Architectures for Embedded Systems

Serial interfaces
RS232, SPI and I2C
Questions for autonomous study
Device drivers for I2C and SPI sensors
Laboratory assignments

Arnaldo S. R. Oliveira

Academic year 2024/25

Universidade de Aveiro – Dep. de Eletrónica, Telecomunicações e Informática

Outline

Serial interfaces (autonomous study questions)

- RS232
- SPI
- I2C

Lab assignments

- Device driver for the I2C TC74 sensor
  - Analysis and test of a demonstration program
  - Visualization of the I2C transfers
- Device drivers for the I2C/SPI BME280 sensor
  - Development of the device drivers (I2C and SPI)
  - Creation of demonstration applications
  - Visualization of the I2C/SPI transfers

Autonomous Study Questions

1. What distinguishes serial and parallel interfaces in a computer system?
2. What are the advantages of serial interfaces?

For RS232, SPI, and I2C serial interfaces (separately):

3. What is the purpose / preferred use?
4. What signals make up the interface?
5. How is it characterized regarding communication (bi)directionality (simplex, half-duplex, full-duplex)?
6. Is there symmetry between the communication ends, or is it a Master/Slave configuration?
7. What is the (typical) maximum transmission rate (bits per second)?
8. What voltage levels are used (for each logic level)?
9. How is a data frame organized? How are the bits arranged in a transmission?
10. How is synchronization achieved in data transfer? What type of clock is used?
11. What configuration parameters need to be programmed?

Laboratory Assignment 1 – Visualization of RS232, SPI and I2C transfers with the oscilloscope

- Create a simple program that performs, in infinite loop, a basic transfer using each one of the serial interfaces – a separate program for each one (RS232, SPI and I2C)
- Compile and test the project
  - With the help of the oscilloscope visualize a complete data frame (corresponding to a simple transfer) and interpret the transmitted bits
- Sources of information
  - RS232 UART - https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32c3/api-reference/peripherals/uart.html
  - SPI - https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32c3/api-reference/peripherals/spi_master.html
  - I2C - https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32c3/api-reference/peripherals/i2c.html

Laboratory Assignment 2 – Analysis and test of a device driver for the TC74 I2C temperature sensor

1. Analyse and test the TC74Demo program containing:
   - A main application (TC74Demo.c)
   - A device driver for the I2C (temp_sensor_tc74.h and temp_sensor_tc74.c)
2. Create a new project, with a simple main function to test “each” device driver function with the help of the oscilloscope
   - Call individually “each” function in infinite loop to allow the visualization of the I2C transfers (SCL and SDA signals)

Laboratory Assignment 3 – Development and test of a device driver for the BME280 sensor (**I2C**)

- Create a device driver for the BME280 humidity + barometric pressure + temperature sensor, using the **I2C** interface
  - Follow the same approach as with TC74
  - Define the interface for the device driver (bme280_sensor_i2c.h)
  - Define the implementation for the device driver (bme280_sensor_i2c.c)
- Create a simple main function to test “each” device driver function with the help of the oscilloscope
  - Call individually “each” function in an infinite loop to allow the visualization of the **I2C** transfers (SCL and SDA signals)
- Create a new project with a demonstration application for the sensor (similar to the one provided for the TC74), but supporting humidity, barometric pressure and temperature)

Laboratory Assignment 4 – Development and test of a device driver for the BME280 sensor (**SPI**)

- Create a device driver for the BME280 humidity + barometric pressure + temperature sensor, using the **SPI** interface
  - Follow the same approach as with I2C driver for the same sensor
  - Define the interface for the device driver (bme280_sensor_spi.h)
  - Define the implementation for the device driver (bme280_sensor_spi.c)
- Create a simple main function to test “each” device driver function with the help of the oscilloscope
  - Call individually “each” function in an infinite loop to allow the visualization of the **SPI** transfers (SS\, SCLK, MOSI and MISO signals)
- Create a new project with a demonstration application for the sensor (similar to the one provided for the TC74), but supporting humidity, barometric pressure and temperature)

Fundamental Questions for Discussion

Electrical aspects:

- How the BME280 sensor switches between I2C and SPI modes?
- Which components does the breakout board add “around” the BME280 sensor?
- What is the supply voltage and pin of the BME280 breakout board?
- How to manage efficiently, in the breadboard, the possibility of using the I2C and SPI modes (one at a time)?
- How to visualize the 4 SPI signals with a 2-channel oscilloscope?
- How should a good schematic diagram be organized?

Fundamental Questions for Discussion

Device driver aspects:

- What capabilities does the device driver need to expose to the application?
- Who should manage the calibration of the values? Is the user of the device driver or the device driver itself (internally)?
- When calibration data should be obtained from the sensor?
- What needs to be public in the device driver, and what should be private?
- What should be the contents of the “.h” and “.c” of the device driver source files?
- Where the public “.h” source file of the device driver should be included?
- Which user-defined data types could be useful?
- How can the power consumption of the sensor be decreased? And for the whole system (microcontroller + sensor)?

Final Remarks

- At the end of this week, you should be familiar with the:
  - three serial interfaces (RS232, SPI and I2C) – refresh knowledge from previous courses, e.g. AC2
    - identify the main characteristics of each one
    - write and test programs using the corresponding ESP32 device drivers
    - debug the interfaces with the help of the oscilloscope
  - purpose of device drivers
  - programming of I2C and SPI interfaces
  - visualization of complex signals with the oscilloscope for debugging purposes