Assignment #4 Winter 2013
SYSC 2003 Computer Organization
Department of Systems and Computer Engineering

You must submit the files identified below <u>using the electronic Submit application</u> found in your SCE account on the undergraduate network. The submission process will be canceled at the deadline. No assignments will be accepted via email or on disk.

***<u>THIS ASSIGNMENT MUST BE DONE IN TEAMS OF 2 STUDENTS.</u>* <u>Only ONE student must submit the assignment, clearly stating who the members of the team are.</u>**

Due Date: March 19th, 2013 @ 2:00 PM

<u>You must use the Pair Programming technique (check the course webpage for details). There will be questions about the assignment on the Final exam; students not being able to respond them will lose the Assignment marks.</u>

Important instructions about submissions:

- Submit ONLY ONE ASSIGNMENT PER TEAM.
- Each file in your Assignment should include a Header with the names/numbers of BOTH students
- Students submitting TWO copies of the assignment will receive the LOWER of the two marks and will receive a demerit of 20% of the final mark.
- Students who, by mistake, submit more than one copy, should contact the instructor BEFORE the deadline to fix the problem.

Objectives: Programming simple I/O devices in the Project Board. C programming of the HC12 with ICC12. Programming advanced I/O devices. Hardware interrupts.

In this assignment, we will continue building a Control system that we started discussing on Assignment 3. As a reminder, ALL THE EXERCISES MUST RUN ON THE BOARD. ALL THE RESULTS (FINAL AND/OR INTERMEDIATE) MUST APPEAR ON THE TERMINAL.

In the last three assignments in this course, we will build a Control system for a mobile robotic controller. It will include software to control the robot's speed, direction, collision detection and other control applications. We will start by creating very simple subroutines, but, in order to understand the whole idea of the system you will have built at the end of the course, we will start with a general description now. All the systems in the house are controlled using a single microcontroller.

There are a few subsystems controlled:

1. Speed: the robot's speed should be controlled according to the desired speed value. A PWM controlled motor will be used with that purpose.
2. Collision Detection: if a robot approaches an obstacle, it should rotate its direction. A pulse motor will be used with that purpose (connected to the front wheels of the robot).
3. Emergency management: the robot should detect the current temperature in the building and issue an emergency call if a high temperature is detected.
4. Interface with the user: the user can use a keypad to turn on/off the robot, change the robot's direction and speed. Also, an LCD display and LEDs are used to give information about the robot's status.

We are a part of the Software Engineering team, which is in charge of building the software for this control system. This device is going to be built by a multidisciplinary team including electrical engineers (in charge of the electrical components), mechanical engineers (in charge of the mechanical components), and non-technical staff (user manual's writers, marketing personnel, etc.).

During the requirements analysis, it was decided to build the software on an embedded platform using an HC12 microcontroller (like the one available in the lab). Although most of the hardware for the robot is not still available (i.e., the actual robot has not been built yet), the Electrical Engineering team has already designed the interfaces to be used. We know that we can use the lab's Experimental Board to start doing some basic tests that will serve as the basis for the actual house, as follows:

- The LEDs in the system will be connected to the same circuits used for the lab's LEDs interface in the lab;
- The heater will be connected to our lab's heating device interface;
- The panel display in the car is the LCD display that we have in the lab,
- The temperature sensor is connected to the AD interface (like in the lab)
- The sensors, buttons of the user interface, etc, use exactly the same interface than the one used for the keypad in our lab

Therefore, we have all the equipment needed to start developing the software for this control system. In that way, when the mechanical and electrical components of the robots are built and ready for testing, we can integrate our board, software, connect the tool interfaces to the I/O lines in the robot, and we are ready to do integration testing on the actual robot.

In order to build such complex application, we will start with very simple tasks. At this point, you will:

a. Define the different subroutines discussed here, using the Policies included on each exercise
b. Write a Main program (to be delivered with your subroutine) with the simple purpose to test your subroutine
c. Write good documentation for your main program and your subroutine

<u>In brief, you have to solve the following exercises:</u>

1. [5 marks] (**assign41.prj, assign41.c, assign41asm.s, DP256reg.s (provided), assign41.SRC, and assign41.s19)** Write a program that will detect the keys pressed in the keypad and act accordingly. The keypad contains the following keys:

column / row	0	1	2	3
0	'1'	'2'	'3'	'A'
1	'4'	'5'	'6'	'B'
2	'7'	'8'	'9'	'C'
3	'E'	'0'	'F'	'D'

We will use the keypad to operate the robot. '1' will be used to rotate to the left. 'A' to rotate to the right. 'E' to accelerate and 'D' to decelerate. 'F 'is used to power the robot on/off.

As we mentioned, we do not have a proximity sensor. When we press '5' that will emulate we are close to an obstacle.

Build a **C application** that will check the keys pressed and will act accordingly.
The program runs forever until you PRESS '0'
.
DISPLAY EACH KEY PRESSED ON THE TERMINAL (in this exercise you just need to display every key on the terminal).

2. [5 marks] (**assign42.prj, assign42.c, assign42asm.s, DP256reg.s (provided), assign42.SRC, and assign42.s19)**

Add an interface with the LCD display. This routine must:

. Display: "Speed: **** " on the first line (**** is a number representing the current speed in km/h. Every time the "E" key is pressed, this number should be incremented; when you press 'D', the number should be decremented).
. Display "Temperature XXC" (where XX is the current temperature) on the second line.

You can use Assembly or mixing C and Assembly, but your code built in Part 1 must be reused.

The program runs forever until you reset the board.

Hint:

- First build a routine that only takes care of the display with “dummy” values, and ensure they are displayed properly.
- Display every button pressed on the terminal for testing.

3. [10 marks] Build a simple sequential controller for the collision detection algorithms.

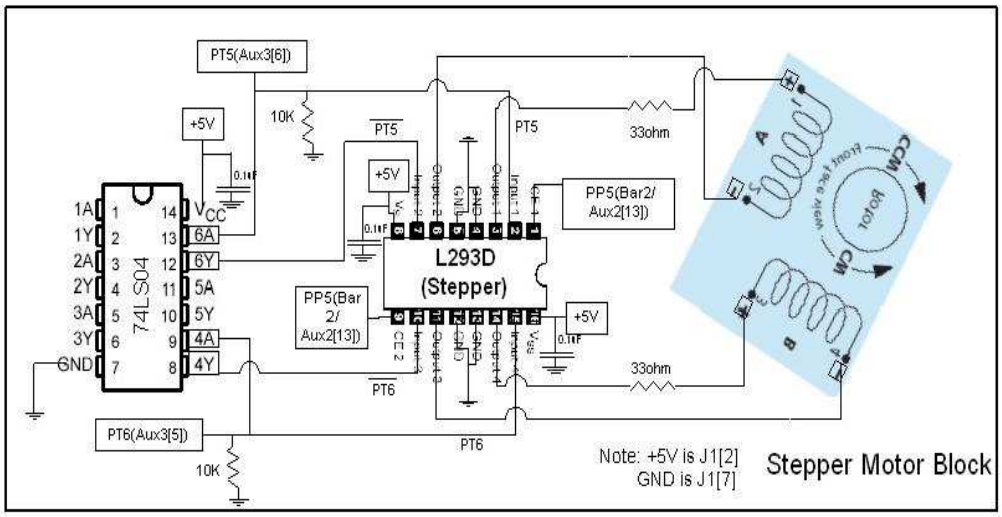
a. [5 marks] **assign3a.prj, assign3a.c, assign3basm.s, DP256reg.s (provided), assign3a.SRC, and assign3a.s19**

The Stepper Motor will be used to control robot’s direction.

Write a program that will turn the stepper motor on each of the 4 possible directions for approximately 3 seconds. IN THIS EXERCISE, DO NOT WORRY ABOUT TIMING. DO NOT USE THE TIMER ISR (unless you want to). DO NOT WORRY FOR PRECISION IN THE DIRECTION.

Your program must turn clockwise 2 turns, and then counterclockwise 2 turns (in Assignment 5 you will do this depending on the time of the day and the room’s temperature).

The basic behavior of stepper motors is described in the book. The following figure shows the actual schematic of the stepper circuit on our Project Board in the lab.



You do not have to necessarily understand the whole schematic, but you should be able to discern the signals involved in the stepper motor circuit, namely PT5, PT6 and PP5. Hence the programmer’s model for our circuit involves two ports:

	7	6	5	4	3	2	1	0	Bit
Port T	x	B	L	x	x	x	x	x	

where PT6 is an output sent to the Bottom Inductor coil of the stepper
where PT5 is an output sent to the Left inductor coil of the stepper
where x is not used in this circuit (but may be used in other circuits)

Port P

7	6	5	4	3	2	1	0	Bit
x	x	E	x	x	x	x	x	

where PP5 is the enable for the stepper
where x is not used in this circuit (but may be used in other circuits)

But how do we get the motor to move? You need to establish a sequence of values to be written to PT6 and PT5 by your program so as to induce magnetic field in a particular direction. The first thing you need to know is that the 74LS04 is an inverter. This means that we also have !PT5 and !PT6 (the inverted signals). If we now compare this circuit to the one in the textbook, we find that PT6 and !PT6 are the inputs to the left inductor coil (corresponding to PP3 and PP2 in the textbook) while PT5 and !PT5 are the inputs to the bottom inductor coil (corresponding to PP1 and PP0) in the textbook. Using these two inverted pairs (i.e. two pins instead of four separate pins) means that control of our stepper motor is limited to full-step rotations, shown in Table 7.7. You must put all this information together and formulate the values required to send our stepper motor through a (counter-) clockwise rotation. More precisely, make your own version of Table 7.7 but with only two columns of PT6 and PT5.

With your output sequence for PT6/5 in hand, there are these further considerations:

1. You must allow some time between writing out the next number of the rotation sequence for the stepper motor to physically move. Put delays of about 10 ms between writes.
2. Each number in the rotation sequence cause a *step* movement but one step is not necessarily equal to ¼ turn. In fact, on our motor, there are about four-five steps per ¼ turn. Just use four – close enough.
3. ***Beware: Both the 7-segment and the stepper use Port T.***

You can program this exercise in Assembly OR C (or a mix of both).

b. [5 marks] **assign43b.prj, assign43b.c, assign43basm.s, DP256reg.s (provided), assign43b.SRC, and assign43b.s19**

We will build a timing controller for collision detection as follows:

1) Whenever a collision is detected, the robot should stop and wait 3 seconds (in this case, just wait 3 seconds after an obstacle is detected, as the motor used to drive the robot is still disabled).
2) After 3 seconds, rotate 90 degrees to the right.

3) Wait 2 seconds (during which the robot is steering to the right; in this case, there is no motor driving the robot, so there will be no “steering”).
4) Continue straight.

The timing control must be driven by the RTI. All the activities are done by the interrupt service routine, while the main program sets up the system. You have to reset the board to finish.

Write a main program that sets up different timing/temperature values in global variables. The RTI will read these variables and will start the routines above.

5. [Bonus – 10 marks]
assign44.prj, assign44.c, assign44asm.s, DP246reg.s (provided), assign44.SRC, and assign44.s19

Put together Exercises 3a. And 3.b.

All the functions should be included now in the RTI and Keypad ISRs. You have to react to the user’s request, display the commands on the LCD display, and integrate the collision detection and rotation into the system (in this case you must use the RTI to time the execution of the opening/closing of the window).

DO NOT WORRY ABOUT INTERRUPT INTERFERENCE.

Assignment 4 Marking Criteria:

ASSIGNMENT: 20 + 10 BONUS
An assignment will receive **UNS** (UNSatisfactory 0) if it is late **OR** any of the directions are not followed (including late submission or printing rude messages).
Assignments will receive the marks specified on each exercise if it assembles and runs cleanly when run by the TA, AND subroutine policies have been followed.