EXAM #3

- Per MSU Academic Operating Policy 10.08, you **MUST** turn your cell phone off – not on vibrate, but completely off. If you have a pager, you must also turn it off.
- You may only use the provided reference materials.
- Unless stated otherwise, all multi-byte data values are stored in *little-endian* ordering.
- Absolutely NO cheating is allowed. If you are caught in the attempt of, the act of, or the past action of academic dishonesty, you will receive the maximum punishment allowed by University policy.

Pinout (shaded pins are 5V tolerant):

Reference for problem #1:

<table>
<thead>
<tr>
<th>Flag Bit</th>
<th>Set by:</th>
<th>Cleared by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAPR (RCON&lt;15&gt;)</td>
<td>Trap conflict event</td>
<td>POR, BOR</td>
</tr>
<tr>
<td>IOPUWR (RCON&lt;14&gt;)</td>
<td>Illegal opcode or initialized W register access</td>
<td>POR, BOR</td>
</tr>
<tr>
<td>CM (RCON&lt;9&gt;)</td>
<td>Configuration Mismatch</td>
<td>POR, BOR</td>
</tr>
<tr>
<td>EXTR (RCON&lt;7&gt;)</td>
<td>MCLR# Reset</td>
<td>POR</td>
</tr>
<tr>
<td>SWR (RCON&lt;6&gt;)</td>
<td>reset instruction</td>
<td>POR, BOR</td>
</tr>
<tr>
<td>WDTO (RCON&lt;4&gt;)</td>
<td>WDT time-out</td>
<td>*<code>pwa</code>&lt;sub&gt;av&lt;/sub&gt; instruction, <em>clrd&lt;sub&gt;ct&lt;/sub&gt; instruction, POR, BOR</em></td>
</tr>
<tr>
<td>SLEEP (RCON&lt;3&gt;)</td>
<td>*<code>pwa</code>&lt;sub&gt;av&lt;/sub&gt; instruction</td>
<td>POR, BOR</td>
</tr>
<tr>
<td>IDLE (RCON&lt;2&gt;)</td>
<td>*<code>pwa</code>&lt;sub&gt;av&lt;/sub&gt; #1 instruction</td>
<td>POR, BOR</td>
</tr>
<tr>
<td>BOR (RCON&lt;1&gt;)</td>
<td>BOR</td>
<td>n/a</td>
</tr>
<tr>
<td>POR (RCON&lt;0&gt;)</td>
<td>POR</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Note: All Reset flag bits may be set or cleared by the user software.

As a Mississippi State University student I will conduct myself with honor and integrity at all times. I will not lie, cheat, or steal, nor will I accept the actions of those who do.

Signature: ___________________________ Date: ____________

Blank  blank@msstate.edu
(3 points each) A PIC24 programmed with the following code. See the previous page for reference information. What is printed on the screen? Enter X is nothing was printed.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time Period</th>
<th>Event (occurs at the beginning of the time period)</th>
<th>Message printed on screen during this time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.0 – 1.0 sec.</td>
<td>Power applied.</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>1.0 – 2.0 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>2.0 – 3.0 sec.</td>
<td>Master clear button pressed.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>3.0 – 4.0 sec.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```c
int main() {
   // Config code (not shown)
   _POR = 0;
   if (_POR) {
      outString("A");
   }

   if (_BOR) {
      outString("B");
      _BOR = 0;
   }

   if (_EXTR) {
      outString("C");
      _EXTR = 0;
      DELAY_MS(1200);
      RESET(); // Execute the reset assembly instruction.
   }

   if (_SWR) {
      outString("D");
      _SWR = 0;
   }

   DELAY_MS(1200);
   outString("E");
   DELAY_MS(500);
   outString("F");
   DELAY_MS(400);
   RESET(); // Execute the reset assembly instruction.
   outString("G");
}
```
(3 points each) A PIC24 programmed with the following code is powered on. What is printed on the screen, and at what times? Assume the watchdog timer is programmed to expire in 2 seconds. Enter X is nothing was printed.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time</th>
<th>Messages printed on screen during this time period</th>
<th>Explanation (not graded)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>0 – 1 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>1 – 2 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>2 – 3 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>3 – 4 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>4 – 5 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>5 – 6 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>6 – 7 sec.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```c
int main() {
    // Config code (not shown).
    _SWDTEN = 0;
    outString("A");
    DELAY_MS(900); // Delay 1.
    outString("B");
    _SWDTEN = 1;
    DELAY_MS(500); // Delay 2.
    outString("C");
    SLEEP();
    outString("D");
    DELAY_MS(3000); // Delay 3.
    outString("E");
    
    while (1) {
        _SWDTEN = 1;
    }
}
```
(4 points each) Use the following code snippet to answer questions 12-14.

```c
int main() {
    CONFIG_RB13_AS_DIG_OUTPUT(); // Line A
    LATB13 = 1; // Line B
    DELAY_MS(1); // Line C

    while(1) {
        DELAY_MS(100); // Line D
    }
}
```

12. Assume Line A contains the code: `ENABLE_RB13_PULLUP();`
What line of code needs to be changed (A, B, C, or D) in order for the PIC to output Z (a high-impedance value) on pin RB13? If no lines need to change, your answer should be “no change”. If this cannot be done by changing a single line, your answer should be “not possible”.

Line letter: ________

New line of code: _________

13. Assume Line A contains the code: `ENABLE_RB13_OPENDRAIN();`
What line of code needs to be changed (A, B, C, or D) in order for the PIC to output 0 V on pin RB13? If no lines need to change, your answer should be “no change”. If this cannot be done by changing a single line, your answer should be “not possible”.

Line letter: ________

New line of code: _________

14. Assume RB13 is connected to ground by a wire and Line A contains the code: `ENABLE_RB13_OPENDRAIN();`
What line of code needs to be changed (A, B, C, or D) in order for the PIC to output 3.3 V on pin RB13? If no lines need to change, your answer should be “no change”. If this cannot be done by changing a single line, your answer should be “not possible”.

Line letter: ________

New line of code: _________
15. (5 points) Given PR2 = 3, what is the time between timer interrupts, assuming Fcy = 10 MHz and T2CON = 0x8010? Show your work.

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TON</td>
<td>—</td>
<td>TSIDL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

bit 15 | bit 8

<table>
<thead>
<tr>
<th>U-0</th>
<th>RW-0</th>
<th>U-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>U-0</th>
<th>RW-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>TGATE</td>
<td>TCKPS&lt;1:0&gt;</td>
<td>T32</td>
<td>—</td>
<td>TCS</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

bit 7 | bit 0

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

bit 15

TON: Timer On bit

- **When T32 = 1 (in 32-bit Timer mode):**
  - 1 = Starts 32-bit TMRy<T:M-TMRx timer pair
  - 0 = Stops 32-bit TMRy<T:M-TMRx timer pair
- **When T32 = 0 (in 16-bit Timer mode):**
  - 1 = Starts 16-bit timer
  - 0 = Stops 16-bit timer

bit 14

Unimplemented: Read as ‘0’

bit 13

TSIDL: Stop in Idle Mode bit

- 1 = Discontinue timer operation when device enters Idle mode
- 0 = Continue timer operation in Idle mode

bit 12-7

Unimplemented: Read as ‘0’

bit 6

TGATE: Timer Gated Time Accumulation Enable bit

- **When TCS = 1:**
  - This bit is ignored
- **When TCS = 0:**
  - 1 = Gated time accumulation enabled
  - 0 = Gated time accumulation disabled

bit 5-4

TCKPS<1:0>: Timer Input Clock Prescale Select bits

- 11 = 1.256 prescale value
- 10 = 1.64 prescale value
- 01 = 1.8 prescale value
- 00 = 1.1 prescale value

bit 3

T32: 32-Bit Timer Mode Select bit

- 1 = TMRx and TMRy form a 32-bit timer
- 0 = TMRx and TMRy form separate 16-bit timer

bit 2

Unimplemented: Read as ‘0’

bit 1

TCS: Timer Clock Source Select bit

- 1 = External clock from TxCK pin (on rising edge)
- 0 = Internal clock (Fosc/2)

bit 0

Unimplemented: Read as ‘0’
(3 points each) A pushbutton attached between RB13 and ground is pressed at 500 ms, released at 1500 ms, pressed at 2500 ms, and so on. Fill in the table below. X = nothing printed.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time</th>
<th>Messages printed on screen during this time period</th>
<th>Explanation (not graded)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.</td>
<td>0 – 1 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td>1 – 2 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td>2 – 3 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td>3 – 4 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td>4 – 5 sec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td>5 – 6 sec.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```c
void _ISR_CNInterrupt(void) {
  _CNIF = 0;
  outString("A");
  if (_CNIP < 5)
    _CNIP = _CNIP + 2;
}

void _ISR_INT1Interrupt(void) {
  _INT1IF = 0;
  outString("B");
  if (_INT1IP > 0)
    _INT1IP = _INT1IP - 1;
}

void main(void) {
  CONFIG_RB13_AS_DIG_INPUT();
  ENABLE_RB13_PULLUP();
  DELAY_US(1);
  ENABLE_RB13_CN_INTERRUPT();
  _CNIF = 0;
  _CNIE = 1;
  _CNIP = 1;
  CONFIG_INT1_TO_RP(RB13_RB);
  _INT1EP = 1; // Falling edge triggered.
  _INT1IF = 0;
  _INT1IE = 1;
  _INT1IP = 2;
  while (1) { // Do nothing.
    }
  }
```

```
(4 points each) Fill in the blanks in the timing diagram below, assuming a PIC is programmed with the following code.

```c
void _ISR _T2Interrupt(void) {
    _T2IF = 0;
    _LATB12 = !_LATB12;
    PR2 = PR2 + u16_1_ms;
}

void main(void) {
    // Config RB12 as a digital output.
    // Config Timer 2 to produce interrupts.
    u16_1_ms = msToU16Ticks(1, getTimerPrescale(T2CONbits));
    PR2 = u16_1_ms - 1;
    _LATB12 = 0;

    while (1) {
    }
}
```

This diagram is NOT drawn to scale.

22. First time interval: __________

23. Second time interval: __________

24. Last time interval: __________
(4 points each) In the program below, a pushbutton switch is connected to the PIC; it pressed from 125 ms to 175 ms, then pressed at 225 ms to 325 ms, but released at all other times. A timer interrupt is configured to call `update_state()` every 50 ms, with the first call occurring at 50 ms. Fill in the table; assume no switch bounce.

```c
uint16_t u16_count = 0;

void update_state(void) {
    switch (e_state) {
        case STATE_A:
            ++u16_count;
            outString("A");
            if ((PB_PRESSED() || u16_count == 2) {
                outString("Z");
                e_state = STATE_B;
            }
            break;
        case STATE_B:
            outString("B");
            if (PB_RELEASED()) {
                outString("Y");
                e_state = STATE_A;
                u16_count = 0;
            }
            break;
    }
}
```

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time</th>
<th>Messages printed on screen during this time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.</td>
<td>0 – 75 ms</td>
<td></td>
</tr>
<tr>
<td>26.</td>
<td>75 – 125 ms</td>
<td></td>
</tr>
<tr>
<td>27.</td>
<td>125 – 175 ms</td>
<td></td>
</tr>
<tr>
<td>28.</td>
<td>175 – 225 ms</td>
<td></td>
</tr>
<tr>
<td>29.</td>
<td>225 – 275 ms</td>
<td></td>
</tr>
</tbody>
</table>