EE443 - Embedded Systems

Experiment 2

Input Port Pins and Execution Timing

Purpose: Read MCU port inputs and control an output port according to the input values. Precisely adjust timing of output signals generated by the MCU code.

Preliminary Work

1. Port-D of ATmega328p will be used to read input signals in this experiment. Read the information on "***I/O ports***" in the ATmega328p datasheet. The section "***14.2 Ports as General Digital I/O***" of the datasheet describes how the port pins are accessed by using **DDRx** (Data Direction Register of Port-x), **PORTx** (Data Register of Port-x for output), and **PINx** (Data Input Register of Port-x) SFRs.

2. Starting with the program used in Experiment-1, add the necessary statements to initialize the Port-D pin-0 and pin-1 for input. Leave all other Port-D pins as output pins. Modify the program to perform the following operations according to the switch positions connected to Port-D:

- LED0, LED1, LED2, and LED3 are ON while **SW0** is pressed.
- LED4, LED5, LED6, and LED7 are ON while **SW1** is pressed.

You can **AND** the register contents with a **mask** to check the individual bit values. For example, the following if condition checks the value of pin-3 of port-D:

```
if ( ( PIND & _BV(PIND3) ) == 0x00 )
```

If the pin-3 of port-D is zero, then this condition will be true. **`_BV(.)`** is a macro function provided by WinAVR compiler that returns a constant with only one bit set according to the macro parameter. In the above example, **`_BV(PIND3)`** produces the mask value, **`0x08`**, or **`00001000`** in binary. This macro function can be used to generate masks for all bit labels specified by the register descriptions in the datasheet.

As an alternative to masking register bits, WinAVR compiler also provides two macro functions that correspond to the machine instructions for testing individual bit values in the SFRs:

```
bit_is_set(SFR, bit)
```

tests whether the specified **`bit`** (0, 1, 2, ...7) in the IO register, **`SFR`**, is **set**. This will return a **`0`** if the bit is cleared, and a non-zero value if the bit is set.

```
bit_is_clear(SFR, bit)
```

tests whether the specified **`bit`** (0, 1, 2, ...7) in the IO register, **`SFR`**, is **cleared**. This will return a **`0`** if the bit is set, and a non-zero value if the bit is cleared.

These macros generate a machine code that makes use of the following conditional branch instructions of ATmega328 CPU:

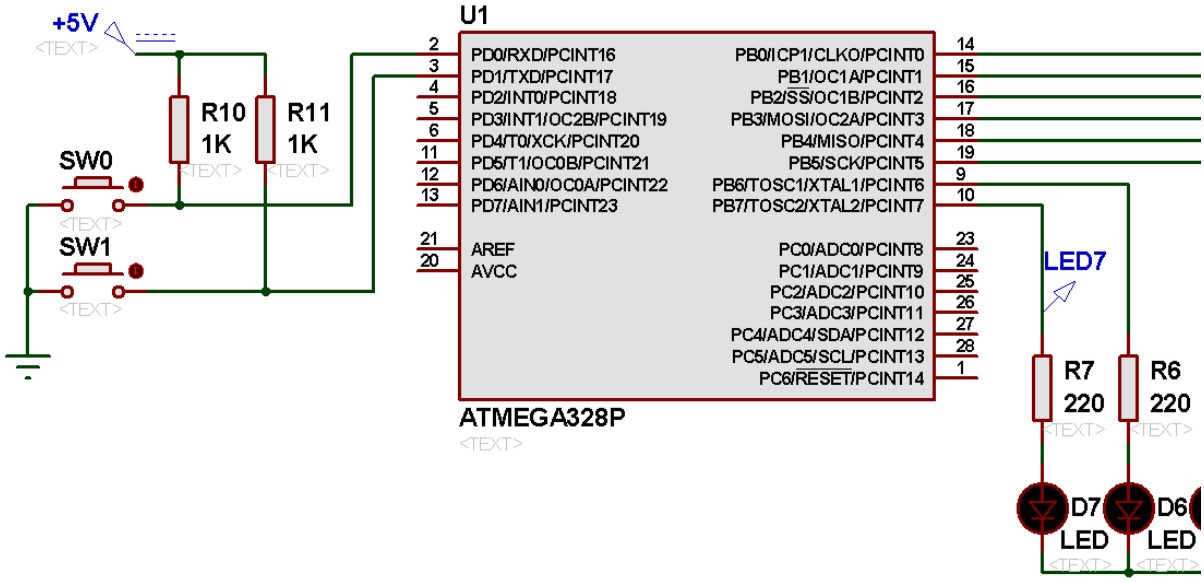
```
SBIC R, b ;  Skip if Bit b in I/O Register R is Cleared
SBIS R, b ;  Skip if Bit b in I/O Register R is Set
```

Procedure

1. Create a new project directory for Experiment 2 on your hard disk following the suggestions given in Experiment 1.

2. Open the design file for Experiment 1, **Exp1_LED.dsn**, in **ISIS**, and save it with a different name in the new project directory using the "**File>Save Design As...**" menu option.

3. Modify the schematic you saved for Experiment 2 adding two push-button switches and **1 KΩ** pull-up resistors as shown in the following figure. The push-button switch is listed as "**BUTTON**" in the "**Switches & Relays**" category of the component libraries.

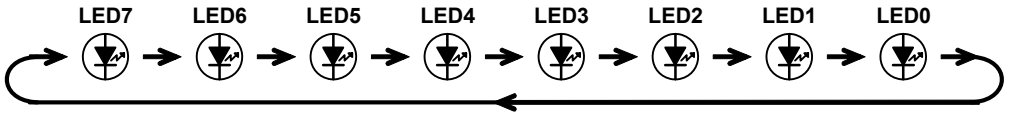


4. Start **Code::blocks**, and create a new AVR project in the Experiment 2 directory following the guidelines given for Experiment 1. Compile the program you prepared in the preliminary work.

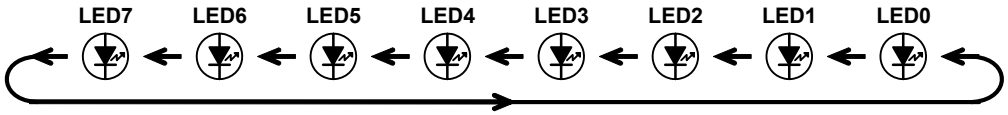
5. Open the **Edit Properties** window for **ATMEGA328P** microcontroller on the ISIS schematic and select the **.hex** file generated by the compiler for Experiment-2. Test your program looking at the animated simulation results, and debug your program, if necessary.

6. Modify the program again as described below.

- One, and only one LED will be ON at a time.
- If **SW0** is not pressed then the lit LED position does not change. LED light rotates continuously while **SW0** is pressed. The rotation direction is determined by **SW1** as follows.
- If **SW1** is pressed then LED light rotates **right**:



- If **SW1** is not pressed then LED light rotates **left**:



Test your program looking at the animated simulation results, and debug if necessary. You should change the simulation CPU clock and the animation timestep to be able to follow the LED lights on the screen.

- Change **CKSEL Fuses** setting of ATMEGA328P to **(0011) Int. RC Ocs. 128KHz**.
- Select the **System>Set Animation Options ...** menu item and change the **Timestep per Frame** setting to **100u**.

7. Open a digital graph window and **set the stop time to 20 µs.**

Place voltage probes on all Port-B pins. Add digital graph traces for all of the eight Port-B probing nodes.

Close either **SW0** or **SW1** permanently (click on the red circle near the switch).

Restore the **CKSEL Fuses** setting of ATMEGA328P to **(0010) Int. RC Ocs. 8MHz**.

Simulate the graph and observe the ON-time of all LEDs.

8. Modify the program so that the ON-time of all LEDs will be the same while rotating left and rotating right. Make sure that the LED0 and LED7 have the same ON-time. You should compare the LED timing with one CPU clock cycle (125 ns) accuracy.

You can insert dummy instructions into your code where you need a longer execution time. For example, repeat **PORTB** write operation to increase the LED cycle time by one clock cycle.

Question 1: How can you remove the two pull-up resistors connected to the switches without changing the circuit functionality? Find a solution by using the programmable port features described in the ATmega328 datasheet.