Lab 3

Parallel Port

EEL 4746C / EEL 5756C: Microcomputers

Fall 2018

Student Name:	
Student ID:	
Section:	

Contents

1 Objective

The objective of this lab is to be able to interface with a simple input and output devices using the parallel port.

2 Introduction

The parallel port is one technique to interface peripheral devices to the microporcessor or the microcontroller. The parallel port allow multiple bits to be transferred at the same time. The general purpose I/O (GPIO) pins are digital pins that can be used for inputs and outputs. In this experiment, we will use the GPIO pins for parallel input and parallel output.

2.1 GPIO ports of Arduino UNO (Rev. 3)

The ATMega328P has three I/O ports: B, C, and D. The GPIO ports are aka digital ports. Each port is an 8-bit port. However, only the first 7-bits of port C are mapped to the pins of the microcontroller. So, a total of 23-bits can be used for the GPIO. Note, that this number will be decreased if any of the other features of the microcontroller are used, like the ADC pins. This reeducation is due to the multiplexing of functionalities on each pin. In our case, the two most pins PB6 and PB7 are used to connect the 16MHz crystal to the ATmega328p.
Each port has three registers: 1) Data register; 2) Data direction register; and 3) Input register. Data register is used to hold the output. Data direction is used for input or for output: a) 1 in a bit is used to set the corresponding pin as an output pin; and b) 0 in a bit is used to set the corresponding pin as an input pin. Input register is used to read the data. When writing 1 to a bit in the input register, the corresponding bit is toggled in the data register.

Port	Register Name	Register Reference	Memory Address	I/O address	Pins
B	Data Register	PINB	0x0023	0x03	8
	Data Direction Register	DDRB	0x0024	0x04	
	Input Register	PORTB	0x0025	0x05	
C	Data Register	PINC	0x0026	0x06	7
	Data Direction Register	DDRC	0x0027	0x07	
	Input Register	PORTC	0x0028	0x08	
D	Data Register	PIND	0x0029	0x09	8
	Data Direction Register	DDRD	0x002A	0x0A	
	Input Register	PORTD	0x002B	0x0B	

3 Procedure

3.1 Part 1 (Simple I/O using I/O instruction and I/O address space)

Step 1. Create four simple switches as shown in section 4.
Step 2. Check the switches using a digital multimeter.
Step 3. Create four simple output circuits as shown in section 5.
Step 4. Check the output circuits using V_{cc} and GND from your Arduino UNO board.
Step 5. Now, connect the output of the four switches, in order from right to left, to pins PB_0, PB_1, PB_2, and PB_3 respectively.
Step 6. Connect the input of the four output circuits, in order from right to left, to pins PC_0, PC_1, PC_2, and PC_3 respectively.
Step 7. Write the program in section 6.1.
Step 8. Assemble and Link the program.
Step 9. Follow the steps in section 7 to convert and write your program to the Flash Memory.
Step 10. Test your circuit.
Step 11. What does the circuit do?

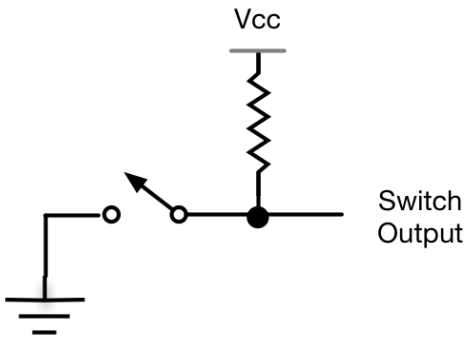
3.2 Part 2 (Simple I/O using Load Instruction and Data Memory address space)

Step 1. Create four simple switches as shown in section 4.
Step 2. Check the switches using a digital multimeter.
Step 3. Create four simple output circuits as shown in section 5.
Step 4. Check the output circuits using V_{cc} and GND from your Arduino UNO board.
Step 5. Now, connect the output of the four switches, in order from right to left, to pins PB_0, PB_1, PB_2, and PB_3 respectively.
Step 6. Connect the input of the four output circuits, in order from right to left, to pins PC_0, PC_1, PC_2, and PC_3 respectively.
Step 7. Write the program in section 6.2.
Step 8. Assemble and Link the program.
Step 9. Follow the steps in section 7 to convert and write your program to the Flash Memory.
Step 10. Test your circuit.
Step 11. What does the circuit do?

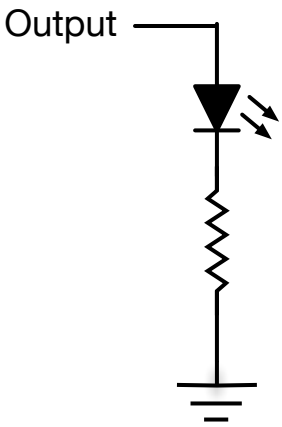
3.3 Part 3 (Write your own code)

Step 1. Create four simple switches as shown in section 4.
Step 2. Check the switches using a digital multimeter.
Step 3. Create four simple output circuits as shown in section 5.
Step 4. Check the output circuits using V_{cc} and GND from your Arduino UNO board.
Step 5. Now, connect the output of the four switches, in order from right to left, to pins PB_0, PB_1, PB_2, and PB_3 respectively.
Step 6. Connect the input of the four output circuits, in order from right to left, to pins PC_0, PC_1, PC_2, and PC_3 respectively.
Step 7. Write a program that takes two 3-bit numbers add them and display the result on the output circuit.
Step 8. Assemble and Link the program.
Step 9. Follow the steps in section 7 to convert and write your program to the Flash Memory.
Step 10. Test your circuit.

4 Simple Input Circuit (1-bit)



5 Simple Output Circuit (1-bit)



6 Appendix

6.1 Code 1

```
.global start
.text
# PORT D will be used for input
.set  PIND, 0x09
.set  DDRD, 0x0A
.set  PORTD, 0x0B

# PORT B will be used for output
.set  PINB, 0x03
.set  DDRB, 0x04
.set  PORTB, 0x05


.org 0x0000
reset_vector:
              jmp start     ; skip interrupt vector table

.org 0x0100
start:
              ; Set the first four bits of port B as
                  outputs
              ldi r16, 0x0F
              out DDRB, r16

              ; Set the first four bits of port D as
                  inputs
              ldi r17, 0x00
              out DDRD, r17

repeat:

              ; Read the input from port D
              in  r18, PIND

              ; Clean the inputs
              and  r18, r16

              ; Output Value
              out PORTB, r18

              rjmp repeat

.end
```

6.2 Code 2

```
.global start
.text
# PORT D will be used for input
.set  MPIND, 0x0029
.set  MDDRD, 0x002A
.set  MPORTD, 0x002B

# PORT B will be used for output
.set  MPINB, 0x0023
.set  MDDRB, 0x0024
.set  MPORTB, 0x0025


.org 0x0000
reset_vector:
              jmp start     ; skip interrupt vector table

.org 0x0100
start:
              ; Set the first four bits of port B as
                  outputs
              ldi r16, 0x0F
              sts MDDRB, r16

              ; Set the first four bits of port D as
                  inputs
              ldi r17, 0x00
              sts MDDRD, r17

repeat:

              ; Read the input from port D
              lds  r18, MPIND

              ; Clean the inputs
              and  r18, r16
              com  r18

              ; Output Value
              sts MPORTB, r18

              rjmp repeat

.end
```

7 Convert and Write to Flash

```
>> avr-objcopy -O ihex -R .eeprom -R .fuse -R .lock -R .signature <filename>.x <filename>.hex

>> sudo avrdude -F -V -c arduino -p ATMEGA328P -P /dev/ttyACM0 -b 115200 -U flash:w:<filename>.hex
```