3. I recognise that any work that has been plagiarised (in whole or in part) may be subject to penalties, as outlined in the documents above.
4. I have not already submitted this work, or any version of it, for assessment in any other module in this University, or any other institution.
5. The work described in this report is all the original work of my team, and this report is all my own work, except where otherwise acknowledged in the report.

Signed: Aidan O’Sullivan **Date:** 9 March 2021

Introduction

In this assignment, we were asked to design an instrument capable of measuring voltage and frequency. The signals were both generated and measured on the ADuC841 microcontroller board, with data displayed in real time on the MAX7221 7 segment display via SPI communication. A variable voltage signal could be produced using an on-board potentiometer, that works by creating an adjustable voltage divider. An ADC samples this signal and averages the signal over a set time. The design was also capable of updating the maximum and minimum voltage over a set period. The frequency function used hardware generated signals with known parameters, which were measured using the counting function on the on-board timers. As the signal cycle counter was accumulated over time, this also introduced an averaging effect to the result. A hardware user interface was developed that allowed different modes to be selected using switches located on the microcontroller, with corresponding sets of LEDs that would light up to indicate what mode was currently in use. The total data memory used by this project was 49 bytes and the total program memory was 1818 bytes.

Several hardware blocks were identified and divided between both group members. As voltage measurement was given the most priory this was the first to be focused on, Josh took care of this block and added the extra feature of maximum and minimum measurement. As there was no voltmeter available to the group it was decided that the display block should be designed in conjunction with the voltage measurement. This meant that when it was operational the voltage could be displayed to confirm that the correct values were being measured. Aidan dealt with this part. It was also decided that Aidan would design the frequency measurement block. The integration of the system and user interface overview were discussed by both members to decide on the design and the implementation was carried out by Josh initially then Aidan helped to finish it off. In this way the workload was split evenly and made manageable for each member.

Program Description

User Interface

A user interface allowed the user to select specific modes of measurement. This was achieved using on-board switches. Corresponding sets of LEDs lit up as an extra feature to inform a user on what mode was currently set up.

<u>Voltage Measurement:</u>

Select: P2.0	LEDs: P0.0

<u>Voltage Maximum:</u>

Select: P2.1	LEDs: P0.1

<u>Voltage Minimum:</u>

Select: P2.2	LEDs: P0.2

<u>Frequency Measurement:</u>

Select: P2.3	LEDs: P0.3

The frequency generated could be selected using P2.6 and P2.7, giving four separate options.

P2.6	P2.7	Frequency (Hz)
1	1	200
1	0	1963.636364
0	1	568.4210526
0	0	7200

Frequency

One main function of the instrument was to measure frequency. In the provided SigGenFunctions.c file the program instructed the microcontroller to generate 4 different frequencies, ranging from 200 Hz to 7200 Hz. These could be selected individually using switches 6 and 7, with each permutation giving a different signal. Timer 0 was used for generation. The associated header file was included in the full system measurement file, allowing the functions, macros and variables to be shared between both .c files. To measure the frequency of the signal, two timers were used. Timer 1 was used to count the edges where a 1-0 transition occurred, causing the TL1 and TH1 counters to be incremented. Meanwhile Timer 2 controlled the time at which the counter value would be extracted and then reset. This could be divided by the counting period to give the frequency to be displayed. Five digits on the display could be used to display the numerical frequency value, with accompanying “Hz” units. An extra design feature was added, whereby a frequency input that was outside the range produced an error warning on the display.

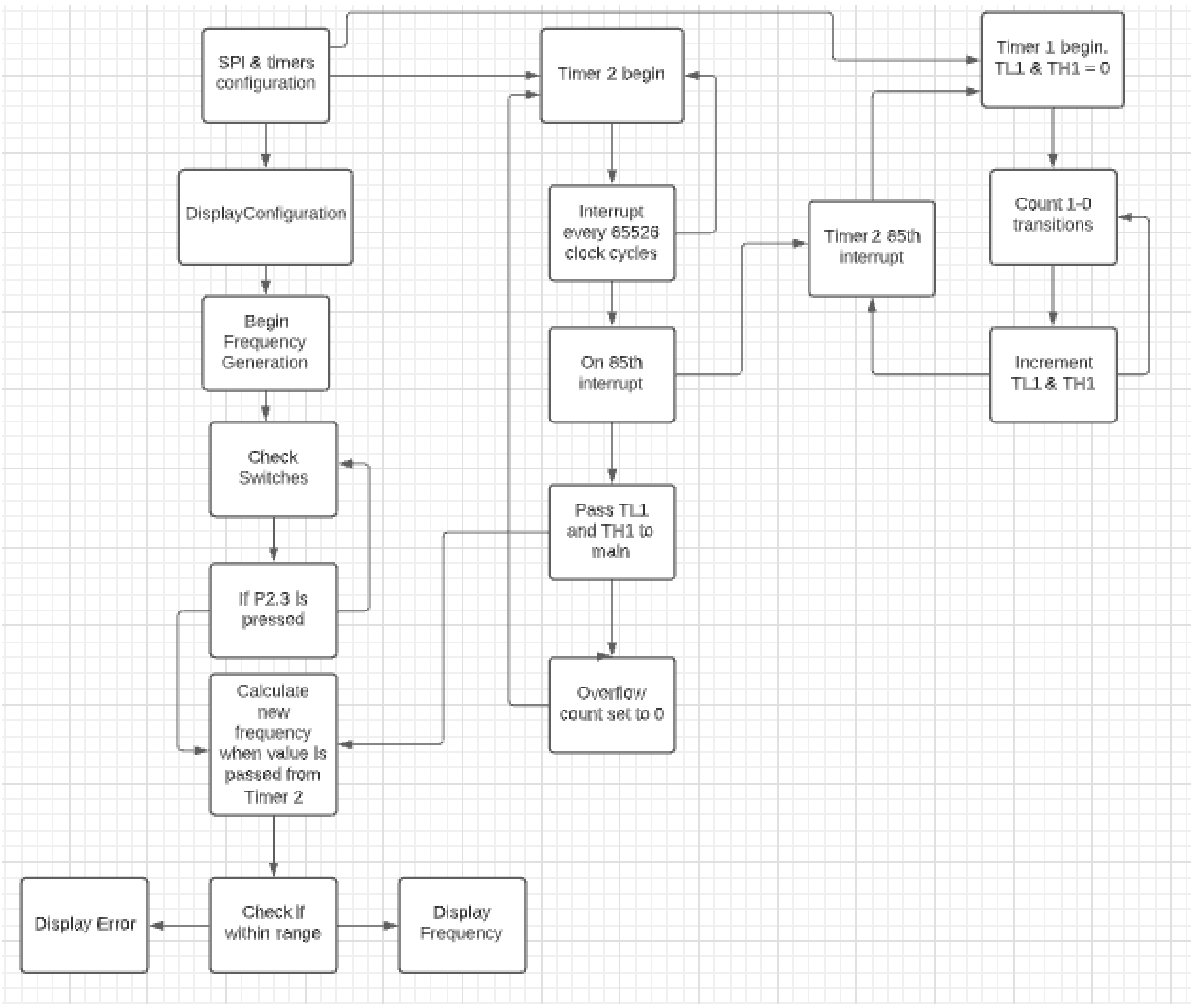

Figure 1 shows a flow diagram of frequency measurement and display.

Timer 1

For frequency measurement, timer 1 was set up as a 16-bit counter. It counts the edges of the signal on a 1-0 transition. An overflow never occurs for the timer as it is reset every 0.5037 seconds by timer 2, by which time it will never reach its maximum count value of 65536, for the frequencies being tested for this experiment. If this experiment were to be expanded for much higher frequencies, timer 1 would set the limitation on the highest measurable frequency. Assuming a standard where a new measured value should be generated at the same averaging time, theoretically the max frequency is:

$$Max\ Frequency = \frac{\max Timer\ 1\ count}{averging\ time} = \frac{65536}{0.5037} = 130\ kHz$$

Any frequency below 130 kHz would be able to be measured, with TL1 and TH1 extracted and reset to zero before an overflow had occurred. For the purpose of this experiment the maximum frequency that will be displayed is 65,536 Hz.

Timer 2

The purpose of timer 2 is to set the time over which timer 1 counter is averaged and the interval between the refresh of the display with the new measurement value. The time interval selected was around 0.5 seconds, as this would give a good real time display, while the changes were not so rapid that they could not be interpreted by the human eye. As this time was too long to be achieved in one timer overflow, several interrupts had to be carried out before the Timer 1 count value would be extracted, even with the timer being reloaded to zero each time. The number of interrupts were calculated as follows:

Total Cycles in 0.5 seconds = $\frac{0.5\ seconds}{clock\ frequency} = 5529600$

Total interrupts = $\frac{5529600}{cycles\ per\ interrupt} = \frac{5529600}{65536} = 84.375\ interrupts$

As this was not an exact number of interrupts it caused an inaccuracy in frequency measurement. Instead, an exact number of interrupts close to this was chosen. It was decided that 85 interrupts would occur before the count value was extracted to calculate the frequency, giving a period of 0.5037 seconds.

On the 85th interrupt the count value was passed into a global variable to calculate the frequency. Timer one counting registers were also rest to zero to begin calculating the next averaged frequency measurement and the write to display flag was set. This flag meant that the display was only written to once when a new frequency value was calculated, improving efficiency and use of the processor.

A commented test line is added into the timer 2 ISR. Its purpose is to check to see if a frequency value exceeding the limit of 65,536 Hz will show an error message on the screen. This is necessary as there is no high frequency option in the signal generator.

Display

The display is communicated with over an SPI line, which is configured to idle low and at a bit rate of the internal clock divided by 16. The 'SPI_transmit' function was used to send data across the SPI line. Two 8-bit variables are inputted to the function. The address of the register the data will be sent on the SPI register SPIDAT first and the actual data is sent second. In this way the MSB in the register will be the address and the LSB will be the data. Load is put low to begin the transfer. In between the transfer of the address and data time must be taken to wait for the ISPI flag to be set, indicating that the transfer has been successful. The flag is then cleared, and a small waiting time added, using a for loop to meet the requirement that DIN must be stable for 25 ns before the next transfer occurs. When all the data has been shifted into the 16-bit register load is set high, causing the data to be accepted by the display.

To configure the display different registers are written to as described, where intensity, number of digits to be scanned and different modes of operation can be selected. Decode mode is used mainly when writing numerical values but sometimes it is switched off to write certain letter in units or error messages. Therefore, the decode mode is dealt with the 'write_to_display' function instead of in the 'Max_setup' function. It requires special settings according to what should be displayed. Registers 1-8 are used to write what will be displayed on-screen. To ensure the 'write_to_display' function is only entered once per generated measurement the 'write_to_display' flag is cleared at the end of the function. This prevents the processor from repeatedly demanding the SPI to display communication that only needs to be carried out once and improves efficiency. The function will also write the error message "Err" to the display if the maximum frequency of 65336 is exceeded. This is an arbitrary number and is chosen for experimental purposes, to show proof of concept.

Results

Switch Values	**Generated Frequency (Hz)**	**Measured Frequency (Hz)**	**% Error**
11	200	200	0
10	1963.636364	1963	0.0324
01	568.4210526	567	0.074
11	7200	7200	0

The results show that the measurements were quite accurate, with only a small rounding error that is introduced because we are dealing with integers and not doubles or floating point. The errors exist in only two cases and at highest is 0.074%, which is negligible.

In the timer 2 ISR, any value of test variable ‘count_meas’ that when divided by the period 0.5037 seconds and produced a frequency greater than 65,536 Hz was also tested to give an error message on the display.

Voltage

The dual purpose of the instrument was to make voltage related measurements. This was carried out by sampling a variable analogue voltage signal using the on-board 12-bit ADC. The voltage could be varied using a potentiometer. As we did not have access to a voltmeter to know the true value, it could be tested by displaying the voltage on screen. When the potentiometer was turned to its minimum it should read close to 0 V and when it was turned to its maximum it should read close to 2.5 V. This is the range of voltages capable of being produced by the potentiometer and it is assumed that they can be varied linearly by turning the knob. The voltage samples were averaged over a block of 1000 samples before a new value was displayed, to give an accurate representation of the true voltage value. It was decided that the voltage would be scaled to mV, to avoid using computationally heavy double data types and for ease of display.

ADC

The ADC sampled the through the analogue pin ADC0, which was wired to the potentiometer. A maximum acquisition time of 4 ADC clock cycles was set to provide the most accurate results. The ADC sample rate was set by timer 2, which could be controlled using the reload value to achieve a suitable period. The Timer 2 clock was then divided by 32. A refresh rate of as close as possible to 0.5 seconds was used again. The timer 2 reload value was calculated as follows:

1000 samples were required. The ADC requires 16 ADC clock ticks to sample, plus acquisition time, and the acquisition time is set to 4 ADC clocks. Therefore, one sample takes the ADC 20 ADC cycles.

The ADC clock is 1/32 of the internal clock.

1000 * 20 * 32 = 64000 internal clock ticks of sampling time

64000 ticks take about 5.787ms

We want to introduce delay in between sampling so the whole 1000 sample cycle takes 500ms

500 - 5.787 = 494.213 ms of delay needs to be introduced over 1000 samples.

Delay in between each cycle will be 494.3/1000 = 0.494213ms

We know that seconds * (cycles per second) = cycles, so

0.494213 * 10^-3 * 11.0592 *10^6 = 5465.6004096

We need 5465.6004096 cycles of delay in between each sample.

To get the T2 overflow value, we subtract that from the max value, 65536.

65536 - 65465.600409 = 60070.3995904

Round to 60070, which in hex is 0xEAA6.

This was set in the ‘adc_setup’ function. For the max and min and slower refresh rate was needed. It was decided that a new max and min should be acquired every 5 seconds, as 0.5 seconds was too short a time to assess what the max and min values were. The same method was used and the reload value of timer 2 was calculated as 0x4B23.

Voltage Scaling

The calculated average voltage over the relevant time period was then sent to the ‘scale_voltage’ function to be converted into mV and displayed. As there was one less digit needed and the units could not be displayed in decode mode, a separate process in the ‘write_to_display’ function was used. The “m” in mV was also not possible to write on a 7 segement display. As a compromise it was written as seen below, which is a standard used by some to overcome this issue.

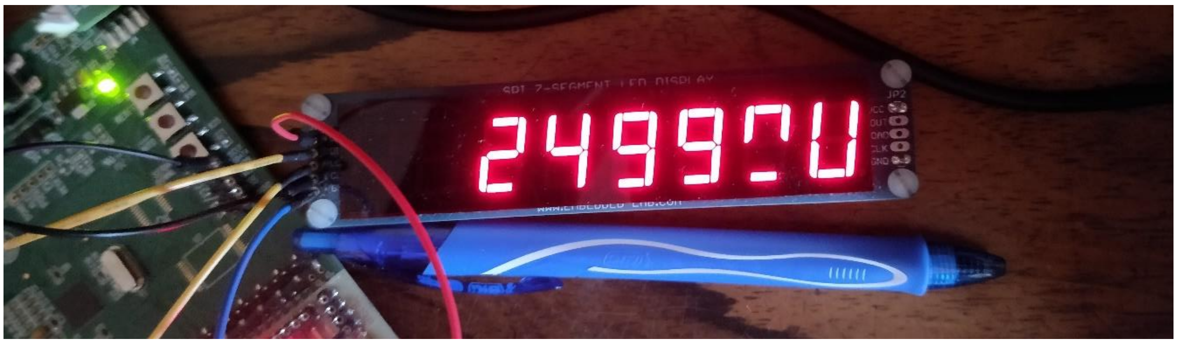

Figure 2 shows voltage measurement with adapted mV units

Results

For voltage measurement when the potentiometer was turned to its minimum it displayed 2 mV and when it was turned to its max it displayed 2499 mV. There is a small inaccuracy present and it may be due to some voltage still passing through the potentiometer when it should be off and not quite reaching the maximum voltage when turned to the max. This could be due to small imperfections in the device but it is quite negligible. As we did not have access to a voltmeter, we could not check the accuracy of the intermediate voltage but through observation they seemed to be varying linearly as expected.

In max and min mode the instrument considered 1000 sampled voltages over 5 seconds before refreshing a new value. The display appeared to confirm this and updated values at the expected time interval.

Conclusion

The instrument could robustly measure the voltage and frequency over the required range with accuracy. The frequency was even capable of measuring frequencies far above the range that was being generated, showing an adaptable design. Familiarity was gained with an array of hardware components, such as timer and ADCs. Integrating many blocks introduced a new challenge and had to be overcome through problem solving as a team to achieve a working final product. Satisfactory results were achieved through the joint contribution of both members. The work was broken down in sections, which allowed for individual focus on specific tasks, while problems encountered were worked upon together in most cases to achieve solutions.