NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1 AY2008/2009

CS2271 – Embedded Systems

Nov/Dec 2008 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

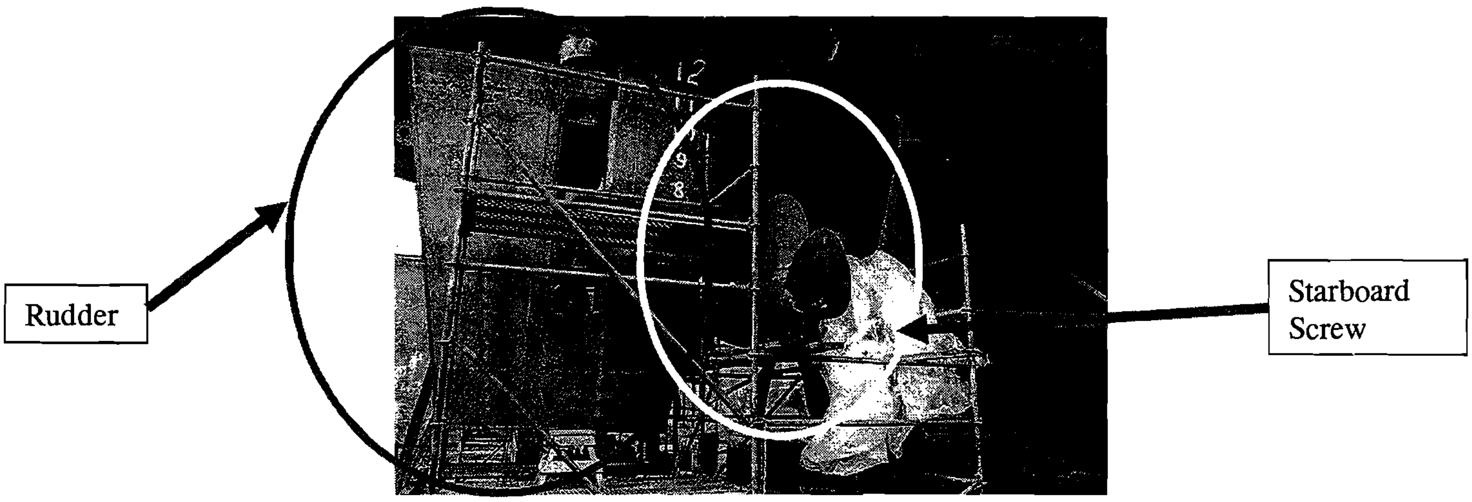
1. This examination paper contains **FOUR** questions and comprises **EIGHTEEN (18)** printed pages, including this page. The weightage for each question is as indicated, and total 100 marks.
2. Answer **ALL** questions within the spaces provided in each question. Anything written outside of the space provided will not be graded.
3. This is an Open Book examination.
4. Please write your Matriculation Number below.

MATRICULATION NO: ____________________________

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/25	
Q2	/30	
Q3	/30	
Q4	/15	
Total		

The picture below shows the rudder and propellers (screws) of a large ship. The rudder turns the ship by deflecting water, while the engine varies the rate of turn of the screws to change the speed of the ship. In addition the screws may turn in reverse to slow the ship to a stop. This ship has two screws, one on the port (left) side of the rudder and one of the starboard (right).side.



In this paper you will develop parts of a real-time system that provides navigation and control services for the ship.

The specifications of the ship are as follows:

Propulsion	A single diesel engine drives two screws at the same speed to propel the ship at a maximum speed of 25 knots (1 knot is 2.56 km/h), and a maximum <u>sustainable</u> speed of 20 knots. The screws are reversible to slow the ship down to a halt. However reversing the screw is a long and complicated process taking several minutes, and is therefore not suited for controlling the speed of the ship, only for complete braking to a halt.
Steering	A single rudder powered by its own diesel engine. Full side-to-side deflection takes approximately 2 minutes. A fully deflected rudder will turn the ship 360 degrees in approximately 8 minutes.
Navigation	A single global positioning system (GPS) receiver is used to maintain the ship's current latitude and longitude coordinates.

The following setpoints (target values for the control computers) are used:

Speed	The speed is set using a lever in the ship's bridge, as a percentage of the ship's maximum sustainable speed of 20 knots.
Waypoints	A series of GPS coordinates called waypoints is set by the navigation officer at the start of the journey. Throughout the journey the current coordinates are tracked using the GPS system. Based on the current coordinates and the coordinates of the next waypoint, a heading is worked out, and based on the current heading and speed, the rudder is deflected accordingly to turn the ship towards the new heading that will take it to the next waypoint. Once the waypoint is intercepted, it is deleted and the process is repeated for the following waypoint.

Question 1 System Design **(25 Marks)**

You are now going to design the Ship Navigation and Control System (SNCS) that the bridge officers will use to control and navigate the ship.

a. Fill the following table about the SNCS. Each entry is worth 2 marks. **(6 marks)**

i.	Is this a real time system? Why/why not?	
ii.	Is this a safety critical system? Why/why not?	
iii.	Does this system have tight timing constraints? Why/why not?	

b. List down the individual sensors (e.g. GPS receiver, etc), computer systems, actuators, input and output systems that the SNCS will have **(6 marks)**

Sensors	
Actuators	
Computer Systems	
Input/Output Systems	

c. Draw a diagram of the complete SNCS. **(6 marks)**

d. A critical problem with GPS is that the signal can sometimes be lost, or spurious signals maybe caused by reflections, interference etc, and these can frequently lead to incorrect GPS coordinate readings. An alternative system called "Inertial Navigation System" or INS, which makes use of gyroscopes (to detect turning), accelerometers (to track accelerations and speeds) and a "dead-reckoning" algorithm to determine where the ship is.

To understand "dead-reckoning", imagine that you walk south at a constant speed of 5 ms^{-1} for 6 seconds, then east at the same speed for another 4 seconds. Based on this knowledge and Pythagoras's Theorem you can "dead reckon" that you are now approximately 36 meters south-east of where you started.

Note however that the accelerometers, gyroscopes and dead-reckoning are not 100% accurate, and readings between two INS systems and the GPS sometimes vary significantly. Discuss, with a diagram, how you can make a reliable navigation system using two INS and one GPS system. Assume that the dead-reckoning algorithm has been implemented in both INS units and that you can treat all units as black boxes.**(7 marks)**

Question 2 (30 Marks)

You are now going to design the bridge circuitry between the GPS receiver chip and the ECPU that is in your lecture notes. The diagrams below show the pinouts for the ECPU, FPGA and the GPS receiver chip.

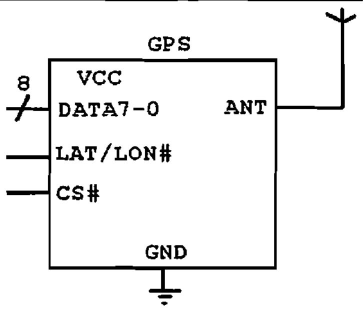
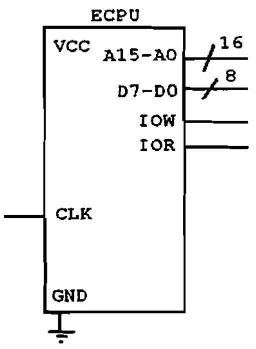
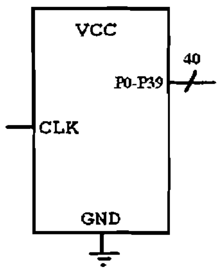
Chip	Pins	Description
	DATA7-0	8-bit data bus. Latitude and longitude are written here as 8-bit unsigned numbers.
	LAT/LON#	Latitude will be written to Data7-0 if this line is pulled HIGH and longitude will be written if this line is pulled LOW. Note that the CS# line must be pulled LOW at the same time.
	CS#	This line must be pulled LOW to read the receiver chip.
	ANT	Antenna connection. Connect to GPS antenna.
	VCC	+5v power supply connection
	GND	Ground connection
 ECPU VCC A15-A0 16 D7-D0 8 IOW IOR CLK GND	A15-A0	16 bit address bus. Device ID is placed here.
	D7-D0	8-bit data bus. Device data is written or read here.
	IOW	This line is pulled HIGH when the CPU is writing to a device
	IOR	This line is pulled HIGH when the CPU is reading from a device.
	CLK	50 MHz external clock source.
	VCC	+5v power supply connection
	GND	Ground connection
	VCC	+5v power supply connection
	CLK	50 MHz external clock source. FPGA operates internally at 50 MHz as well.
	GND	Ground connection
	P0-P39	40 general use pins.

Table 1 - Chip Pin Description

The bridge circuit will be implemented using the FPGA. The FPGA will be connected to the ECPU on one side, and to the GPS receiver on the other. The ECPU will send a latitude read request by reading from port 0x0FE0, which will cause the FPGA to first read the latitude from the GPS receiver and send it to the CPU. Similarly the ECPU will send a longitude read request by reading from port 0x0FE1, and the bridge circuit will read the longitude from the GPS receiver and send to the ECPU.

a. Using the table below, assign which pins on the FPGA (P0 to P39) you want to use for the various interfaces with the GPS receiver and the ECPU. **(3 marks)**

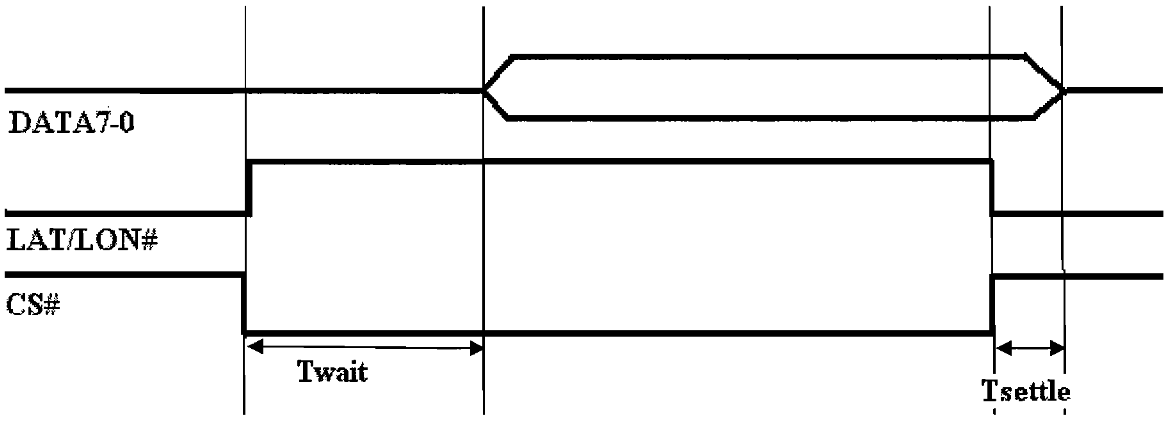
Device	Device Pins	FPGA Pin Assignment (P0, P1-P5, etc.)
ECPU	A15-A0	
	D7-D0	
	IOR	
	IOW	
GPS	D7-D0	
	CS#	
	LAT/LON#	

Table 2 – FPGA Pin Assignments

b. Based on your answers to part a and the pin specifications in Table 2 above, sketch the circuit below showing how the ECPU, FPGA and GPS receiver are connected together. You also have to sketch in connections for other pins like VCC, CLK, GND etc. **(5 marks)**

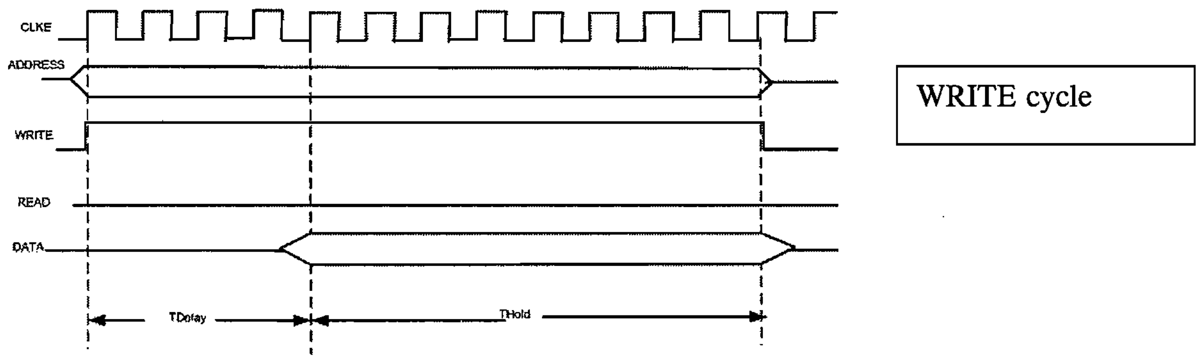
c. Based on your answers in Table 2, write the relevant **interface** statements in Handel-C. To save writing, you are allowed to write “P5”..”P1” instead of “P5”, “P4”, “P3”, “P2”, “P1”. **(5 marks)**

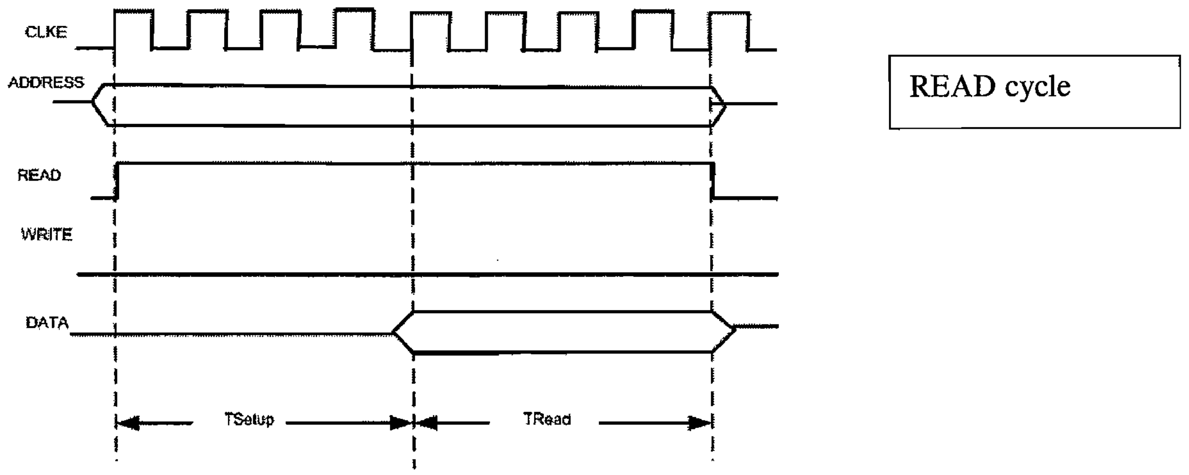
The timing diagram for reading the latitude from the GPS receiver is shown below. Reading the longitude is similar except that the LAT/LON# line is pulled low instead of high.



Timing	Description	Min	Max
Twait	Time for latitude/longitude data to appear on DATA7-0 after CS# and LAT/LON# asserted	30ns	35ns
Tsettle	Time for latitude/longitude data to subside on DATA7-0 after CS# and LAT/LON# de-asserted	5ns	12ns

The timing diagrams for the ECPU is attached below for your convenience. The external clock rate CLKE is 50MHz. You may also refer to your lecture notes for a larger copy of this diagram.





d. Write a complete Handel-C program to implement the bridge circuit, meeting the timing requirements of the GPS receiver and the ECPU, excluding the interface statements you had written above. Write on this page and the following and do not exceed the space given. Also, explain why your code meets the timing requirements of both the GPS receiver and the ECPU. **(12 marks)**

e. Suppose that there is a subroutine at memory location 0xDF00 which takes in latitude and longitude values in registers R0 and R1, and computes the distance of the lat/lon in R0 and R1 from a setpoint, and returns this distance in meters in register R0. Write an ECPU program that runs in an infinite loop, and which reads the GPS latitude and longitude coordinates, outputs it to two LED displays at port 0x0FE2 for latitude and 0x0FE3 for longitude, then calls the subroutine at 0xDF00. Your ECPU program should trigger an alarm if the distance returned exceeds 100 meters. The alarm is triggered by writing a non-zero value to port 0x0FE4, and cleared by writing a 0 to the same port number. Put your program at location 0x100 **(5 marks)**

Question 3 RTOS Programming **(30 Marks)**

We will now implement the computer that controls the ship's speed through the water. The ship's speedometer triggers an interrupt every second to pass in a 8-bit PCM coded value of the ship's current speed in knots. The 8-bit PCM represents a range of 0 to 25 knots.

This is compared against the set speed from the ship's bridge, which is set using a throttle lever. This lever triggers an interrupt, and when serviced, passes an 8-bit PCM coded value representing a percentage (0 to 100) of the ship's maximum sustainable speed of 20 knots.

a. Suppose that the actual current speed of the ship is 19.8 knots. What is the error between the PCM representation and the actual speed? In general, what is the maximum possible error in the PCM representation? **(5 marks).**

b. Suppose that a C compiler has been written for the ECPU. This compiler implements functions as subroutine calls, and arguments are passed into a function through the stack. For example, suppose that we have a function defined as:

```
int f(int foo, int bar);
```

Supposed this function is called with f(1, 2), the stack will look like this:

ECPU Stack
2
1
< return address >

For a non-void function, when the function returns, the result will be left on the stack.

The `asm` keyword is used to insert ECPU assembly language in C. Assuming the availability of an *in r0, (r1)* instruction which reads the I/O port number stored in register *r1* into register *r0* (i.e. if *r1* contains 12, this instruction will read from port 12 and put what is read into *r0*), complete the following function which reads in the port number specified by the function parameter and returns it to the caller. **(6 marks)**

```
int input(int portnum)
{
      asm
      {




      ret; Return to caller.
      }     // asm
}
```

c. Assuming the availability of an ECPU instruction *out r0, (r1)*, which outputs the contents of register *r0* into the port number indicated by *r1*, complete the following C function which takes in a port number argument *portnum* and a data argument *data*, and writes *data* to the port indicated by *portnum*. **[4 marks]**

```
void output(int portnum, int data)
{
      asm
      {















            ret ; Return to caller
      }
}
```

d. State and explain clearly whether or not it is possible to use pointers to access devices connected to the ECPU. **[4 marks]**

e. The ship's throttle is controlled through a PWM channel with an 8-bit duty cycle operating at 256 bits/s. The value $(0000\ 0000)_2$ represents 0% throttle while $(1111\ 1111)_2$ represents 100% throttle. Using MicroCOS running on a 50MHz ECPU, and the functions written in parts b and c of this question, complete the following C function that takes in a throttle percentage, and outputs the approprriate PWM waveform.

   Assume that the throttles are connected to port number 128, that function call overheads are negligible, and that the throttle hardware will lock on correctly to the start of the PCM sequence without any intervention from you. **(6 marks)**

```
void throttle_out(int percent)
{




}
```

f. You will now implement the code for a task called "EngineSpeed" which will control the throttle to maintain a fixed speed, as set in a global variable called "SPD_SETPT" protected by previously declared an initialized semaphore called SetPtSema.

   SPD_SETPT is set by a separate task called "SetEngine" which you may assume has already been written. When a speedometer reading is available, the speedometer triggers an interrupt. The speedometer ISR does nothing except release a semaphore called "Spd_ISR_Sema" which has already been declared. Your EngineSpeed task must then read port 126 for the current speed reading.

   The engine throttle and current speed (shown here as SPD_CUR) are related through the following control law:

$$P_{out}=5K(SPD\_SETPT - SPD\_CUR)$$

   Where K is some predefined constant. The value P_{out} must be clamped (limited) to between 0 and 100 inclusive before being sent to the throttle. Using the MicroCOS API and the functions defined in other parts of this question, write the EngineSpeed task which controls the throttle based on the control law given above. **(5 marks)**

(PAGE INTENTIONALLY LEFT BLANK)

```
void SetEngine()
{










}
```

(PAGE INTENTIONALLY LEFT BLANK)

Question 4 Real Time Scheduling Analysis **(15 Marks)**

The navigation system on the SNCS consists of three aperiodic tasks with the following characteristics as measured by a profiling tool.

Task	CPU Time	Shortest Period	Longest Period
P_1	1	4	6
P_2	2	6	8
P_3	3	8	10

a. What are the periods of each task in the optimistic and pessimistic cases? **(4 marks)**

Task	**Optimistic Period**	**Pessimistic Period**
P1		
P2		
P3		

b. Show whether the tasks are schedulable under RMS in the optimistic case, and in the pessimistic case. Comment on may happen if we choose to schedule based on optimistic periods. **(6 marks)**

c. Are the tasks schedulable under EDF in the optimistic case? Pessimistic case? Explain why or why not. **(5 marks)**

~~~ End of Paper ~~~
~~~