Assignment #3 Winter 2013
SYSC 2003 Introduction to Real-Time Systems
Department of Systems and Computer Engineering

Due Date: February 26, 2013 @ 2:00 PM

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.

Objectives : Advanced assembly language programming: using arrays, subroutines. Getting to know the HC12 board. C programming.

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

In brief, you have to solve the following exercises:

1. [10 marks] (**Assign31.asm; Assign31.s19**) Implement the following subroutine in Assembly:

```
bool displaySystemStatus (byte speed, byte temperature,
unsigned int proximity; unsigned int &keysPressed);
```
// parameter `speed` contains the current speed of the motor in RPM. `temperature`
// contains the temperature in the robot's area. `proximity` contains information about
// the robot's direction and proximity to an obstacle. If the MSB=1, an obstacle is near.
// the 3 LSB (bits no. 0-2) represent the current robot's path: 1 = N; 2 = S; 3 =E; 4 = W;
// 5 = NE; 6 = NW; 7 = SE; 8 = SW. Bit No. 6 is used to turn on/off the motor
// Bit No. 7 is used to give the motor's direction: 1: forward; 0: backward.
// Bits No. 8 and 9 are used to rotate the robot: 01: rotate 90 degrees to the right;
// 10: rotate 90 degrees to the left.
// `keysPressed` contains information about the keys pressed
// by the user (16 keys = 16 bits; when a key is pressed, a bit is set).

	OBSTACLE	xxxxxxxxxxx	rotate	direction	power	xxxxxx	PATH
Bit No.	15	14 13 12 11 10	9 8	7	6	5 4 3	2 1 0

You must write a Main program to test the subroutine. This subroutine will be used in future assignments to display information you need. The main program must call the function with various parameters, and display the results on the terminal. Your output should look like:

Speed: < value >
Temperature: < value >
Proximity: <value> // here you must show the direction of the robot and tell if there are obstacles nearby
Keys Pressed: < value >

The values can be printed in binary or hexadecimal. Print the strings on the Terminal (using the Template routine you used for Assignment 2). The function returns a boolean value (success / error in printing; in your example, always return 1 = sucess).

2. [10 marks] (**Assign32.asm; Assign32.s19**) Write the following subroutine:

```
boolean collision_detection (byte speed, unsigned int
&proximity);
```

// parameter `speed` contains the current speed of the motor in RPM.
// `proximity` contains information about the robot's direction and proximity to an
// obstacle as discussed in Exercise 1.

The function should report (on the terminal) a collision condition, as follows:
. If there is an obstacle detected, and the robot is going in direction N or NE: rotate 90 degrees to the left
. If there is an obstacle detected, and the robot is going in direction S or SW: rotate 90 degrees to the right
. If there is obstacle detected, and the robot is going in any other direction, divide Speed by 8.
. If there is no obstacle detected, multiply Speed by 2

Write a Main program to test the subroutine. The main program must send different combinations of the three parameters, and the main program is the one in charge of displaying the results on the terminal. The outputs should change accordingly on the Terminal (using the Template routine you used for Assignment 2).

3. [10 marks] (**assign33.prj, assign33.c, ASSIGN33.SRC, and assign33.s19)** Rewrite exercise 1. in C. Use C policies for subroutines.

4. [Bonus: 5 marks]. (**assign34.prj, assign34.c, ASSIGN34.SRC, and assign34.s19)** Rewrite exercise 2. in C. Use C policies for subroutines.

5. [10 marks] (**Assign35.asm; Assign35.s19**) Write a program to turn on/off the LEDs using the subroutines used in Exercise 1 and 2. Red LED: direction = N; Green LED: direction = S; Yellow LED = Direction E; Blue LED = Direction W. Motor off = LEDs

off. Collision detected: turn on the buzzer (make it sound approximately 1 second; precision in timing is not important here).

Display the values on the terminal, as in previous Exercises.

Assignment 3 Marking Criteria:
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.