Name David M. Koppelman

Real Time Computing Systems
EE 4770
Final Examination

15 May 1998, 17:30-19:30 CDT

Problem 1	_____	(30 pts)
Problem 2	_____	(30 pts)
Problem 3	_____	(20 pts)
Problem 4	_____	(20 pts)
Exam Total	_____	(100 pts)

Alias The Solution

Good Luck!

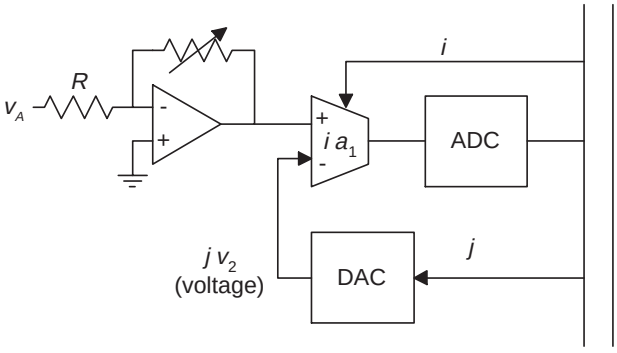
Problem 1: Precise temperature measurements over a 3 °C range are to be made using a thermistor. Though the range is 3 °C, it can fall within 250 K to 350 K, that is, it might be $[250\,\mathrm{K}, 253\,\mathrm{K}]$ for one set of measurements and $[302\,\mathrm{K}, 305\,\mathrm{K}]$ for another. Design a circuit to convert the temperature, $x \in [250\,\mathrm{K}, 350\,\mathrm{K}]$, into floating point number $H(x) = x/\,\mathrm{K}$ to be written to variable `tee` so that two consecutive measurements that fall within a 3 °C range are made precisely, as described below.

To achieve the high precision over a narrow range the circuit will use an instrumentation amplifier with an adjustable gain and a digital-to-analog converter (DAC) as a bias source, as shown below. The gain of the amplifier is set to $i \times a_1$ from the interface routine by calling `setGain(i)`, where $i \in [1, 256]$ and a_1 is a chosen value for the amplifier. The voltage is set to $j \times v_2$ by calling `setVolt(j)`, where $j \in [0, 2^{16} - 1]$ and v_2 is another chosen value. See the sample code below.

The thermistor has model function $H_{t2}(T) = R_0 e^{\beta/T}$, where $\beta = 3000\,\mathrm{K}$ and $R_0 = 0.059\,\Omega$. The circuit uses a 16-bit, 10-volt ADC. Resistor $R = 9289\,\Omega$ and voltage source $v_A = -10\,\mathrm{V}$.

- Except for a_1 and v_2 the circuit is already designed. **Do not** find values for R and v_A, use the values specified above.
- For partial credit find an interface routine (to write `tee`) assuming a fixed gain for the instrumentation amplifier (choose a convenient value) so that the entire temperature range can be measured ($[250\,\mathrm{K}, 350\,\mathrm{K}]$). Also choose a convenient value for v_-.
- For full credit the interface routine must provide the high precision for small changes from call to call. The routine should remember the previous gain and bias setting used. If the current temperature can be measured using those values they won't be changed before reading the temperature. If the temperature can't be measured (it falls outside of the range previously used) new gain and bias values should be chosen so that the current temperature is at the center of a new 3 °C range (or as close as possible), and the temperature should be measured using the new values. (The ADC output is zero for negative values and is $2^{16} - 1$ for values above 10 volts.) Be sure to provide values for a_1 and v_2. (30 pts)

```
/* Sample code. */
setGain(3); /* Set amp gain to 3 * a_1 */
setVolt(4); /* Set DAC output to 4 * v_2 */
r=readInterface();  /* Read ADC output.. */
```

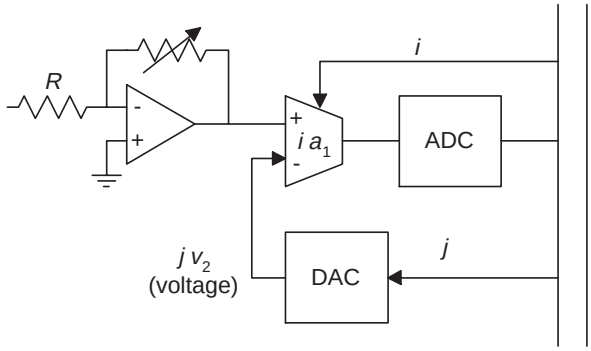


Use next page for solution, schematic and data repeated.

Problem 1, continued.

The thermistor has model function $H_{\mathrm{t2}}(T) = R_0 e^{\beta/T}$, where $\beta = 3000\,\mathrm{K}$ and $R_0 = 0.059\,\Omega$. The circuit uses a 16-bit, 10-volt ADC. Resistor $R = 9289\,\Omega$ and voltage source $v_A = -10\,\mathrm{V}$.

Temperature will be measured as follows. First the routine will determine if the temperature is out of range by checking if the ADC output is zero or $2^{16} - 1$. If it's in range the temperature will be measured without changing the gain and bias. Otherwise the amplifier gain and bias will be set so the entire temperature range can be measured ($[250\,\mathrm{K}, 350\,\mathrm{K}]$) and gain and bias settings will be computed and set so the temperature is at the center of a new $3\,^\circ\mathrm{C}$ range. (In the solution below it may take two bias settings to read the temperature.) The temperature will then be re-measured with the new settings.



To compute the gains an expression is needed for the ADC input; call that voltage $H_{\mathrm{tcc}}(x)$. Applying the response of the thermistor (H_{t2}), inverting amplifier ($H_{\mathrm{ca}}(y) = -v_{\mathrm{A}}\frac{y}{R}$), and instrumentation amplifier ($H_{\mathrm{cb}}(z) = (z - jv_2)ia_1$) yields

$$H_{\mathrm{tcc}}(x) = H_{\mathrm{cb}}(H_{\mathrm{ca}}(H_{\mathrm{t2}}(x))) = \left(-v_{\mathrm{A}}\frac{R_0 e^{\beta/x}}{R} - jv_2\right) ia_1.$$

The maximum gain is needed for the $3\,^\circ\mathrm{C}$ range where the thermistor is least sensitive, at the high end of the range, $H_{\mathrm{tcc}}(347\,\mathrm{K}) - H_{\mathrm{tcc}}(350\,\mathrm{K}) = 10\,\mathrm{V}$. Solving for the gain, ia_1:

$$ia_1 = -10\,\mathrm{V}\frac{R}{v_{\mathrm{A}} R_0 \left(e^{\beta/(347\,\mathrm{K})} - e^{\beta/(350\,\mathrm{K})}\right)} = 387.8$$

Solving $256a_1 = 387.8$ yields $a_1 = 1.515$.

The minimum gain is needed to read the full range of temperatures, the difference in voltage from the highest to lowest temperature should be 10 volts (to match the ADC), $H_{\mathrm{tcc}}(250\,\mathrm{K}) - H_{\mathrm{tcc}}(350\,\mathrm{K}) = 10\,\mathrm{V}$. Solving for the gain, ia_1:

$$ia_1 = -10\,\mathrm{V}\frac{R}{v_{\mathrm{A}} R_0 \left(e^{\beta/(250\,\mathrm{K})} - e^{\beta/(350\,\mathrm{K})}\right)} = 1.000.$$

With the value of a_1 chosen to achieve maximum gain it is not possible to read the entire range of temperatures ($1 \times 1.515) > 1$, so either the temperature window could be set to a smaller than $3\,^\circ\mathrm{C}$ range at high temperatures or the full-range temperature could be determined in two steps. The later approach is used below.

The minimum offset is needed for reading the highest temperature, $H_{\mathrm{tcc}}(350\,\mathrm{K}) = 0.33528\,\mathrm{V}$. The maximum offset is needed for the lowest temperature, $H_{\mathrm{tcc}}(251.5\,\mathrm{K}) = 9.624\,\mathrm{V}$. (The lowest measurable temperature is $250\,\mathrm{K}$, so a $3\,^\circ\mathrm{C}$ range centered there would end at $251.5\,\mathrm{K}$.) To read the full range of temperatures two biases are used, one is for the maximum temperature computed above, the other can be set halfway to the maximum instrumentation amplifier input, $5.33528\,\mathrm{V}$.

The code appears on the next page.

```
/* Solution to problem 1 (program). */

#define V2 ( 9.624 / ((1<<16)-1) )
#define A1 (387.8/256) /* = 1.51467 */
#define lowRangeBias ( 0.33528 / V2 )
#define highRangeBias ( 5.33528 / V2 )

void readTemp()
{
  static int curr_gain = -1, curr_bias = -1;
  int raw = readInterface();
  if( raw == 0 || raw == (1<<16)-1 || curr_gain == -1 )
  {
    int rawfr, bias = lowRangeBias, gain = 1;
    double tempfr, low_volt, high_volt;
    setGain(gain); setVolt(bias);
    rawfr = readInterface();
    /* Below, change bias if reading out of range. */
    if( rawfr == (1<<16)-1 ) { bias = highRangeBias; setVolt(bias); }
    tempfr = hf(rawfr, gain, bias);
    /* Note: instr. amp. input voltages lower at higher temperatures. */
    low_volt      = htcc( tempfr + 1.5 );  /* For an exact 3 C range. */
    mid_volt      = htcc( tempfr );
    high_volt     = htcc( tempfr - 1.5 );
    curr_gain     = ( 10.0 / (high_volt - low_volt) ) / A1; /* Round down. */
    true_low_volt = mid_volt - 5.0 / ( curr_gain * A1 );
    curr_bias = true_low_volt / V2;  /* Accounting for rounding of gain. */
    setGain(curr_gain); setVolt(curr_bias);
    raw = readInterface();
  }
  tee = hf( raw, curr_gain, curr_bias );
}

#define VADC 10.0
#define BETA 3000.0
#define R0 0.059
#define R  9289.0
#define VA -10.0

/* hf(raw,gain,bias): Return thermistor temperature (based on model ht2) corresponding
   to adc output RAW with instrumentation amplifier set to GAIN and BIAS. */
double hf(int raw, int gain, int bias )
{
 /* The long way. */
  double adc_input = ( ((double)raw) / (1<<16) ) * VADC;
  double instr_amp_input = adc_input / ( gain * A1 ) - bias * V2;
  double thermistor_resistance = - ( instr_amp_input / VA ) * R;
  double temp = BETA / log( thermistor_resistance / R0 ); /* log is base e */
  return temp;

 /* The short way.
  return beta / log( ( VA * raw / ( (1<<16) * gain * A1 ) + bias * V2 ) / ( VA * R0 / R )); */
}

/* htcc(temp): return instr_amp input (volts) when thermistor at TEMP. */
double htcc(double temp) {return - VA * R0 * exp( BETA / temp ) / R;}
```

Problem 2: Events and their handlers are described in the first table below. Fill in the blank tables. (30 pts) To be eligible for partial credit, show the event sequences used.

Event Name	Strong Priority	Weak Priority	Handler Run Time	Occurrence
A	3	3	$4\,\mu$s	Periodic, $25\,\mu$s period.
B	3	2	$20\,\mu$s	Periodic, $150\,\mu$s period.
C	3	1	$30\,\mu$s	Periodic, $200\,\mu$s period.
D	2	1	$300\,\mu$s	$\geq 400\,\mu$s spacing, ≤ 2 in any 2 ms period.
E	1	1	40 ms	At initialization and 40 ms after each E response.

Event	Latency	Actual Run Time	Response Time
A	$30\,\mu$s	$4\,\mu$s	$34\,\mu$s
B	$38\,\mu$s	$20\,\mu$s	$58\,\mu$s
C	$24\,\mu$s	$30\,\mu$s	$54\,\mu$s
D	$127\,\mu$s	$527.14\,\mu$s	$654\,\mu$s
E	$1054\,\mu$s	$155,800\,\mu$s	$156,800\,\mu$s

Event	Load	Load Set	Loading Factor	Loaded Duration
A	$\frac{4}{25} = 0.16$	$\emptyset$	1	$4\,\mu$s
B	$\frac{20}{150} = 0.13$	$\emptyset$	1	$20\,\mu$s
C	$\frac{30}{200} = 0.15$	$\emptyset$	1	$30\,\mu$s
D	$\frac{300}{1000} = 0.30$	$\{ A \}$	0.84	$357.14\,\mu$s
E	$\frac{40}{156.8+40} = 0.203$	$\{ A, B, C, D \}$	0.257	$155,800\,\mu$s

Total System Load: 0.947

Highlights:

The handler for event B must wait for two occurrences of A.

The handler for event D must wait for a previous occurrence of event D.

When computing D's latency the minimum separation between event D is used ($400\,\mu$s), when computing the load of D an effective period of $1000\,\mu$s is used.

Problem 3: The table below describes tasks that run on a system with task-preemptive scheduling. R1 and R2 are resource names.

Task	Priority	Arrival	Activity
A	3	13	Compute for 10, lock R1, comp. for 10, unlock R1, comp. for 10, exit.
B	2	12	Compute for 15, lock R2, comp. for 15, unlock R2, comp. for 15, exit.
C	1	0	Compute for 11, lock R1, comp. for 11, unlock R1, comp. for 11, exit.
D			Compute for 17, lock R2, comp. for 17, unlock R2, comp. for 17, exit.

(*a*) Suppose the system does not employ any locking protocol. Assign priorities and arrival times so that one of the tasks suffers priority inversion. Show CPU activity and task states for these assignments, identify the suffering task. (Not all tasks have to be used.) (7 pts)

(Task states not shown, but should be in a complete solution. This should be finished before the Spring 1999 final exam, remind me to finish it if it's April 1999 or later.) Task A suffers priority inversion because while waiting for resource R1 it must wait for B to finish running (even though B does not use R1) so that C can run and release the resource.

(*b*) Show CPU activity and task states with the arrival times and priorities chosen above but for a system using a priority inheritance protocol. (6 pts)

With priority inheritance task C gets priority 3 as soon as A attempts to lock the resource. This way B doesn't run while A is waiting.

Problem 3, continued.

(*c*) Add another task that would have a lower maximum blocking time with a priority ceiling protocol (either ceiling protocol) than with a priority inheritance protocol. Specify the task's priority and activity and any other necessary data. Explain how the ceiling protocol helps, execution (CPU activity and task states) does **not** have to be shown. (7 pts)

Note: In the original exam the word "maximum" was not included so a much simpler solution was possible.

With inheritance protocols a higher priority task need only wait (be blocked) for the critical regions of lower-priority tasks. But the number of critical regions it would have to wait for is the number of distinct resources it would lock, if there are many resources the blocking time can still be high. Here there are only two resources, so the time isn't that bad. A task that might encounter this not-so-bad blocking is E in the table below:

Task	Priority	Arrival	Activity
E	4	27	Compute for 10, lock R1, lock R2, comp. for 10, unlock R2, unlock R1, comp. for 10, exit.
A	3	16	Compute for 10, lock R1, comp. for 10, unlock R1, comp. for 10, exit.
B	2	0	Compute for 15, lock R2, comp. for 15, unlock R2, comp. for 15, exit.
C			Compute for 11, lock R1, comp. for 11, unlock R1, comp. for 11, exit.
D			Compute for 17, lock R2, comp. for 17, unlock R2, comp. for 17, exit.

In the system described above E must wait for both A and B to finish their critical regions. An immediate ceiling protocol in which R1 has ceiling 4 and R2 has ceiling 5 would avoid the problem since A could not run while B had R2 locked (so R1 could not get locked).

Problem 4: Answer each question below.

(*a*) How is the Schedulability test below used? What do the symbols denote? For what type of system is it used? What does it mean when the relation holds, and when it does not hold?

(5 pts)

$$L < N(2^{1/N} - 1)$$

The test is applied to systems in which rate-monotonic scheduling is used on purely periodic events (or tasks). Symbol N is the number of tasks and L is the load imposed by the tasks. If the relation holds then tasks will finish before the next corresponding event. (That is, their response times will be less than their periods, so deadlines will be met if the deadlines are equal tot he periods.) If the relation does not hold then the response times may or may not be less than their periods, other analysis methods have to be used.

(*b*) Describe a similarity and a significant difference between a phototransistor and a photomultiplier. The similarity should be something that sets the two apart from many other light sensors. The difference should not refer to physical structure. (For partial credit, mention physical structure.) (5 pts)

Both devices are closely integrated with amplifiers. A significant difference is the photomultiplier's very high sensitivity (and cost of the device itself and the conditioning circuit).

(*c*) What is the difference between the running of an interrupt handler and a task? What are some restrictions that might be placed on an interrupt handler that would not be placed on a task? (5 pts)

A task can allocate resources, and can otherwise make use of OS services. Interrupt handlers run in a task's context and are very limited in what additional resources they can request. Context switching during the execution of a handler is usually not allowed. (An answer given in many papers was that an interrupt handler cannot itself be interrupted. This is true on many systems but not the systems considered in class. For example, problem 2 would be much easier if handlers could not get interrupted.)

(*d*) What is the difference between irradiance and illuminance? What is the difference between irradiance and radiant flux? (5 pts)

Irradiance is flux per unit area, illuminance is flux per unit area considering human perception. (Photons are weighted by their visibility to the human eye.)

Irradiance is flux per unit area, radiant flux is total flux (light output, power).