

EE 332 Final Examination

April 17, 2002

- 3 Hours
- Open Book
- Two pages plus DASH-8 manual excerpt

1. 20 Marks (5 marks each)

- Describe how devices like the DS1620 digital thermometer which use a single line for both receiving and transmitting data avoid the problem of contention for that line when switching between receive and transmit modes of operation.
- Describe the relationship between task periods and task priorities in a rate-monotonic system.
- Describe and give an example of the problem of priority inversion in a real-time application.
- Describe two mechanisms which RTEMS provides to avoid the problem of priority inversion.

2. 20 Marks (4 marks each)

Write the C code to invoke the RTEMS directives to perform the following operations. Provide comments to describe each of the directive arguments. The return status of all RTEMS directives should be checked and the routine `rtemsFail` to be called if any directive fails. This routine takes as its only argument the status code returned by the failing directive.

- Create and start two tasks.

Both tasks should be local, preemptable, operate at interrupt priority 0, allow operation of asynchronous service routines and should time-slice. Both tasks may perform floating-point arithmetic operations and are local to the processor on which they are created. The argument to each task should be the task ID of the other task. The names and entry points of the tasks should be `TskA` and `TskB`, respectively.

- Send events 3 and 7 to a task whose ID is stored in the global variable `tidA`.

- Wait for up to 10 seconds for either event 4, event 5, or both to arrive. If an event arrives, the corresponding handler routine (`event4Handler()` and/or `event5Handler()`) should be called. If neither event arrives `eventTimeout()` should be called. If the directive fails for any other reason, `rtemsFail()` should be called. The interval between RTEMS clock ticks is 25 milliseconds.

400 ticks

- Receive messages from a queue whose name is `MSQ3`. Each messages consist of between 0 and 10 double-precision floating-point variables. After verifying that the message was successfully received, your code should determine the number of variables sent in the message and then call the routine `processData` which takes two arguments. The first is a pointer to the base of an array of double-precision values to be processed and the second is the number of values to process:

```
void processData(double *dp, int n)
```

- Create a priority-inheriting mutex.

3. 5 Marks

Explain which of the following RTEMS directives could be used in an interrupt service routine, and why.

- `rtems_event_send(tid, RTEMS_EVENT0);`
- `rtems_event_receive(RTEMS_EVENT0, RTEMS_NO_WAIT, 0, &events);`
- `rtems_rate_monotonic_period(&period, 12);`
- `rtems_semaphore_obtain(semId, RTEMS_WAIT, RTEMS_NO_TIMEOUT);`
- `rtems_message_queue_urgent(msgqId, &buffer, size);`

4. 15 Marks

You are designing a real-time system using a rate-monotonic scheduler. The system will include 4 tasks with execution periods and times as shown in the following table. All values are in milliseconds and the execution times were measured on a completely unloaded processor (i.e. with neither I/O nor interrupts).

| Period | Time |
|--------|------|
| 50 | 10 |
| 100 | 20 |
| 500 | 75 |
| 1000 | 100 |

When the application is actually running, there will be DMA I/O operations running which could slow the processor by as much as 10% due to contention for the address and data buses. As well, interrupt handlers which take 1 millisecond to execute may occur at up to a 100 Hz rate.

Is this system scheduable? If so, using which rule? Show how you arrived at your decision.

5. 10 Marks

You are the member of a design team for an automated face-recognition security system. A camera subsystem will use DMA to transfer an image into memory and then generate an interrupt whenever a person passes through a doorway. Given that the inter-arrival time of images is exponentially distributed with a mean of 5 seconds, that the system contains enough memory to hold 20 images, and that the time to process an image is exponentially distributed, what average image processing time is required to ensure that no more than one image per 5000 is lost due to inadequate memory? On average, how many images will be held in the buffer? On average, how long will an image spend in the buffer and being processed? How much slower could the processor be if the image memory buffer were increased to 40 images?

6. 10 Marks

Write C I/O subroutines for an 8051 system to read and write a device similar to the DS1620, but which transfers data most-significant bit first.

7. 10 Marks

This function is used by multiple RTEMS tasks. Is it thread-safe? If not, write a thread-safe version.

```
static int logCounter (int i)
{
    static int count;

    count = count + i;
    if ((count < -10) || (count > 10))
        printf ("Count out of range!\n");
    return count;
}
```

8. 10 Marks

Modify the input routine from the PID controller example (shown below) so that it takes a single integer argument and returns the digital value corresponding to the voltage on the input channel (0-7) specified by the value of that argument. Programming information for the DASH-8 may be found on the following page. The routine must not affect the values of the digital output lines.

```
int adc (void)
{
    short v; char s;

    rtems_outb (ADC_PORT_BASE + 1, 0);
    do { s = rtems_inb (ADC_PORT_BASE + 2); } while (s & 0x80);
    v = rtems_inw (ADC_PORT_BASE);
    return (((v >> 4) & 0xFFFF) - 2048);
}
```

3.1.5 THE DASH-8 CONTROL REGISTER

The control register sets the multiplexer (channel) address, enables and disables interrupts and provides output data to the 4 general purpose digital outputs OP1-OP4. The control register is a write only register located at I/O address BASE ADDRESS + 2 (same location as status register). The data format of the control register is:-

| <u>BIT POSITION</u> | <u>D7</u> | <u>D6</u> | <u>D5</u> | <u>D4</u> | <u>D3</u> | <u>D2</u> | <u>D1</u> | <u>D0</u> |
|---------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| (BASE ADDRESS + 2) | OP4 | OP3 | OP2 | OP1 | INTE | MA2 | MA1 | MA0 |

The bits have the following significance:-

OP4-OP1: These bits correspond to the four general purpose digital output lines OP1 thru OP4. These lines can be used for external control functions e.g. driving an input sub-multiplexer to increase the number of analog input channels. A 16 channel mux. on each of DASH-8's 8 analog channels can expand the system to 128 channels.

INTE: DASH-8 generated interrupts are enabled onto any of the selected IBM P.C. interrupt levels 2-7 if INTE = 1 (logic high). Interrupts are disabled if INTE = 0 (logic low). Interrupts from the INT.IN input (pin 24) are passed through to the selected level and are positive edge triggered. It is the programmer's responsibility to set up an interrupt handling routine, interrupt vectors and initialize the 8259 interrupt controller on the IBM P.C. processor board. Writing to the control register will clear the IRQ bit of the status register.

MA2-MA0: These bits select the current analog multiplexer channel address as follows:-

| MA2 | MA1 | MA0 | CHANNEL |
|-----|-----|-----|---------|
| --- | --- | --- | ----- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

The multiplexer channel address can be determined at any time by reading the status register.

One further note about the control register. During power up of the IBM P.C. when the RESET line of the IBM P.C. is asserted, the DASH-8 control register is cleared. This insures that DASH-8 interrupts are disabled, sets digital outputs OP1-4 to zero and sets the multiplexer channel address to zero.