

DESIGN (E) 314
TECHNICAL REPORT

Digital Multimeter and Signal Generator

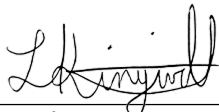
Author:
Luke Kingwill

Student Number:
20725728

April 5th 2022

Plagiaatverklaring / Plagiarism Declaration

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.
3. Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.
4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelike aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.



Handtekening / Signature

LEV Kingwill

Voorletters en van / Initials and surname

20725728

Studentenommer / Student number

01-05-2022

Datum / Date

Introduction

In the design of a new system, the steps and decisions made need to be documented so that they can be followed and possibly recreated for future projects or development. This report aims to document the design and testing of the debug LED and the push button. The report details the hardware decisions made to meet the system requirements and how it is integrated with the software. The software is primarily detailed with the aid of flowcharts. Finally, the testing was conducted on each individual system and then finally the complete system was tested.

Hardware

This section of the report will detail the hardware design for the digital multimeter with specific focus on the design decisions involved with the design of the debug LED (D2) and the middle push button (S3).

The first of which to consider is D2. D2 is used to indicate that the digital multimeter systems menu state, D2 on means that the system is in the menu state and visa versa. The menu state will change during different operations of the digital multimeter and so D2 is connected to one of the microprocessors GPIO pins. Table 1 shows the GPIO pins that were chosen for the 4 debug LEDs. These pins were chosen because of their connector pin position, J5, is close to LEDs and therefore keep the wiring relatively simple.

Table 1: LED pin connections

LED	CPU Pin	Connector Pin
D2	PB12	J5-15/16
D3	PB15	J5-25/26
D4	PB14	J5-27/28
D5	PB13	J5-25/26

The second design decision was choosing the resistor for the LED circuit. Because the LED has very low resistance, having only an LED connected to the development board would have caused a large current flow out of the development board which can damage the development board. A resistor connected in series with the LED is used to reduce the current out of the development board and through the LED.

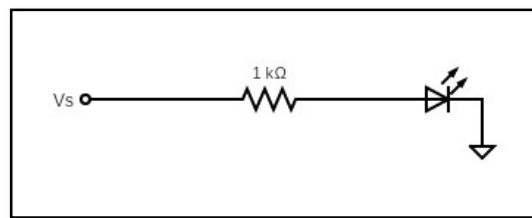


Figure 1: Circuit Diagram for LED Circuit

Per the datasheet the development board can source an absolute maximum of 25 mA. However, because there are 4 LEDs and 5 push buttons that the development board will need to source, together with the fact that LEDs would require very little power to be bright enough, a design current of 800 μ A was chosen. A further assumption is the voltage drop across the green LED is 2.2 V. Given that the supply voltage is 3V, equation 1 can be used to calculate V_R , the voltage drop across the resistor and equation 2 can be used to calculate the value of the required resistor.

$$V_R = V_S - V_{LED} \quad (1)$$

$$R = \frac{V_R}{I} \quad (2)$$

The voltage drop across the resistor was calculated to be 0.8 V. Moving on to equation 2 as previously mentioned, the value of the resistor to be used in series with the LEDs was calculated to be 1 k Ω .

Secondly, is the design of S2, the middle push button. Where the LED was set up as a output, the push button was setup as a input with interrupt. The need for an interrupt influenced the pin selection as there are several pins connected to each interrupt (PA1, PB1, PC1 are all connected to EXTI1). Similarly, to the LEDs, the pins selected for the push buttons S1 through S5 were chosen based on ease of wiring and pin availability. The pins chosen for the push buttons is shown in table 2.

Table 2: Push button pin connections.

Push Button	CPU Pin	Connector Pin
S1	PA6	J11-11/12
S2	PA7	J11-13/14
S3	PA10	J9-11/12
S4	PA9	J11-19/20
S5	PA8	J9-1/2

Similarly, to the LEDs, a resistor is needed in series to protect the development board. However, unlike the LEDs, the voltage drop across the push button can be considered negligible and no power is required to power the push button and so it can be operated with a very small current and therefore the series resistor will be much larger. However, instead adding a physical resistor as with the LEDs an internal resistor on the development board can be used to reduce the number of components required. The internal resistance was configured as a pull up resistance, this is shown in figure 2.

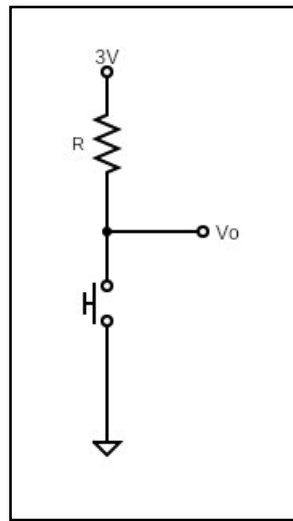


Figure 2: Pull-up switch configuration.

A pull up resistor configuration means that the signal is high while the switch is open. The output voltage is taken between the pull-up resistor and the switch. While the switch is open the output voltage can be calculated using equation 3. When the switch opens, a short circuit is created, and the output voltage is pulled down to ground.

$$V_o = 3V - I \times R \quad (3)$$

Software

Button debouncing can be resolved with hardware or with software. In this project it was decided to resolve the button bounce with software as it would be impractical to build a debounce circuit for all the buttons. It would take up too much surface area on the PCB and it would take unnecessary time.

For our uses, we will not need to differentiate between single click and multiple clicks and so we do not need to use a click counter and can simply use a delay so that the interrupt is only called once for a button click. The delay will be used inside the interrupt request handler, slowing down the processing of the interrupt and therefore once the initial button press interrupt has been processed, the button will have finished bouncing. This is because the delay is before the interrupt can be cleared and so no new interrupt flag can be set, and all further bounce is simply ignored.

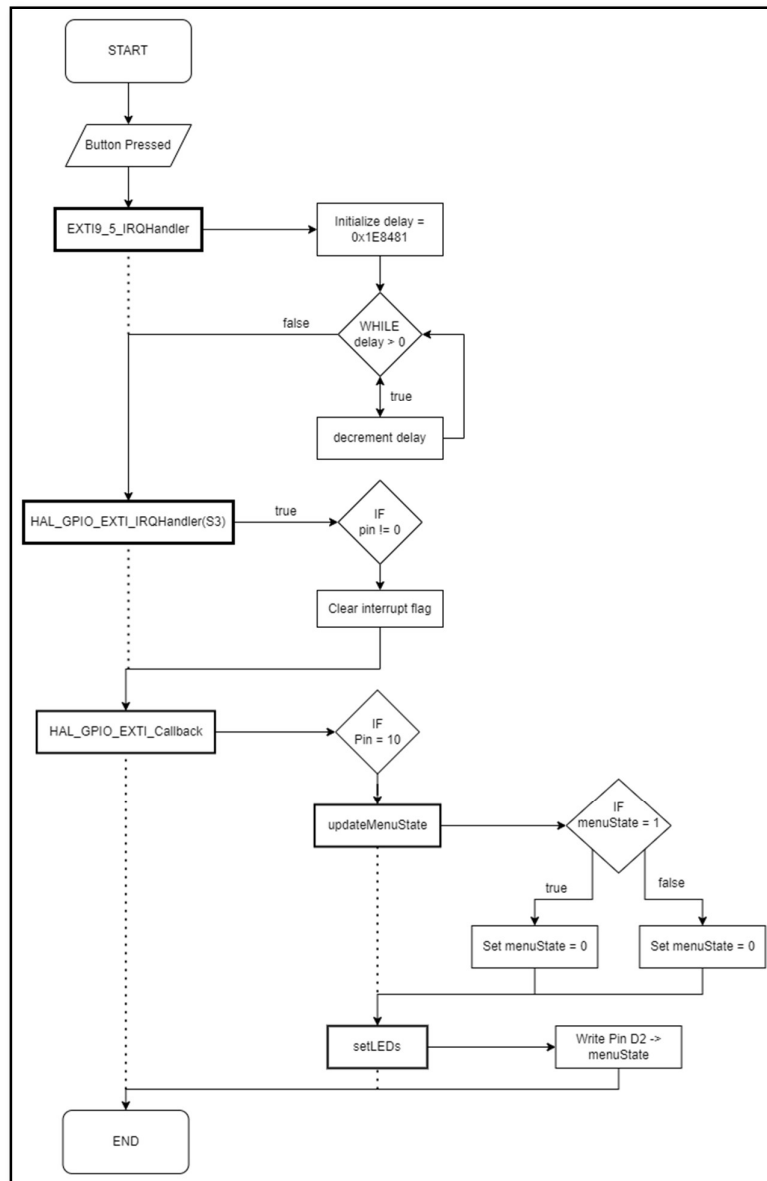


Figure 3: Button Pressed Flow Chart.

In most areas of the software, a simple *HAL_Delay* can be used to delay the program for the required time. However, because the *HAL_Delay* is calculated using the *SysTick* timer, calling the delay inside of an interrupt can cause issues. Instead, a brute force delay is implemented. By creating a large number and counting the number down in a *While* loop, the program is delayed while it must process the calculation. An initial value of 0xFFFF (65535) which can be related to an equivalent delay of around 50 ms, was used. However, this delay was too short and was then fine tuned with trial and error until a final value of 0x1E8481 was used.

Figure 3 shows the flowchart of the whole button press process. The thickened borders highlight function names and while the dotted lines show the step-over flow, the branches to the right of the functions show the step-into flow through the program. Much of the code to trigger the interrupt of button is generated code while the last few functions have been created to serve the button press process.

Testing

Testing is naturally a highly important design step. There were several techniques used to test the designs that were implemented on the development board using tools such as an oscilloscope, multimeter and visual confirmations.

Firstly, considering the LED there were a couple of assumptions used in the design of the LED circuit. These assumptions needed to be confirmed to ensure that the design was sufficient and that the LED circuit meets the requirements. The initial assumption of the voltage drop across the LED can be confirmed by using a multimeter to read the voltage across the LED. Like the design of the LED circuit, equation 1 can be used to calculate the voltage across the resistor. Equation 2 can then be used to calculate the current being sourced by the development board. The measured and calculated values are shown in table 3 in comparison to the values used in the design of the LED circuit. The current is well below the maximum allowed 25 mA and so the design requirements have been satisfied. Finally, a visual check was used to ensure that the LED had enough power to be comfortably visible to the human eye as this is the required use for the LED. The brightness of the LED can be seen in figure 6 and it can be clearly seen that the LED is on while the other LEDs are off.

Table 3: Design Test Values for LED Circuit Parameters.

Parameter	Design Value	Test Value
V_{LED}	2.2 V	
V_R	0.8 V	
I	0.8 mA	

Secondly, the push button debouncing needed to be tested. An oscilloscope was used to measure the mechanical bounce of the push button and is shown in figure 4. This bouncing is what needed to be flattened out to ensure that the button click is only read once. Due to the way the debouncing was implemented, using software, it is much harder to measure the bounce post debouncing. Instead, a software check was implemented by using a 'while loop' and setting an output pin to the value of input pin of the button. This was implemented so that the output pin would mirror the behaviour of the debounced input. The output was also measured using an oscilloscope and the results are shown in figure 5. It can clearly be seen that the button-press in figure 5 is much cleaner than that of the bouncing button in figure 4.

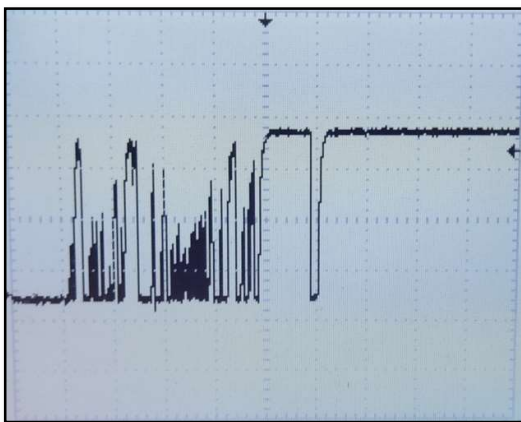


Figure 4: Bouncing Signal of Push Button

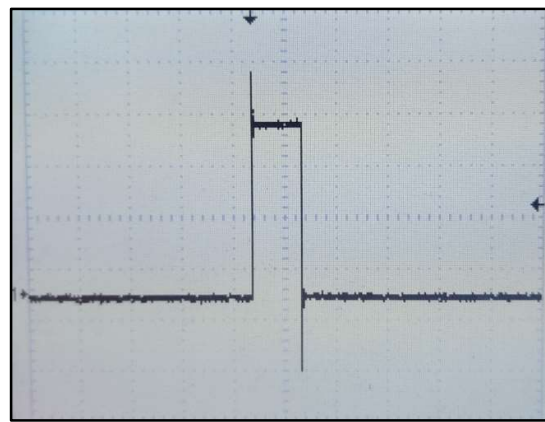


Figure 5: Debounced Signal of Push Button

Finally, to ensure that the whole system was working together with each other, a visual test was conducted once the software and hardware were all implemented. The visual check consisted of repeatedly pressing the middle push button for varying durations and varying frequency. The LED was monitored to ensure that it did not flicker on a single button-press and that all button-presses were picked up within reason. At a very high frequency of button-presses, the LED can sometimes miss some of the inputs as the delay is still being implemented and the interrupt flag has not been reset. However, this was deemed to be within the acceptable limits as the buttons would not realistically be used in that frequency. Figure 6 shows the brightness of the LED during the visual system tests.

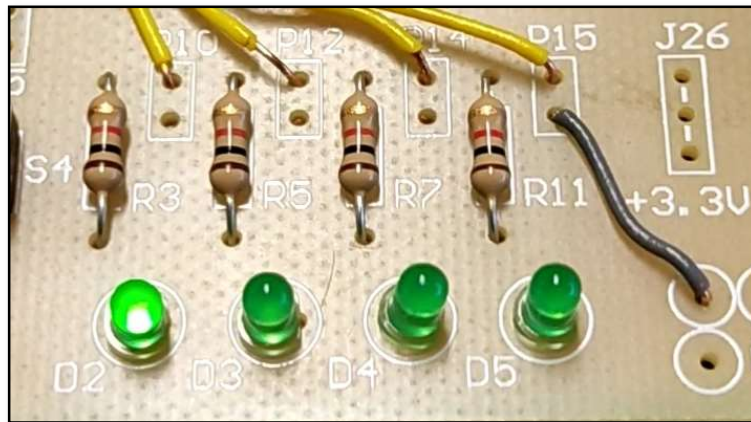


Figure 6: LED D2 on.