Lab 2 Report

Authors:

Christopher Andrade, 2221525
Theo Favour, 2169814

Date: 7/5/2024

Introduction

This lab primarily concerned the implementation of LED control using non-blocking methods (directly interfacing with the registers of the ESP32 instead of simply using Arduino library functions). Implementing these methods increased the speed of the ESP32's responses, and required us to utilize things like bitmasks and timers along with researching the different registers in the ESP32's documentation.

System Design and Implementation

approach, the rationale behind your design decisions, and how these decisions help solve the programming challenges identified.

Our system focuses on completing simple tasks using different methods of interfacing with the ESP32. We began by comparing differences in speed between two programs that accomplish the same task of flashing an LED using different methods - Direct register access and Arduino library functions. Afterwards, we performed specific tasks such as using a timer to toggle an LED and reading analogue data to smoothly manage an LED's brightness or play a sequence of notes through a buzzer. One major design decision we made was to use bitmasks when editing register values in the ESP32, which is a key part of utilizing the registers; many have reserved bits that should not be changed when interacting with them.

Code Structure

Part I

Part one consisted of controlling an off-board LED through the use of Arduino macros corresponding to addresses within the ESP32 which determine the functionality of the board's pins. Step two consisted of a single arduino file with a setup function that initialized pin D2 as a GPIO output pin and a loop function which writes digital high and low with appropriate delay to allow for D2 to flash an off-board LED. Both functions were implemented by manipulating bits in registers GPIO_Enable and GPIO_Out which correspond to D2.

Step 3 aimed to measure the difference in performance between the digitalWrite Arduino Library function and Direct Register access to change the HIGH/LOW status of an output pin. Both approaches were contained within their own separate .ino file and contained identical setup functions which simply initialized the D2 pin as a GPIO output. However, the files differed within the loop function which toggled the D2 from HIGH to LOW 1000 times, one using digitalwrite and the other masking a 0 and 1 into the GPIO_Out register. Both implementations are then timed and the resulting length of time (to switch between HIGH and LOW output 100 times) is then printed to the Serial Monitor.

Part II

Part two concerned extensive use of registers to create a timer which was then utilized to blink an off-board LED on and off at one second intervals. Similarly to part 1, within the setup function a single pin (D2) was initialized as a GPIO output however the setup function also contained necessary bit shifting and masking to enable a timer of 1 MHz utilizing registers corresponding to timer 0 of group 0. This timer is then utilized within the loop function to determine if enough time had passed to toggle the state of the D2 output and therefore off-board LED. In the case that enough time has passed the GPIO_Out register has its D2 bit toggled.

Part III

This section of the code implements ambient light detection to automatically change the brightness of an external LED. The relevant functions are those which handle analogue input - namely ledcSetup and ledcWrite. We use the former to set up a pulse width modulation channel, of which we alter the duty cycle using the latter function in order to change the brightness percentage of the LED, updating with a frequency of 1KHz.

Part IV

Part IV of the code utilizes similar methods from part III in order to implement a buzzer which sounds a three note cycle when the lighting is below a certain threshold, and plays nothing otherwise. We use the same functions ledcWrite and ledcSetup in order to again initialize a pulse width modulation channel that reads ambient light, along with ledcAttachPin to assign the peripherals to the specified pin. The key difference from part III is the use of four arbitrary duty cycle values to create a four note sequence, with blocking delays inserted between since the sequence is guaranteed to play in full each time the threshold is met.

Discussion

In step three of part one two different approaches were used to toggle the output of the D2 pin, one used the digitalWrite function from the Arduino Libraries and the other used direct register access. Each approach toggled the output 1000 times and was timed in microseconds. This displayed that directly accessing appropriate registers was roughly twice as fast as using Arduino Library functions (ALF) (6973µs and 3013 µs respectively). Direct register access (DRA) is much faster and will allow for faster switching between HIGH and LOW outputs however it also involves casting, dereferencing, bit masking and bit shifting. While the digitalWrite function simply takes two parameters, the pin being manipulated and the value it should be changed to. Therefore, while DRA is faster it also requires more code, understanding and is less readable in comparison to simply using digitalWrite from ALF.

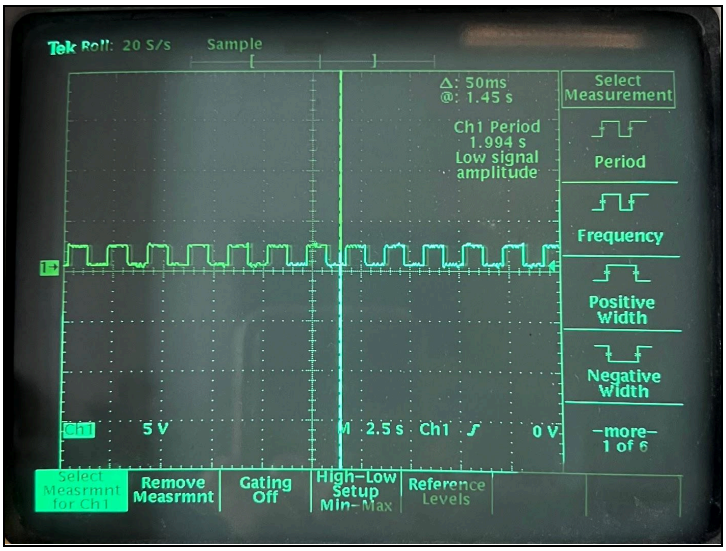

Fig. 1. Oscilloscope reading concerning part two.

To measure our results for part II of the project, we connected our ESP32 to an oscilloscope in the ECE lab. After calibrating the correct settings, we were able to see the signal from our board and measure the period of the LED's flashing to be 2 seconds, with a margin of error smaller than 10 milliseconds as seen in Fig. 1.

A major challenge we faced related to the editing of the timer registers of the ESP32 for part two of the lab. In doing so, we often found that certain random bits in the divider section of the register would be different than we intended and expected. In fixing this bug, we came together multiple times and tried different strategies; we first tried debugging together, then researching the timer registers further within the ESP32's documentation, and finally bringing our challenges to class to ask for input from the course staff. In doing so we eventually found a fix based on saving the bitmask as a distinct variable before using it to edit the timer register.

Our main approach for division of work was based on strong communication; since we know each other and speak to each other every day, we are able to get quick updates and give honest feedback while reviewing each other's work. We ended up separating the different parts of the assignment, and reviewing the different code sections together, and when issues arose we attempted to work through them in conversation together which resulted in stronger and more efficient debugging.

Division of Work

Chris Andrade was responsible for creating the code for parts one and two, and aided in the creation of the lab report. Theo Favour contributed significantly in developing code for parts three and four, debugging the timer registers in part 2, and creating the lab report. Both members contributed an approximately equal amount of time and effort to the overall project (50% each).

Suggestions

We faced multiple challenges while working through this assignment, mostly based on either vagueness of the specifications or of the ESP's own register documentation. A clearer explanation on how to appropriately initialize the counter in part two would have saved a great deal of time which was lost to trying multiple different approaches and not understanding the behavior of the timer configuration register. However, the code given in this same step which used many Arduino Library functions was extremely helpful in developing its corresponding direct register implementation to achieve expected functionality.

Additionally, the given breadboard example circuits might be a bit unclear to many students within this class. With the only prerequisite of this introduction to Embedded Systems class being an introductory Java series it is expected that some students have no breadboard experience. The example circuits contain extensive use of jumper wires and result in a "schematic" which may be confusing to many beginners, especially when multiple wire ends appear to terminate in the same hole.

Conclusions

This project successfully implemented several distinct functions of the ESP32 while using and analyzing different methods such as DRA and ALF. By completing the project, we both strengthened our basic abilities in managing the ESP32 with library functions and increased our understanding of the inner workings of the ESP32 and the different registers, such as timer registers, within it. We also increased our familiarity with some peripheral types such as those which read analogue input, along with LEDs and buzzers. This project serves as a base of understanding and practice for us with the boards and their documentation.

References

ESP32 Technical Reference Manual (Version 5.1), Arduino, 2024, www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

Appendices

Total Number of hours Spent

21 hours.