Université d'Ottawa
Faculté de genie

École d'ingénierie et de technologies de l'information

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information Technology and Engineering

COURSE: CEG3136/CEG3536
SEMESTER: Fall 2009

PROFESSORS: Gilbert Arbez
DATE: Dec 18, 2007
TIME: 14h00 – 17h00

Computer Architecture II

FINAL EXAMINATION

Solution

`

NAME and STUDENT NUMBER:_____/_____

Instructions:

- Answer ALL questions on the examination paper.
- This is a close-book examination**.**
- Use the provided space to answer the following questions. If more space is needed, use the back of the page.
- Show all your calculations to obtain full marks.
- Calculators are allowed.
- Read all the questions carefully before you start.
- Pages starting at page 10 can be detached from the exam.
  (total pages 19)

There are three (3) parts in this examination.

Part 1	**Short Answer (Theory)**	**20 marks**	
Part 2	**Assembler Prog.**	**25 marks**	
Part 3	**An Alarm System**	**35 marks**	
Total		**80 marks**	

Part 1 Short Answer Questions (Theory) (total 20 points)

Mark the correct answer or give a short answer to the following questions.

1. (2 points) Interrupts are masked by default during the execution of an ISR (interrupt service routine).
a) True b) False

2. (2 points) Which of the following indicates to the HCS12 CPU where a specific ISR (interrupt service routine) is stored in memory?
a) The $\overline{\text{IRQ}}$ input b) The Reset vector
c) The Program Counter (PC) d) The interrupt vector

3. (2 points) What is the difference between the inputs $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$?

IRQ is a masked interrupt while XIRQ is a non-maskable one.

4. (4 points) The following list contains events that are performed in a CPU as an interrupt request is serviced. Re-organize the list in the order of the events occurrences and fill out the table with their corresponding indices (e.g., e a d c f b???).
a) The return address is stored on the stack.
b) The CPU is driven to the corresponding interrupt service routine ISR by the interrupt vector.
c) The CPU returns to the main program.
d) The ISR is executed.
e) The CPU tests the status of the interrupt mask bit I.
f) The return address is loaded into the Program Counter.

1	e
2	a
3	b
4	d
5	f
6	c

5. (2 points) Select the instruction that clears the flag bit C7F (the most significant bit of the timer status register TFLG1):
a) bclr TFLG1,$80 b) movb #127, TFLG1
c) bclr TFLG1,127 d) bset TFLG1,80

6. (4 points) Consider a signal, containing frequences from 10 kHz to 23 kHz (10000 to 23000 Hz), applied to a pin on the HCS12 ATD port. Assuming that this signal is converted to an 8-bit binary value using the SCAN mode and an 8 cycles used for the 2nd phase of the sampling time, determine and selecte an ATD clock frequency available in the HCS12 ATD module to process this signal. Assume a 24 MHz system clock.
Sampling frequency = 46 kHz, sampling time = conversion time= 21.7 µsec. Number of conversion cycles/conversion = 8+2+8 = 18 ATD clock cycles/conversion. ATD clock cycle = 22.7 µsec/conversion divided by 18 ATD clock cycles/conversion = 1.21 µsec/ ATD clock cycles. Minimum ATD frequency = 829 kHz. Use 860 kHz clock available in the ATD module.

7. (2 points) What is largest value that can be reached by the timer's TCNT register?

2^{16}-1 or 65535

8. (2 points) Is is possible to read the contents of the SCI Transmit Data Register? Explain your answer

No, the Transmit Data Register and Receive Data Register occupy the same address, thus when reading from the address, it is the contents of the Receive Data Register is read, not the Transmit Data Register.

Part 2 – Assembler Programming (total 25 points)

a) (16 points) The following .lst file contains a sequence of instructions that is to be run on a 68HCS12 based microsystem. Give the content of the specified registers (in hexadecimal) after the execution of the indicated instructions (with comments). The initial state of the CPU is given in Annex A. (2 points per value for a total of 14 points).

```
 1:          =00002000                          ORG $2000
 2:     2000                           main:
 3:     2000 CF 1500                           lds #$1500
 4:     2003 96 7F                             ldaa $7f
 5:     2005 CE 2500                           ldx #BCDCODES
 6:     2008 16 200C                           jsr convertBCD     PC = $200C
 7:     200B 3F                                swi
 8:
 9:        ; Subroutine: convertBCD
10:        ; Parameters: hexValue - in accumulator A
11:        ;             bcdAddr - in register Y
12:        ; Converts the hexadecimal value, hexValue, into 3 binary
13:        ; coded decimal values, to store at address provided by
14:        ; bcdAddr.  Registers used in the subroutine are preserved.
15:     200C                           convertBCD:
16:     200C 34                                PSHX
17:     200D 3B                                PSHD               SP = $14FC
18:     200E 180E                              TAB
19:     2010 87                                CLRA
20:     2011 CE 000A                           LDX #10            X = $000A
21:     2014 1810                              IDIV
22:     2016 6B 42                             STAB 2,Y           Y = $2500
23:     2018 B7C5                              XGDX          D =      $000C X = $0007
24:     201A CE 000A                           LDX #10
25:     201D 1810                              IDIV
26:     201F 6B 41                             STAB 1,Y           B = $02
27:     2021 B7C5                              XGDX
28:     2023 6B 40                             STAB 0,Y
29:     2025 3A                                PULD               SP = $14FE
30:     2026 30                                PULX
31:     2027 3D                                RTS                PC = $200B
32:
33:          =00002500                         ORG $2500
34:     2500 +0003                     BCDCODES DS.B 3   ; for storing BCD digits
```

b) (9 points) Give the contents (Hex values) of the following memory locations **after** the execution of the above code:

$2500: ***$01***

$2501: ***$02***

$2502: ***$07***

Part 3 – An Alarm System (45 points)

This part of the exam deals with emulating an alarm system using the Dragon-12 card. The bank of 8 DIP switches is used to emulate door and window contacts. LEDs and an additional DIP switch are used to emulate the alarm panel. Finally a speaker is used to emulate the alarm siren. The developpement of the alarm system is completed using three separate questions that reflect a modular design.

Question 3.1 (15 points) – Door and Window Monitor Module

Fig. 1 shows how 8 DIP switches, used to emulate the door and window contacts, are connected to the HCS12 microcontroller Port H on the Dragon-12 card. The figure also shows how the 8 LEDs are connected to Port B.

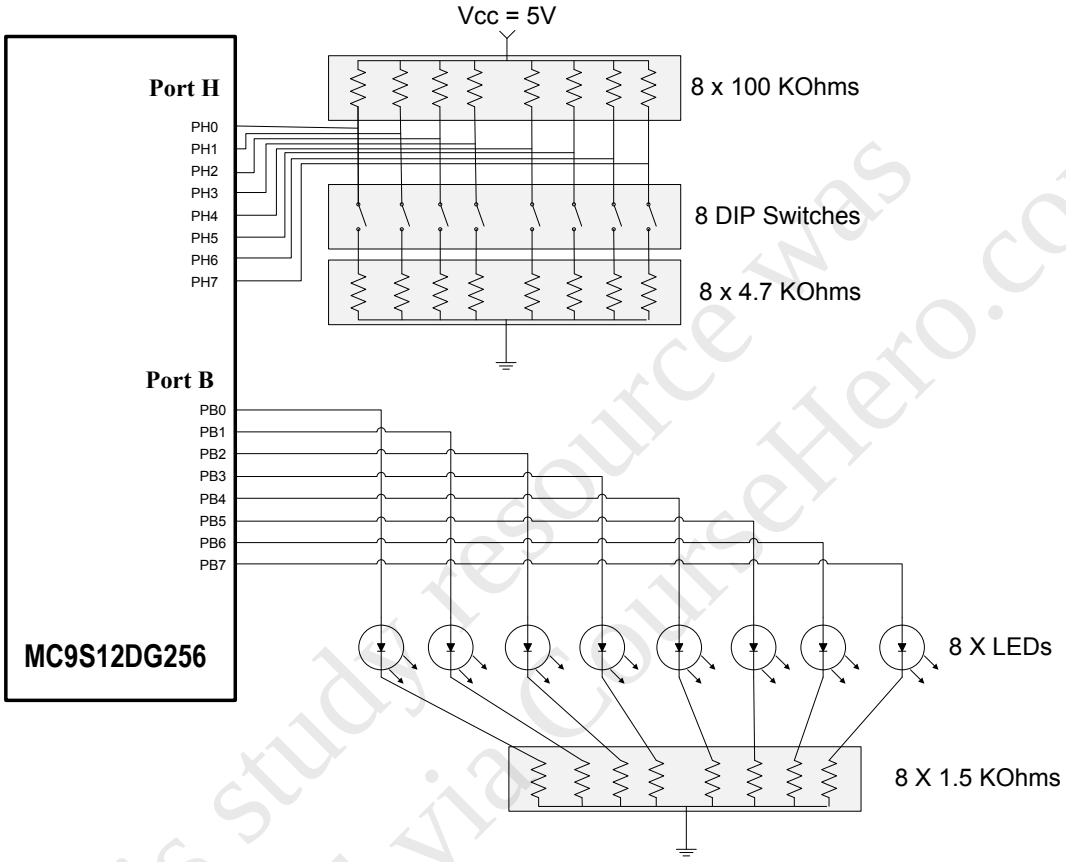

Figure 1.

A. Briefly explain the purpose of the resistors for in the circuit used to connect the DIP switches.

Sets voltage at pin to 0 when switch is closed and 5V when open. 100 Kohm resistor acts as a pullup resistor.

B. Briefly explain the purpose of the resistors in the circuit used to connect the LEDs.

Limits the current when 5 V is applied to the LED.

C. Software Design: Door/Window Monitor Module

The purpose of the module is to

- Monitor doors and windows (i.e. DIP switches) using an ISR triggered by timer channel 0 every 100 ms.
- Reflect the status of the door/window contacts on the LEDs; when a switch is open, a corresponding LED is turned ON. Assume that LED connect to PB0 reflects the status of the switch connected to PH0, LED connect to PB1 reflects the status of the switch connected to PH1, etc.
- Set a global variable, `alarmStatus`, to TRUE (value 1) when one of the switches is open (i.e. a door or window is open) and FALSE (value 0) when all switches are closed.

(a) Develop an initialisation function, `initDoorWindowMonitor()`, that setups up PORT B, PORT H and Timer channel 0 (note that for Timer Channel 0, configure only the channel, not the general functions of the Timer – this shall be done later; you may assume a Timer clock frequence of 187.5 KHz, i.e. a clock cycle of 5 1/3 μsec).

```
#define ONE_HUNDRED_MS 18750   // Number of 5 1/3 micro-sec ticks for 100 ms
void initDoorWindowMonitor()
{
   // Setup Port B
   DDRB = 0xFF;  // Setup all pins as output
   PUCR &= 0b11111101; // Deactivate pull-up registers
   // Setup Port H
   DDRH = 0x00;  // Setup as input pins - no pullups for port H
   PORTH = PORTB;  // bit=0 when switches are closed
   // Setup Timer Channel 0
   // Assume 5 1/3 micro-second ticks
   TIOS |= 0b00000001;  // Set TC0 as output-compare
   TIE |= 0b00000001;  // Enable TC0 interrupts
   TC0 = TCNT + ONE_HUNDRED_MS;  // For first interrupt in 100 ms
}
```

(b) Develop the ISR, `doorWindowMonitorISR()`.

```
#define TRUE 1
#define FALSE 0
int alarmStatus;
void interrupt VectorNumber_Vtimch0 doorWindowMonitorISR()
{
   PORTH = PORTB;  // reflect status of switches on LEDS
   if(PORTB != 0x0)  // Are all Switches closed?
        alarmStatus = FALSE
   else
        alarmStatus = TRUE;
   TC0 = TC0 + ONE_HUNDRED_MS;
}
```

Question 3.2 (15 points) – Siren Module

A speaker is connected to pin 5 (PT5) of the Timer port (controlled by timer channel 5). Generating a 1 kHz signal rectangular wave-shape with a 40% duty cycle (40% of the period, the signal is at 5V) the on PT5 to drive the speaker emulates an alarm siren.

The siren module provides the means to turn on and off the siren (i.e. control the 1 kHz signal to the speaker). Interrupts are used to generate the waveform continuously.

A. Develop an initialisation function, `initSiren()`, to setup timer channel 5. (configure only the channel, not the general functions of the Timer – this shall be done later; you may assume a Timer clock frequence of 187.5 KHz, i.e. a clock cycle of 5 1/3 μsec).

```
#define POINT_4_MS 75
#define POINT_6_MS 112
void initSiren()
{
      TIOS |= 0b00100000;  // Set TC5 to output-compare
}
```

B. Develop an the function, `turnOnSiren()`, to turn on the siren (i.e. apply a signal to the speaker)

```
#define HIGH 1
#define LOW 0
int levelTC5;  // level on TC5
void turnOnSiren()
{
      TCTL1 |= 0b00001100; // Sets high on pin 5 at output-compare event
      CFORC = 0b00100000; // Force an event on TC5 (i.e high on pin 5)
      levelTC5 = HIGH;
      TCTL1 &= 0b11110111; // Set to toggle
      TC5 = TCCNT + POINT_4_MS;
      TIE |= 0b00100000;  // Enable Interrupt.
}
```

C. Develop an the function, `turnOffSiren()`, to turn on the siren (i.e. remove a signal from the speaker).

```
void turnOffSiren()
{
      TCTL1 |= 0b00001000;
      TCTL1 &= 0b11111011; // Sets low on pin 5 at output-compare event
      CFORC = 0b00100000; // Force an event on TC5 (i.e low on pin 5)
      TIE &= 0b11011111;  // Disable Interrupt.
}
```

D. Develop the ISR, `sirenISR()`.

```
void interrupt VectorNumber_Vtimch5 sirenISR ()
{
      if(levelTC5 == HIGH)
      {
            TC5 += POINT_6_MS;
            levelTC5 = LOW;
      }
```

```
        else
        {
                TC5 += POINT_4_MS;
                levelTC5 = HIGH;
        }
}
```

Question 3.3 (15 points) Alarm Module

Figure 2 below shows the complete "Alarm System". Note that an additional DIP switch has been added to pin 0 of PORT A. The switch is debounced in hardware. See Figure 1 of Question 3.1 to see the details on how the DIP switches are connected to PORT H and how the LEDs are connected to PORT B.

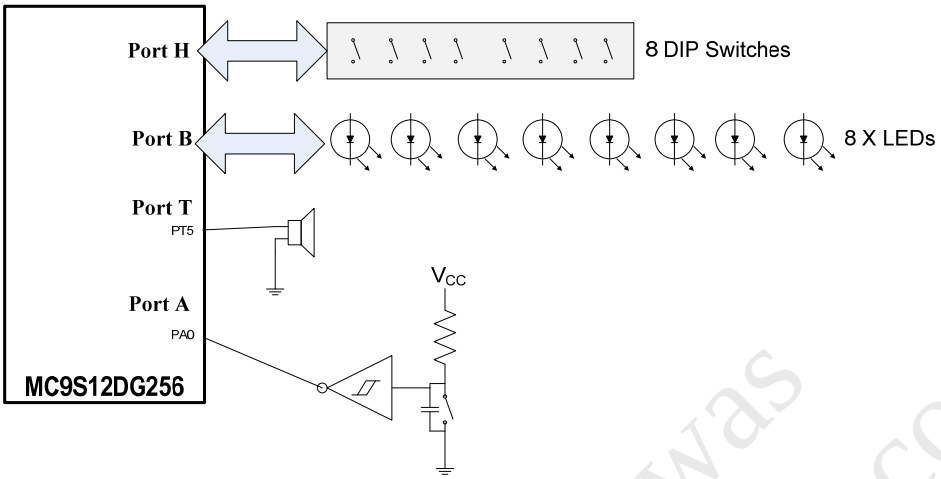

Figure 2.

The Alarm Module is the main module that controls the main function of the alarm system. Initially the system is disarmed; the LEDs still show the status of the 8 DIP switches and the siren is off. The alarm system should function as follows:

- The 8 LEDs should always reflect the status of the 8 DIP switches connected to port H, as specified in question 3.1.
- To arm the system, the switch connected to PORT A should be closed.
- When the system is armed, if one of the switches connected to PORT H (i.e. a window or door) is opened, the siren (i.e. speaker) should be turned on (used the functions in the Siren module developped in question 3.2). The siren should be maintained even if the switch is closed, that is, the only way to turn off the siren is to disarm the system.
- To disarm the system, open the switch connected to PORT A.

A. Develop the main intialisation function, `initMain()`, that intialises the global functions in the Timer (not the specific channels), PORT A, and calls the initialisation functions developed in questions 3.1 and 3.2.

```
initMain()
{
      // Setup the timer
      TSCR1 = 0b10010000;  // Enable the timer and enable fast clear
      TSCR2 = 0b00000111;  // Setup prescaler to 128, for 5 1/3 micro-sec. tick
      // Initialise PORTA
      DDRA = 0x0; // Set all pins to input
      // Call other initialisation functions
      initDoorWindowMonitor();
      initSiren();
      // Enable interrupts
      asm cli;
}
```

B. Develop the function `main()` that implements the logic of the alarm system.

```
void main()
{
	initMain();  // Initialisation
	while(1)
	{
		turnOffSiren();
		// Wait until the system is armed
		while(! (PORTA & 0b00000001)) /* wait */ ;
		// System is armed – wait until a window/door is opened
		while(!alarmStatus) /* wait */ ;
		turnOnSiren();  // turn on the siren
		while(PORTA & 0b00000001) /* wait */; /* wait until disarmed */
	}
}
```