UNCA CSCI 255
Exam 3 Fall 2011

This is a closed book and closed notes exam. Laptops, cell phones, and any other electronic storage or communication devices may not be used during this exam.

Name:_____KEY_____

If you want partial credit for imperfect answers, explain the reason for your answer!

True/False Questions (20 points)

1. **(true or false)** The ***c****onfiguration bits* are stored in program memory because they control various processor options used at system startup.

2. **(true or false)** The Port B register bits are _RB0 through _RB27.

3. **(true or false)** Assuming that pin RB1 has been set to digital output, making the assignment _RB1=1 is the preferred way to set that pin to high.

4. **(true or false)** The *Interrupt Vector Table (IVT)* is stored in data memory.

5. **(true or false)** A pin configured as an *open drain output* is internally connected to Vdd.

6. **(true or false)** The PIC24 can be configured to use either an external crystal or an internal oscillator as its clock.

7. **(true or false)** Bring the MCLR# pin to low causes a system reset.

8. **(true or false)** The *Interrupt Vector Table (IVT)* stores the starting address of the interrupt Service Routine ISR for each interrupt source.

9. **(true or false)** Every interrupt (excluding traps) has a priority between 0 and 7.

10. **(true or false)** The priority of any trap is greater than the priority of any interrupt.

Short Answer Questions (80 points)

1. Use the pre-defined macros and inline functions (you may not need all of them) provided below to write the inline function: CONFIG_RB8_AS_DIG_OD_OUTPUT()

   ENABLE_RB8_PULLUP(), DISABLE_RB8_PULLUP(),
   CONFIG_RB8_AS_DIG_INPUT(),
   ENABLE_RB8_OPENDRAIN(), DISABLE_RB8_OPENDRAIN(),
   _TRISB8, _RB8, _LATB8, _RP8

   ANS:
   ```
   CONFIG_RB8_AS_DIG_OD_OUTPUT() {
       _TRISB8 = 0;
       ENABLE_RB8_OPENDRAIN();
   }
   ```

2. Briefly describe 5 of the functionalities associated with the physical pins of the PIC24, as shown below.

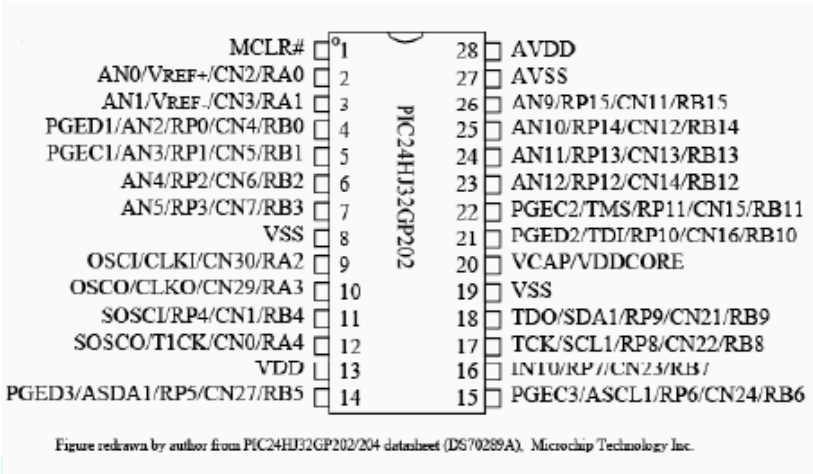


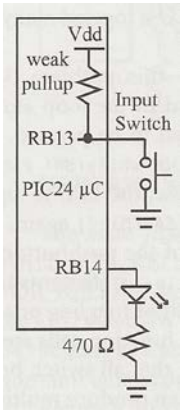
Figure redrawn by author from PIC24HJ32GP202/204 datasheet (DS70289A), Microchip Technology Inc.

ANS: 1. VDD, VSS: power and ground pins, respectively.
2. MCLR#: This pin resets the processor when brought low.
3. CNn: Change notification pins.
4. RAn, RBn: Parallel port I/O pins.
5. RPn: Remappable peripheral pins that allow internal functions to be connected to physical pins.

Questions 3 through 5 below refer to the following program running on a PIC24 processor configured as shown in the adjoining figure.

```
void _ISR _CNInterrupt(void) {
** _CNIF = 0;
   // replace this comment
}

int main (void) {
   CONFIG_RB13_AS_DIG_INPUT();
   ENABLE_RB13_CN_INTERRUPT();
** _CNIF = 0;
** _CNIP = 2;
** _CNIE = 1;
   while (1);
}
```



3. Explain the functionality of each stared line in the program above. Be detailed and specific.

ANS: `_CNIF = 0;` (inside the ISR): Resets the change notification interrupt flag to 0. Its value at the time the IRS is called is 1.
`_CNIF = 0;` (in main): Initializes the change notification interrupt flag to 0---no interrupt has occurred.
`_CNIP = 2;` Sets the change notification interrupt priority to 2.
`_CNIE = 1;` Enables the change notification interrupt.

4. What purpose is served by the weak pull-up resistor on pin RB13? Why is it necessary to have this configuration when running the program provided above.

   ANS: The pull-up resistor ties RB13 to Vdd. Without the pull-up resistor, RB13 would be *floating*, like a disconnected wire, when the pushbutton switch is not active (i.e., not pushed). The voltage on RB13 can vary when it is floating. **Tying RB13 to Vdd with a pull-up resistor assures that the voltage on the pin is high when the pushbutton switch is not active.** Pushing the pushbutton switch drops the voltage to low.

5. Assume that the pushbutton switch is pressed and released one time and there is no switch bounce. How many times will the interrupt service routine be called if the comment line, `// replace this comment,` is replaced by the line of code shown below. Consider each replacement separately and be sure to explain your answers.

   a. _CNIE = 0;
   ANS: Setting CNIE to 0 disables the change notification interrupt. Once the interrupt is disabled, the IRS is not called even if the interrupt flag is raised. **When the switch is pressed, the IRS is called and it disables the interrupt. Even though the interrupt flag is raised again when the switch is released, the IRS is called only once.**

   b. _CNIP = 0;
   ANS: Setting CNIP to 0 sets the interrupt priority to the lowest possible level which is also the priority level of the code in main(). Because an interrupt can only "interrupt" code with lower priority, an interrupt with 0 priority can never run. **When the switch is pressed, the IRS is called and it sets the interrupt priority to the lowest possible level. Even though the interrupt flag is raised again when the switch is released, the IRS is called only once.**

Questions 6 through 9 below refer to the following program running on a PIC24 processor configured as shown in the adjoining figure.

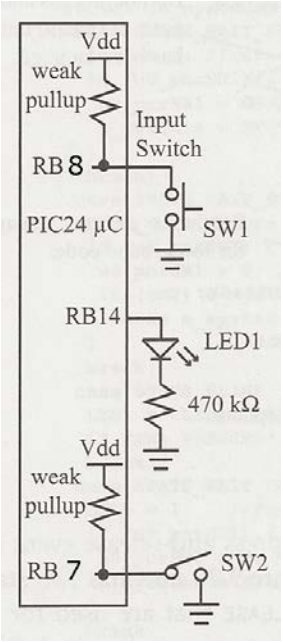
```
#include "pic24_all.h"

#define CONFIG_LED() CONFIG_RB14_AS_DIG_OUTPUT()
#define LED  _LATB14
#define SW1          _RB8
#define SW2          _RB7

inline void CONFIG_SW1()  {
   CONFIG_RB8_AS_DIG_INPUT();
   ENABLE_RB8_PULLUP();
}

inline void CONFIG_SW2()  {
   CONFIG_RB7_AS_DIG_INPUT();
   ENABLE_RB7_PULLUP();
}

** volatile uint8 SW1value = 0;
   volatile uint8 oldSW1value = 0;
   volatile uint8 SW2value = 0;
   volatile uint8 oldSW2value = 0;
```



```
   void _ISRFAST _T3Interrupt (void) {
      _T3IF = 0;
**    SW1value = SW1;
      SW2value = SW2;
**    if(SW1value == 1 && oldSW1value == 0) {
**       LED = 1;
      }
      if(SW2value == 1 && oldSW2value == 0) {
         LED = 0;
      }
      oldSW1value = SW1value;
      oldSW2value = SW2value;
   }

#define ISR_PERIOD     15     // in ms
void  configTimer3(void) {
  T2CONbits.T32 = 0;     // 32-bit mode off
  T3CON = T3_OFF |T3_IDLE_CON | T3_GATE_OFF
          | T3_SOURCE_INT
          | T3_PS_1_64 ;  //results in T3CON= 0x0020
  PR3 = msToU16Ticks (ISR_PERIOD, getTimerPrescale(T3CONbits)) - 1;
  TMR3  = 0;
  _T3IF = 0;
  _T3IP = 1;
  _T3IE = 1;
  T3CONbits.TON = 1;
}

int main (void) {
   configBasic(HELLO_MSG);
   CONFIG_SW1();
   CONFIG_SW2();
   CONFIG_LED();
   configTimer3();
   while (1) {
     doHeartbeat();
   }
}
```

6. In the program above, identify the following:
   a. *The Interrupt Service Routine (ISR)*
   b. The function used to configure the timer
   c. The function used to configure the pin attached to SW1
   d. The function used to configure the pin attached to the LED

7. Explain the functionality of each stared line in the program above. Be detailed and specific.

ANS: `1. volatile uint8 SW1value = 0;` Declares and initializes an 8-bit unsigned integer variable that is *volatile*. Variables that are used in ISRs should be volatile. Volatile variables are stored in program memory and not automatically initialized therefore they hold their values through system resets. `2. SW1value = SW1;` Sets `SW1value` to the current value of RB8, in effect taking an instantaneous sample of RB8 for use throughout the remainder of the IRS.

`3. if(SW1value == 1 && oldSW1value == 0)` Tests for a switch movement from active or pressed, `(oldSW1value == 0),` to non-active or released, `(SW1value == 1).` This test will be true if the switch has just been pressed and then released.
`4. LED = 1;` Sets _LATB14 to high turning on the LED driven by pin RB14.

8. Analyze the program:
   a. Assume that the pushbutton switch, SW1, is pressed and released one time and there is no switch bounce, what will happen as a result of this action? Explain your answer.

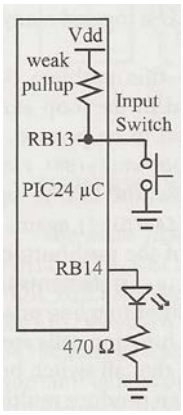
      ANS: Each time SW1 is pressed and then released, the LED on pin RB14 is turned on. If the LED is already on when SW1 is cycled, there will be no visible change in the state of the LED.

   b. Assume that the switch SW2 is pressed and released one time and there is no switch bounce, what will happen as a result of this action? Explain your answer.

      ANS: Each time SW2 is pressed and then released, the LED on pin RB14 is turned off. If the LED is already off when SW2 is cycled, there will be no visible change in the state of the LED.

Questions 9 and 10 below refer to the following two programs, each running separately on a PIC24 processor configured as shown in the adjoining figure.

Program 1:




```
#include "pic24_all.h"
#define CONFIG_LED1() CONFIG_RB14_AS_DIG_OUTPUT()
#define LED1  _LATB14

void _ISRFAST _T3Interrupt (void) {
    LED1 = !LED1;
    _T3IF = 0;
}

#define ISR_PERIOD     300  //the timer period in ms
void  configTimer3(void) {
    T2CONbits.T32 = 0;
    T3CON = T3_OFF | T3_IDLE_CON | T3_GATE_OFF
            | T3_SOURCE_INT
            | T3_PS_1_256 ;
    PR3 = msToU16Ticks (ISR_PERIOD, getTimerPrescale(T3CONbits)) - 1;
    TMR3  = 0;
    _T3IF = 0;
    _T3IP = 1;
    _T3IE = 1;
    T3CONbits.TON = 1;
}

int main (void) {
    configClock();
    CONFIG_LED1();
    LED1 = 1;
    configTimer3();
```

```
    while (1) {
       IDLE();  // the processor enters a low power mode
    }
}
```

Program 2:

```
#define CONFIG_LED1()  CONFIG_RB14_AS_DIG_OUTPUT()
#define LED1  _LATB14

int main(void) {
   configClock();
   CONFIG_LED1();
   LED1 = 1;
   while (1) {
      DELAY_MS(300);
      LED1 = !LED1;
  }
}
```

9. Explain what happens when each program runs. Could an observer determine which program was running if they did not already know, and, if so, how?

   ANS: Both programs reverse the state of the LED on pin RB14 every 300 milliseconds. In both programs, the LED is initially on. It is not possible to distinguish which program is running without looking at the code.

10. Which program uses the processor more efficiently and why?

    ANS: The first program is more efficient because it is in idle mode (i.e., low power mode) except when it is servicing the timer 3 interrupt. The timer 3 IRS is responsible for cycling the LED. The second program is always running at full power.

Extra Credit Problem:

Explain the functionality of each stared line in the program below. Be detailed and specific.

```
#define ISR_PERIOD     300  // in ms
void  configTimer3(void) {
    T2CONbits.T32 = 0;
    T3CON = T3_OFF | T3_IDLE_CON | T3_GATE_OFF
            | T3_SOURCE_INT
            | T3_PS_1_256 ;
**  PR3 = msToU16Ticks (ISR_PERIOD, getTimerPrescale(T3CONbits)) - 1;
```

ANS: Sets the PR3 register to the number of ticks corresponding to `ISR_PERIOD`. This set the timer to the desired time interval.

`**  TMR3  = 0;` **ANS**: Clears the tick counter for timer 3.

`**  _T3IF = 0;` **ANS**: Sets the timer 3 interrupt flag to 0 or off.

`**  _T3IP = 1;` **ANS**: Sets the timer 3 interrupt priority to 1.

`**  _T3IE = 1;` **ANS**: Enables the timer 3 interrupt.

`**  T3CONbits.TON = 1;` **ANS**: Turns on timer 3.

```
}
```