ESE381 Embedded Microcontroller Systems Design II

Spring 2022, K. Short   revised April 3, 2022 10:03 pm

PRELIMINARY PRELIMINARY PRELIMINARY PRELIMINARY PRELIMINARY

Laboratory 09: GPIO Expander with I2C (Two-Wire) Interface

This will be a two week laboratory. To be performed the weeks starting April 3rd and April 10th. The prelab and signatures will be due the week starting April 10th.

Prerequisite Reading

(All items are posted on Blackboard.)

1 AVR128DB48 Data Sheet Section 29 TWI - Two-Wire Interface
2. UM10204 I2C-Bus Specification And User Manual
3. Microchip MCP23017/MCP23S17 16-Bit I/O Expander with Serial Interface
4. Microchip AN1043 Unique Features of the MCP23X08/17 GPIO Expanders. (This document covers both the SPI and I2C versions, we are only interested in the I2C version at this time>

Overview

While a microcontroller like the AVR128DB48 has an number of extremely useful modules inside, to take advantage of most of them you have to configure some GPIO pins for their alternative functions. This makes these pins no longer available for use as general purpose IO pins. So, pins become a critical resource and there always seems to be too few of them. A low cost IC that can help in these situations is a GPIO expander.

GPIO expanders provide easy I/O expansion using standard serial interfaces. GPIO expanders are used to provide additional GPIO pins in applications where the microcontroller does not have the needed number of GPIO pins directly available. GPIO expanders are also used to provide remote I/O using a serial interface. The objective of this laboratory is for you to understand the functionality and use of a GPI expander in general and the use of a GPIO expander with an I2C interface in particular. It will also give you the opportunity to design the interface to a read/write I2C slave.

You will use a Microchip MCP23017 16-Bit I/O expander with I2C to provide GPIO pins to connect a DIP switch and bargraph to the AVR128DB48 microcontroller. The connection of the GPIO expander to the microcontroller requires two pins for the I2C interface in comparison to the 16 pins that would be required to directly connect an 8 position DIP switch and 8 LEDs of the 10 element bargraph directly to the microcontroller. Thus, in this case, 14 microcontroller pins are freed up to be used for other purposes by using a GPIO expander.

Design Tasks

Design Task 1: Designing the Interface Connection and Verifying I2C Writes from the AVR128DB48 to the MCP23017.

Draw a schematic diagram of the interface of a MCP23017 to the Curiosity board using TWI0. Connect the DIP switches to pins GPA7 to GPA0 of the MCP23017. Connect the bargraph to pins GPB7 to GPB0 of the MCP23017. On the MCP23017, SDA must be driven by PA2 and SCL must be driven by PA3. PA2 and PA3 are the SDA and SCL signals, respectively, from TWI0. Hard wire A2, A1, and A0 of the MCP23017 to ground. Use external 47 kohm pullup resistors on SDA and SCL.

Note that in your design, the only supply voltage to be used is 3.3V from the Curiosity board.

Write a function named `I2C0_MCP23017_init` that is passed no arguments and returns no value. This function initializes the AVR128DB48's I2C0 to communicate with the MCP23017. The bit transfer rate between the AVR128DB48 and the MCP23017 must be as fast as possible, but less than or equal to 400 kb/s.

Write a function named `MCP23017_I2C_write` that is passed three `uint8_t` arguments. These are the MCP23017 control byte (opcode), slave register address, and data, respectively, for a byte mode write operation.

Write a function named `MCP23017_I2C0_init` that is passed no arguments and returns no value. This function initializes the MCP23017. Port A of the GPIO (GPA) must be configured as all inputs with pull ups enabled. GPB must be configured as all outputs.

Write a test program named `MCP23017_write_test` that uses the previous three functions to verify your program's ability to write the MCP23017. This program continually increments the binary value displayed on the LEDs. Note that the top left pin on the MCP23017 it the least significant bit of port GPB. Use a delay of approximately 1/2 second between incrementing each value displayed. When the displayed value reaches 0xFF it must next rollover to 0x00.

Submit your schematic and program C source file as part of your prelab. Make sure that your program includes a program header, a header for each function, and clear comments throughout.

Design Task 2: A GPIO Implemented Simple 8-bit Parallel Input Port and Simple 8-bit Parallel Output Port

An 8-bit parallel input port and an 8-bit parallel output port are to be implemented using the GPIO. The input port is connected to the 8 position SPST DIP switch. One end of each switch is connected to ground. The input pins for this port must have their internal resistors enabled as pull up resistors to convert each switches' position to a valid logic level. Thus, when a switch is open it will be read as 1 and when it is closed it will be read as a 0.

The output port drives 8 of the LED elements of a 10 element LED bargraph. The other two elements are not used. The cathodes of the LEDs are connected to the corresponding pins of the output port. Current limiting for the LEDs is accomplished using the 330 SIP resistor network.

The basic operation of the system is to continually read (poll) the switches and, based on the logic level read, turn on the LEDs corresponding to the switches that are open (logic 1 outputs).

Write a function named `MCP23017_I2C_read` that is passed two `uint8_t` arguments. These are the MCP23017 control byte (opcode) and register address, respectively, for a byte mode read operation. The function returns the byte read as a `uint8_t`.

Write a test program named `MCP23017_inout_test`. This program continually reads the switches of the input port and uses the information read to turn on the LEDs corresponding to the switches that were open. Note that the order of the pins on port GPA is the reverse of the order on port GPB. That is, the lsb of GPB is at the top of the MCP23017 IC on the left side and the msb of GPA is at the top of the MCP23017 IC on the right side. The bit positions of both ports must be interpreted as having their lsbs at the top, so that the order of the switches and the LEDs correspond when operating the system. You could either do this by wiring GPA to the switches in reverse order or writing a program that reverses the data from the switches before outputing it to the bargraph.

Submit your C source file for this program as part of your prelab. Make sure that your program includes a program header, a header for each function, and clear comments throughout.

Design Task 3: A GPIO Implemented Simple 8-bit Parallel Input Port and Simple 8-bit Parallel Output Port Using MCP23017 Interrupts

The same 8-bit parallel input port and an 8-bit parallel output port from Task 2 are to be implemented using the GPIO. Except, instead of having to poll port GPA of the MCP23017 its pin change interrupt capability is to be used. Use INTA from the MCP23017 to request an interrupt at pin PF3 of the AVR128DB48.

The basic operation of the system is that after configuration an infinite loop is entered. Program execution remains in that loop until an interrupt occurs on AVR128DB48 pin PF3. The interrupt service routine associated with that pin must read the switches and, based on the logic level read, turn on the LEDs corresponding to the switches that are open (logic 1 outputs). After completion the service routine returns and execution continues in the main loop.

Write the ISR for the pin change interrupt at AVR128DB48 pin PF3. Write a test program named `MCP23017_inout_interrupt_test`. When an interrupt at AVR128DB48 pin PF3 occurs, this program reads the switches of the input port and uses the information read to turn on the LEDs corresponding to the switches that were open.

Submit your C source file for this program as part of your prelab. Make sure that your program includes a program header, a header for each function, and clear comments throughout.

Laboratory Activity

Laboratory Task 1: Designing the Interface Connection and Verifying I2C Writes from the AVR128DB48 to the MCP23017.
Wire the MCP23017's I2C interface to the Curiosity board's I2C0 I2C pins. Wire the DIP switch and bargraph LED to the *MCP23017.* Connect the Saleae logic analyzer so you can monitor the data transfer between the AVR128DB48 and MCP23017. **Caution: in your design, the only supply voltage to be used is 3.3V from the Curiosity board.**

Create a project named `MCP23017_write_test` using the program `MCP23017_write_test` that you wrote for Task 1. Set the compiler optimization to -Og. Build the program. Place any variables in a watch window to observe their values. Single step through each instruction in the program. Prior to single-stepping each instruction, determine what changes you expect in the values of the port registers and any variables you created.

Run your program and verify that it continually increments the binary value displayed on the LEDs. Use the Saleae logic analyzer to capture the first transaction sent from the microcontroller to the GPIO expander. Submit this screen capture with your laboratory.

When your program is working correctly, have a TA verify that your program performs as required. Get the TA's signature.

Laboratory Task 2: A GPIO Implemented Simple 8-bit Parallel Input Port and Simple 8-bit Parallel Output Port

Create a project named `MCP23017_inout_test` using the program `MCP23017_inout_test` that you wrote for Task 2. Set the compiler optimization to -Og. Build the program. Place any variables in a watch window to observe their values. Single step through each instruction in the program. Prior to single-stepping each instruction, determine what changes you expect in the values of the port registers and any variables you created.

When your program is working correctly, have a TA verify that your program performs as required. Get the TA's signature.

Laboratory Task 3: A GPIO Implemented Simple 8-bit Parallel Input Port and Simple 8-bit Parallel Output Port Using MCP23017 Interrupts

Add the wire from the MCP23017 INTA pin to PF3 of the AVR128DB48.

Create a project named `MCP23017_inout_interrupt_test` using the program `MCP23017_inout_interrupt_test` that you wrote for Task 3. Set the compiler optimization to -Og. Build the program. Place any variables in a watch window to observe their values. Single step through each instruction in the program. Prior to single-stepping each instruction, determine what changes you expect in the values of the port registers and any variables you created.

When your program is working correctly, have a TA verify that your program performs as required. Get the TA's signature.

Leave the circuit you have constructed on your breadboard insert, it may be used in later laboratories.

Questions

1. Fill out the logic level compatibility check list (in the laboratory folder) for the AVR128DB48 and the MCP23017. Make the AVR128DB48 device A. Both devices are operated with a supply voltage of 3.3V.