UNIVERSITY OF MANITOBA
Midterm

Winter 2007

COMPUTER SCIENCE

Real-time Systems

Date: Thursday, 1st March 2007
Time: 16:00 - 17:15
Room: EITC E2-304, University of Manitoba
(Time allowed: 65 Minutes)

NOTE:
Attempt all questions.
This is a closed book examination.
Use of non-programmable calculators is permitted.
Use of any other electronic equipment is strictly forbidden.
Show your work to receive full marks.

SURNAME: ____________________________
FORENAME(S): __________________________
STUDENT ID: __________________________

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>100</td>
</tr>
</tbody>
</table>

CONTINUED
Section A: GPIO pins and digital IO

1. Define in one sentence the main difference between a hard and soft real-time system. [5 marks]

   In a hard real-time system, the penalty of missing a deadline is orders of magnitude (often catastrophic) larger than the penalty for missed deadlines in a soft real-time system.

2. On a new development board, LEDs 0..3 are connected to pins F2,F3,F5,F7 respectively given the pullup circuit below. The NPN transistor will switch on when a high voltage is supplied from the pin, and off when a low voltage is supplied by the pin.

   A button/switch is connected to F4 given the circuit shown below.

   ![LED Circuit PIN F2, Copies for PIN F3,F5,F7]

   ![Button Circuit]

   Implement routines `initLEDs` (initialize all LEDs) and `turnOnLED_0_2` (turn on LEDs 0 (connected to PF2) and 2 (connected to PF5) and all other LEDs off).

   Make sure that you also include all necessary commands to setup port F for the LEDs.

   Any other functionality of the AtMega169 micro-controller (e.g., PortF0) must remain unchanged.

   [10 marks]
void initLEDs( void ) {
    DDRF = DDRF | 0xAC;
    PORTF = PORTF & ~0xAC;
}

void turnOnLED_0_2( void ) {
    PORTF = ( PORTF & ~ (0xAC ) ) | 0x24;
}

3. Implement routines initButton (initialize button on F4) and getButton (returns 1 if button F4 is pressed and 0 otherwise). You do not need to implement the button debouncing routine.

[10 marks]
```c
void initButton( void ) {
    DDRF = DDRF & (~ ( 0x10 ) );
    PORTF = PORTF | 0x10;
}

Version without debounce:
uint8_t getButton( void ) {
    uint8_t result = 1;
    if ( PINF & 0x08 == 0 ) {
        result = 0;
    }
    return result;
}

Version with debounce:
uint8_t getButton( void ) {
    uint8_t const MAX = 8;
    uint8_t const ON = 6;
    uint8_t const OFF = 2;
    uint8_t const MIN = 0;
    static uint8_t result = 0;
    static uint8_t count = 0;
    if ( ( PINF & 0x08 ) == 0 ) {
        if ( count < MAX ) {
            count++;
        } else {
            if ( count > MIN ) {
                count--;
            }
        }
        if ( count > ON ) {
            result = 1;
        } else if ( count < OFF ) {
            result = 0;
        } else {
            // return the prev result, result must be static
        }
    }
    return result;
}
```

CONTINUED
Section B: Timer Interrupts

4. To be able to implement more accurate delay loops, you want to generate a 10 Hz systick signal. That is every 0.1 second, the value of the 16 bit variable `systicks` is incremented. If it is impossible to generate an exact 10Hz timer signal than generate one as closely as possible to 10Hz.

Assuming that you are working on an AVR Butterfly running in its default configuration at 2MHz show the necessary values to initialize the timer subsystem.

Only change the bits that are necessary to achieve a 10Hz systick and to disable the output compare functionality. All other functionality of the AtMega169 should remain unchanged.

Show the correct value for the following registers. Use 0,1,X to indicate that a bit must be written as 0, 1, or should remain unchanged.

The current state of the bits is unknown.

<table>
<thead>
<tr>
<th>Register</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCCR0A</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OCR0A</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TIMSK0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TIFR0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

Prescaler is 64, see next question.

5. Show the interrupt service routine necessary to generate an interrupt as closely as possible to 10Hz. Also show the necessary to code to manage the variable `systicks`.

[12 marks]
/* Variable definitions */
volatile uint16_t systicks = 0;

2MHz/10 = 200,000 ticks
200000 = 64 * 3125 ,Largest prescalar that divides evenly
3125 = 125 * 25, Count from 256-125 25 times, then incr. systick

ISR( TIMER0_OVF_vector ) {
    static uint8_t count = 0;

    if ( count++ >= 25 ) {
        systicks++;
        count = 0;
    }
    T0CNT = 256-125;
}

Section C: Sound Generation

6. The following code is used to generate a variable frequency and amplitude sound wave on the speaker of the AVR Butterfly. The code uses the phase and frequency correct PWM mode.

```c
void startSineWaveInverted( void ) {
    DDRB = DDRB | (1 << PINB5);
    PORTB = PORTB | (1 << PINB5);

    TCCR1A = (1 << COM1A1);
    TCCR1B = (1 << WGM13) | (1 << CS10);

    ICR1H = 0xA0;
    ICR1L = 0x00;

    OCR1AH = 0x10;
    OCR1AL = 0x00;
}
```

Show the implementation of the code stopSineWave which turns off the sound from the speaker. Comment your code.
If it is impossible to turn off the sound on the speaker then say so in your answer and explain why.

\[5 \text{ marks}\]

```c
void stopSineWave( void ) {
    TCCR1B = TCCR1B & ( ~ 0x07 );
}
```

7. The `startSineWaveInverted` routine works correctly if the output of pin B5 is inverted before it controls the speaker. You are porting your code to a new development board where pin B5 directly controls the speaker without the inversion. What changes do you need to make to the code above to create a sine wave on the new development board. Show the implementation of the function `startSineWaveDirect`.

\[10 \text{ marks}\]

```c
    Change line 3 to
    TCCR1A = (1 << COM1A1) —— (1<<COM1A0);
```

8. The clock frequency of the processor is 2 Mhz. Calculate the frequency of the sine wave created by the routine `startSineWaveInverted`. You can assume that all other pins that affect the timing on TIMER1 are in default configuration of the AVR Butterfly.

\[10 \text{ marks}\]

CONTINUED
Section D: Instruction Timing

9. The code below is used to generate a waveform on a speaker connected to pin PE4. The system uses an 8000Hz sample rate with 2 times oversampling resulting in a bit sample rate of 16000Hz. That means a timer tick has to be generated every 32 microseconds.

The Timer2 was setup to generate an overflow interrupt every 32 microseconds and the code of the TIMER2_OVF_vector is shown below.

You can assume that all other variables and registers have been initialized correctly.

Output: timer.lst

```
12:timer.c **** ISR(TIMER2_OVF_vect) {
  .LM0:
    ...
 13:timer.c **** static uint8_t val = 0;
 14:timer.c **** static uint8_t target = 0;
 15:timer.c **** static enum State {Load,S1} state = Load;
 16:timer.c **** static uint8_t melodyIndex = 0;
 17:timer.c **** static uint8_t mask;
 18:timer.c ****
 19:timer.c **** if ( state == Load ) {
    .LM1:
      90 0012 8091 0000 lds r24,state.1300 2
      92 0016 8823 tst r24 1
      93 0018 81F4 brne .L2 1:4
      20:timer.c **** target = melody[ melodyIndex++ ];
    .LM2:
      96 001a 9091 0000 lds r25,melodyIndex.1301 2
      97 001e E92F mov r30,r25 1
      98 0020 FF27 clr r31 1
      99 0022 E050 subi r30,lo8(-(sound)) 1
      100 0024 F040 sbci r31,hi8(-(sound)) 1
      101 0026 8081 ld r24,Z 2
      102 0028 8093 0000 sts target.1296,r24 2
      103 002c 9F5F subi r25,lo8(-(1)) 1
      104 002e 9093 0000 sts melodyIndex.1301,r25 2:13
```
CONTINUED
21:timer.c  **** state = S1;
106  .LM3:
107 0032 81E0  ldi r24,lo8(1) 1
108 0034 8093 0000  sts state.1300,r24 2
109 0038 02C0  rjmp .L4 2:5
110  .L2:
22:timer.c  **** } else {  
23:timer.c  **** state = Load;
112  .LM4:
113 003a 1092 0000  sts state.1300,__zero_reg__
114  .L4:
24:timer.c  **** }
25:timer.c  ****
26:timer.c  **** if ( val >= target ) {  
116  .LM5:
117 003e 9091 0000  lds r25,val.1295 2
118 0042 8091 0000  lds r24,target.1296 2
119 0046 9817  cp r25,r24 1
120 0048 28F0  brlo .L5 2:7
27:timer.c  **** mask = ( 1 << PINB5 );
122  .LM6:
123 004a 80E2  ldi r24,lo8(32)
124 004c 8093 0000  sts mask.1302,r24
28:timer.c  **** val++;  
126  .LM7:
127 0050 9F5F  subi r25,lo8(-(1))
128 0052 03C0  rjmp .L9
129  .L5:
29:timer.c  **** } else {  
30:timer.c  **** mask = 0;
131  .LM8:
132 0054 1092 0000  sts mask.1302,__zero_reg__ 2
31:timer.c  **** val--;  
134  .LM9:
135 0058 9150  subi r25,lo8(-(1))
136  .L9:
137 005a 9093 0000  sts val.1295,r25 2
32:timer.c  **** }
33:timer.c  ****
34:timer.c  **** PORTB = ( PORTB & (~ ( 1 << PINB5 ) ) ) | mask;
139  .LM10:
140 005e 85B1  in r24,37-0x20 1
141 0060 8F7D  andi r24,lo8(-33) 1
142 0062 9091 0000  lds r25,mask.1302 2
143 0066 892B  or r24,r25 1
144 0068 85B9  out 37-0x20,r24 1
...  
146 006e 1895  reti 4:12

CONTINUED
Assume that the AVR Butterfly is running in its default configuration, with a 2MHz clock. Calculate the runtime of the code given above under the assumption that

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>Load</td>
</tr>
<tr>
<td>melodyIndex</td>
<td>32</td>
</tr>
<tr>
<td>melody[32]</td>
<td>80</td>
</tr>
<tr>
<td>val</td>
<td>30</td>
</tr>
</tbody>
</table>

You can ignore the time required to enter the ISR.

Number of clock ticks: 44.  
Runtime at 2MHz: 22 microseconds.

10. Rewrite the code in C and if necessary using the GCC asm command so that in all cases the execution time is balanced, that is the write to PORTB occurs exactly the same time after the TIMER2 overflow interrupt occured.
If it is impossible or unnecessary to modify the code in such a way, then say so in your answer and explain why.
ISR(TIMER2_OVF_vect) {
    static uint8_t val = 0;
    static uint8_t target = 0;
    static enum State {Load, S1} state = S1;
    static uint8_t melodyIndex = 0;
    static uint8_t mask;

    if ( val >= target ) {
        mask = ( 1 << PINB5 );
        val++;
    } else {
        mask = 0;
        val--;
        asm("nop");
    }

    PORTB = ( PORTB & (~ ( 1 << PINB5 ) ) ) | mask;

    if ( state == Load ) {
        target = sound[ melodyIndex++ ];
        state = S1;
    } else {
        state = Load;
    }
}

Additional work pages