Read-Modify-Write (RMW) Problem with Mid-Range PIC Microcontrollers Programmed in PicBasic Pro and Avoiding the Read-Modify-Write (RMW) Problem with LATx Registers in PIC18Fxxxx
Cornerstone Electronics Technology and Robotics II

Introduction:

This discussion focuses on the **Read-Modify-Write** (RMW) problem in mid-range PIC microcontrollers programmed in PicBasic Pro and avoiding the RMW problem with PIC18Fxxxx. For those using assembly language, see the references below especially: http://ww1.microchip.com/downloads/en/devicedoc/31009a.pdf.

On a mid-range PICmicro, every I/O port has two registers directly associated with the operation of the port:

- TRISx: Data Direction Control register
- PORTx: I/O Port register

The TRISx register controls the direction of each pin in PORTx. For example, TRISB is the direction control for PORTB. If the TRIS bit for an I/O pin is ‘1’, then the pin is an input. If the TRIS bit for an I/O pin is ‘0’, then the pin is an output. Data (1 or 0) on the I/O pins is accessed via a PORTx register. A particular pin can be set HIGH or LOW (1 or 0) using the PORTx register. For instance, to set RB0 – RB2 of the PORTB register to HIGH and the remainder of pins RB3 – RB7 to LOW, you would write: PORTB = %00000111.

Read-Modify-Write (RMW) Problem with Mid-Range PIC Microcontrollers:

The Microchip mid-range PIC microcontrollers use a sequence known as **Read-Modify-Write** (RMW) when changing an output state (1 or 0) on a pin. This can cause unexpected behavior under certain circumstances. When your program changes the state on a specific pin, for example RB3 in PORTB, the PIC microcontroller first **READ**s all 8 bits of the PORTB register which represents the states of all 8 pins in PORTB (RB7-RB0). The PIC microcontroller then stores this data in the CPU. The bit associated with RB3 that you've commanded to **MODIFY** is changed, and then the PIC microcontroller **WRITE**s all 8 bits (RB7-RB0) back to the PORTB register. When you first read the PORT register, you will be reading the actual state of the physical pin (1 for HIGH or 0 for LOW), just as if you were to hook a logic probe to the pin and read a HIGH or LOW.

The problem arises when an output pin is loaded in such a way that its logic state is affected by the load. Instances of such loads are LEDs without current-limiting resistors or loads with high capacitance or inductance. For example, in the circuit in Figure 1 below three LEDs are connected to PIC16F88 pins RB0 – RB2 with a small bypass capacitor in parallel with the LEDs. Following the schematic is the complementary PicBasic Pro program for a PIC16F88 which is suppose to turn on all of the LEDs connected to pins RB0 – RB2.

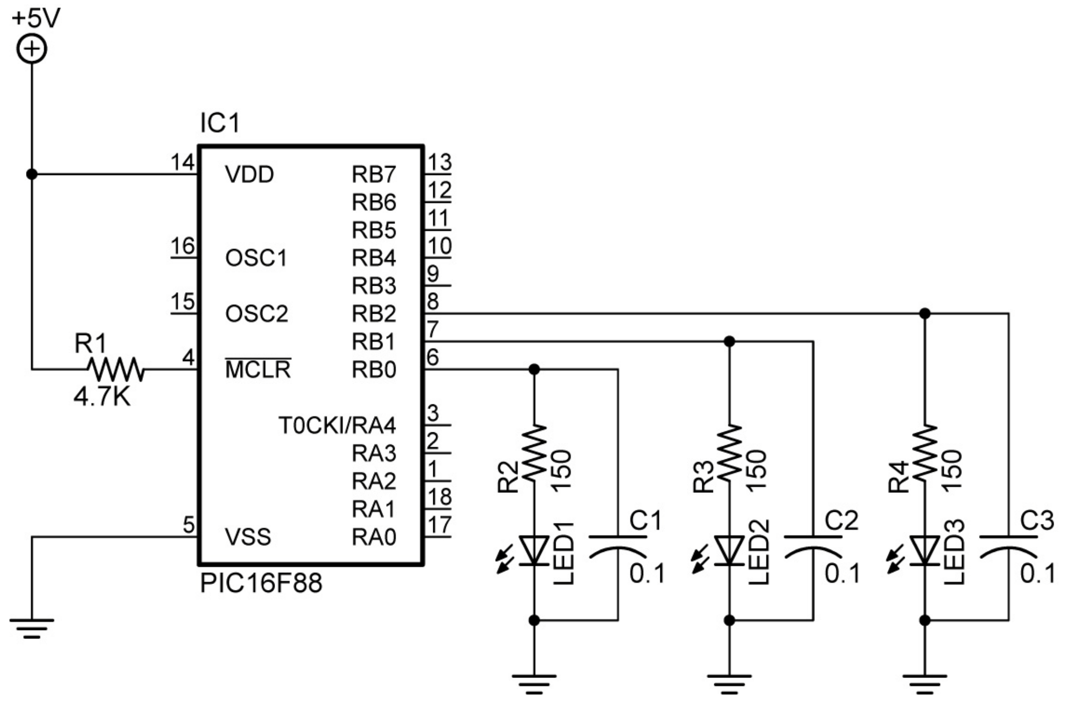

Figure 1: PIC16F88 Schematic

PicBasic Pro Program for 16F88:

```
TRISB = %11111000     ' Set RB0-RB2 as outputs (0) and RB3-RB7 as inputs (1)
PORTB = %00000000     ' Set all PORTB pins to LOW (0V)
ANSEL = 0             ' Sets analog/digital bits to digital.
OSCCON = $60          ' Sets the internal oscillator in the 16F88 OSCCON
                      'register to 4 MHz
'  Main Code
PORTB.0 = 1           ' Change state of pin RB0 from LOW (0) to HIGH (1),
                      ' turn on LED on RB0
PORTB.1 = 1           ' Change state of pin RB1 from LOW (0) to HIGH (1),
                      ' turn on LED on RB1
PORTB.2 = 1           ' Change state of pin RB2 from LOW (0) to HIGH (1),
                      ' turn on LED on RB2
END
```

When the program runs, only the LED connected to RB2 lights. The problem arises due to the time it takes to bring each pin up to a stable +5V. Because of the bypass capacitor on each pin, the voltage does not rise immediately when you set a pin to HIGH. There is a delay while the bypass capacitor on the pin charges. When driving a capacitive load from 0V start condition to +5V, the port pin driver is presented with a short circuit which takes time to overcome (the capacitor charge time).

Here is the process in more detail:

- RB0 is set to HIGH. (This step is abbreviated to get directly to the RMW problem.)
- RB1 is set to HIGH in the following manner:
    - The PIC **READ**s the present states of all the PORTB pins (RB7-RB0) just as if you were to hook a logic probe to each pin and read HIGH or LOW. RB0 <u>should be read HIGH</u> since it was set to HIGH (%0000000**1**), but it <u>actually reads LOW</u> (%0000000**0**) because the capacitor hasn't allowed the pin the time to come up to +5V (HIGH).
    - The PIC microcontroller then stores this data (%00000000) in the CPU.
    - The value of RB1 is **MODIFIED** to HIGH (%000000**1**0).
    - The PIC now **WRITE**s this modified data from the CPU back to the PORTB register. The value for RB0 in the PORTB register (%0000001**0**) has turned off the LED connected to RB0.
- RB2 is set to HIGH in the same manner:
    - The PIC **READ**s the present states of all the PORTB pins (RB7-RB0). RB1 <u>should be read HIGH</u> since it was set to HIGH (%000000**1**0), but it <u>actually reads LOW</u> (%000000**0**0) because the capacitor hasn't allowed the pin the time to come up to HIGH.
    - The PIC microcontroller then stores this data (%00000000) in the CPU.
    - The value of RB2 is **MODIFIED** to HIGH (%00000**1**00).
    - The PIC now **WRITE**s this modified data from the CPU back to the PORTB register. The value for RB1 in the PORTB register (%000001**0**0) has turned off the LED connected to RB1.
    - Because RB2 is the last pin on the port to be set, it stays high long enough for the voltage to stabilize. The LED connected to RB2 will be the only one of the three that lights.

To correct the problem with code, insert a PAUSE after each line PORTB.x = 1 or modify the entire PORTB register by a single line PORTB = %00000111.

For other references in PicBasic Pro, see:

- http://www.melabs.com/faq/02041702.htm

For references in assembly, see:

- http://ww1.microchip.com/downloads/en/devicedoc/31009a.pdf (Section 9. I/O Ports of the PICmicro Mid-Range MCU Family Reference Manual)
- http://ww1.microchip.com/downloads/en/devicedoc/33023a.pdf (The entire PICmicro Mid-Range MCU Family Reference Manual)
- http://www.mikroe.com/forum/viewtopic.php?t=19832

LATx in PIC18Fxxxx Series Microcontroller:

On a PIC18Fxxxx device, every I/O port has three registers directly associated with the operation of the port:

- TRISx: Data Direction Control register;
- PORTx: I/O Port register;
- LATx: Data Latch register;

The function of TRISx and PORTx are the same as described in the introduction.

The LATx register in the PIC18Fxxxx series provides built-in support to avoid the read-modify-write problem. In the 18Fxxxx devices, the LATx register holds the value written to the PORT irrespective of the voltage level on the port pins.

Now when your program changes the state on a specific pin, for example RB3 in PORTB, the PIC microcontroller first **READ**s all 8 bits of the LATB register not the PORTB register. The PICmicro reads the values that were written to each pin (e.g. PORTB.3 = 1) not the actual state of the physical pin (HIGH or LOW). The PIC microcontroller then stores this data in the CPU. The bit associated with RB3 that you've commanded to **MODIFY** is changed, and then the PIC microcontroller **WRITE**s all 8 bits to the LATB register. A write to the LATB register has the same effect as a write to the PORTB register.

For example, in the circuit in Figure 2 below three LEDs are connected to PIC18F4331pins RB0 – RB2 with a small bypass capacitor in parallel with the LEDs. Following the schematic is the complementary PicBasic Pro program for a PIC18F4331 which turns on all of the LEDs connected to pins RB0 – RB2.

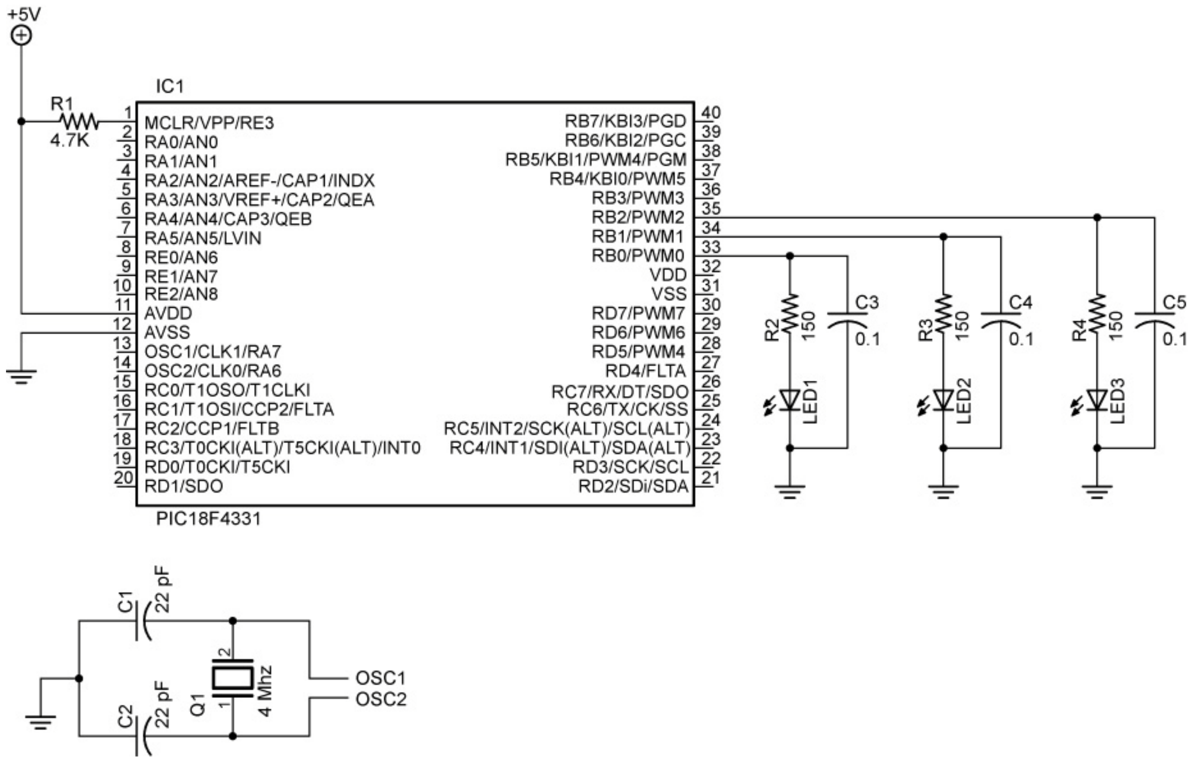

Figure 2: PIC 18F4331 Schematic

PicBasic Pro Program for 18F4331:

```
TRISB = %11111000      ' Set RB0-RB2 as outputs(0) and RB3-RB7 as inputs(1)
PORTB = %00000000   ' Set all PORTB pins to LOW (0V)

'  Main Code
PORTB.0 = 1               ' Change state of pin RB0 from LOW (0) to HIGH (1),
                          ' turn on LED on RB0
PORTB.1 = 1               ' Change state of pin RB1 from LOW (0) to HIGH (1),
                          ' turn on LED on RB1
PORTB.2 = 1               ' Change state of pin RB2 from LOW (0) to HIGH (1),
                          ' turn on LED on RB2
END
```

When you run the program, all of the LEDs light up even though there is still a delay while the bypass capacitor on the pin charges.

Here is the process in more detail:

- RB0 is set to HIGH.
- RB1 is set to HIGH in the following manner:
  - The PIC **READ**s all 8 bits of written values stored in LATB register. The bit tied to RB0 reads HIGH since it was set to HIGH (%0000000**1**) and the Data Latch register LATB is not affected by the actual state of PORTB pins. A read of the LATB register returns the values held in the port output latches instead of the values on the I/O pins. This avoids the read-modify-write problem found in the mid-range PIC microcontrollers.
  - The PIC microcontroller then stores this data (%00000001) in the CPU.
  - The value of RB1 is **MODIFIED** to HIGH (%00000011).
  - The PIC now **WRITE**s this modified data from the CPU back to the LATB register. A write to the LATB register has the same effect as a write to the PORTB register. The value for RB0 in the PORTB register (%0000001**1**) has left on the LED connected to RB0.
- RB2 is set to HIGH in the same manner:
  - The PIC **READ**s all 8 bits of written values stored in LATB register. The bit tied to RB1 reads HIGH since it was set to HIGH (%000000**1**1) and the Data Latch register LATB is not affected by the actual state of PORTB pins.
  - The PIC microcontroller then stores this data (%00000011) in the CPU.
  - The value of RB2 is **MODIFIED** to HIGH (%00000111).
  - The PIC now **WRITE**s this modified data from the CPU back to the LATB register. The value for RB1 in the PORTB register (%000001**1**1) has left on the LED connected to RB1.

The differences between the PORTx and LATx registers can be summarized as follows:

- A write to the PORTx register writes the data value to the port latch.
- A write to the LATx register writes the data value to the port latch.
- A read of the PORTx register reads the data value on the I/O pin.
- A read of the LATx register reads the data value held in the port latch.

For references in assembly, see:

- http://ww1.microchip.com/downloads/en/DeviceDoc/39500a.pdf (Section 11. I/O Ports)