Brandon Davis
11/01/2024
45045888

Report – Driver Lab

Introduction

In this lab, we focus on developing for the Raspberry Pi RP2040 platform using the Pico SDK. This will involve writing C/C++ code that takes full advantage of the architecture and interfaces provided by the SDK. We will also delve into the CMake Configuration System, essential for managing our project builds effectively.

Our primary objectives include creating two well-engineered drivers that encapsulate the complexity of hardware interactions, thereby providing intuitive, safe, and robust interfaces for application software. Specifically, we will develop an I2C driver for the LIS3DH accelerometer and a NeoMatrix display driver. These drivers will serve as libraries that abstract the low-level implementation details, allowing for easier integration into larger applications.

By the end of this lab, we aim to enhance our understanding of embedded systems, improve our coding skills, and gain practical experience in creating reliable software drivers for hardware components. This hands-on experience will not only solidify our theoretical knowledge but also prepare us for future challenges in the dynamic field of embedded systems development.

Design

The design of this lab revolves around implementation of two device drivers and using them in a simple application called a “Bubble Level” example.

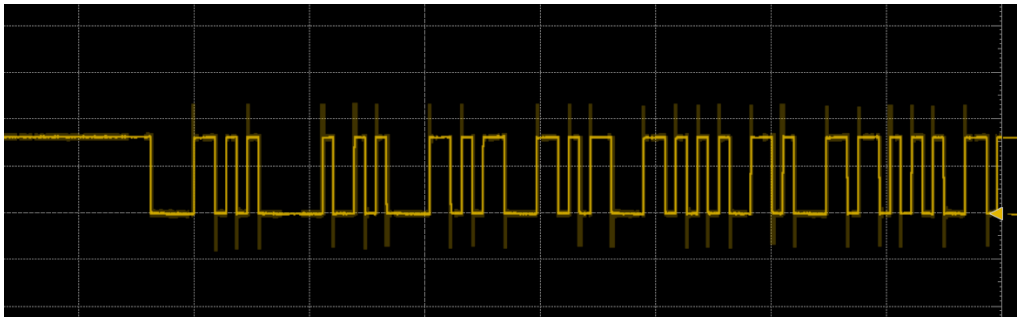
1. What challenges did you encounter when implementing your driver libraries? How did you overcome them?
    a. **Understanding Hardware Specifications**:
        i. **Challenge**: The initial hurdle was comprehensively understanding the hardware specifications and communication protocols (I2C for the LIS3DH). The datasheets contained detailed technical information that was critical for successful implementation.
        ii. **Solution**: I dedicated time to studying the datasheets and relevant application notes. I also referred to community forums and example projects to gain practical insights into the usage of the components.
    b. **I2C Communication Issues**:
        i. **Challenge**: During the development of the I2C driver, I faced issues with inconsistent data readings, which suggested potential problems with communication setup, such as incorrect addressing or timing issues.

ii. **Solution**: I utilized a logic analyzer to monitor the I2C communication between the microcontroller and the accelerometer. This allowed me to verify the signal integrity and identify misconfigurations. I made adjustments to the timing and ensured that the correct I2C address was being used.

c. **Debugging and Testing**:
   i. **Challenge**: Debugging embedded systems can be challenging due to limited debugging capabilities compared to desktop applications. This made it difficult to trace errors effectively.
   ii. **Solution**: I employed systematic debugging techniques, such as adding extensive logging and using breakpoints when possible. I also created a set of unit tests for each driver function to ensure their correctness and reliability.

2. Do you have any suggestions for improvement to the structure of these libraries?
   a. Building the Project
      i. Make File Specification in the documentation was sparse, I feel like if there was more guidance in creating the cmakelists file the project would be done faster and be a more engaging experience.
   b. LISDH
      i. Adding a temperature sensor. The backpack used for the accelerometer also has a temperature sensor, so it would be useful to have an update function for the temperature reading it via I2C
      ii. Adding error handling to the init function for debugging purposes
   c. NeoMatrix
      i. Adding a deconstructor to the library due to use the heap when calling the new keyword; this would benefit from not having memory leaks throughout the code.
      ii. Some of the internal variables of the code are exposed and should be kept private like the double pointer pixel Buffer

3. Please include one oscilloscope screen capture for the I2C bus and NeoMatrix driver line, with a brief explanation of their operation.



i. The Start condition is defined as a high to low transition on the data line while the SCL line is High. Once a transmission has occurred, the I2C master bus will be busy until the data is transmitted. Then the next byte is transmitted after the start contains the address of the slave device in the first 7 bits and the 8th bit is to decide whether to read or write. When the address is sent each device compares the 7 bits after the start address, and if it's a match the device considers itself addressed from the master device. The next byte is the data that is written or read from the device.

Conclusion

The development of the NeoMatrix and LIS3DH driver libraries provides valuable insights into embedded system design and implementation. Through this process, we have created functional drivers that facilitate interaction with hardware components, enabling a deeper understanding of how software communicates with physical devices.

However, this exploration has also highlighted several areas for improvement within the libraries' structure and usability. Enhancing memory management, improving error handling, and encapsulating internal details can significantly increase the robustness and reliability of these drivers. Additionally, utilizing standard containers, allowing for configuration options, and providing thorough documentation will make the libraries more flexible and user-friendly.

By implementing these suggestions, we can create a more efficient and maintainable codebase, ultimately preparing us for the complexities of real-world embedded systems development. This experience not only reinforces our technical skills but also underscores the importance of thoughtful software architecture in the creation of reliable and effective embedded applications.