Name:________________________________

Dordt University Engineering Department

EGR 304, Microprocessor Interfacing — Final Exam, May 4, 2017.
Open book, open notes, take-home.
An Internet-connected computer is required. A calculator is also required.

1.) A small local-area computer network (e.g. a network of two or three computers sharing files, printers, etc.—no Internet connection) could be made using USB connections or using Ethernet connections between the computers. Describe some of the challenges and advantages that would be encountered when using a USB connection for networking as opposed to using an Ethernet connection. How are these challenges rooted in the technological differences of the two connection types? (10 points)

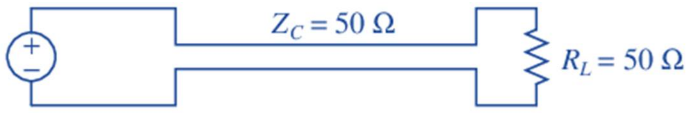
USB is not designed for networking. USB requires that **each computer acts as a host controller** to arbitrate access to the USB network. In order to connect two or more computers together with USB some hardware device will need to buffer the data flow between computers and provide the **client-side connection**. This device will act as a slave with respect to each networked computer. Ethernet on the other hand is designed as a **peer-to-peer** network. The hubs are designed to arbitrate access.

USB connections are limited to about 15 feet unless hubs or other hardware is used. This means the "small local-area computer network" needs to be physically very small indeed, probably all in the same room. Ethernet on the other hand has a typical connection length limit of 100 m or about 330 feet. This provides for a much larger physical area in which the network may exist.

2.) A thirty-foot long asynchronous serial computer interface uses RS-232-style signaling but at "TTL" logic levels meaning 0 volts for logic-1 and idle and mark or 5 volts for logic-0 or space, nominal levels. The interface is in practically continuous use. (There are no significant periods of idle time.) About half of the time the transmission line is sending logic-1 and the other half of the time it is sending logic-0. A low-impedance driver is connected to a 50 Ω transmission line and a 50 Ω matched load is used to prevent reflections. It works fine but battery life is poor due to all the current flowing in the load. Suggest a change or changes to the source and/or load impedances that will improve battery life. (10 points)

Power is being used to drive the 50 Ω load half of the time.

Before:



$$\text{Before: } P = \frac{V^2}{R}(duty\ factor) = \frac{(5\text{ V})^2}{50\ \Omega}(0.5) = \frac{1}{8}\text{W}$$

Changing the load to a high impedance, say 50 kΩ or more, would greatly reduce the power used.

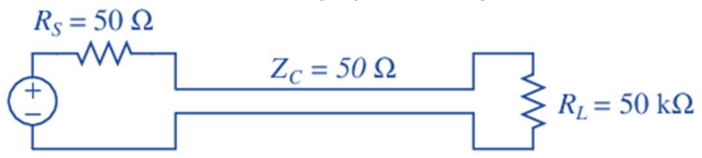
$$\text{After: } P = \frac{V^2}{R}(duty\ factor) = \frac{(5\text{ V})^2}{50\ k\Omega}(0.5) = \frac{1}{8}\text{mW}$$

Except just changing the load resistance would create reflections, $\Gamma_L = (R_L - Z_C)/(R_L + Z_C) = +1$
I'll change the source resistance to 50 Ω. Then the reflection coefficient at that end will be zero.

$$\Gamma_S = (R_S - Z_C)/(R_S + Z_C) = 0$$

Thus the source end will absorb all the reflections. The asymptotic voltage at the load will still be 5 V.

After:



3.) A DC motor with a permanent-magnet field, when connected to a 12 V DC source has a no-load speed of 10 000 RPM and a stall (locked rotor) torque (a.k.a. locked rotor torque) of 0.1 Nm. Assume the motor has an ideal linear torque vs. speed characteristic. This motor is connected to a low-impedance 12 V source on the electrical side and to a mechanical load that requires a driving torque that is related to the speed such that

$$T = \sqrt{S}\left(\frac{1}{2500}\,\mathrm{Nm/RPM}^{0.5}\right)$$

Where T = required driving torque (in Nm)
S = load speed (in RPM)

Find the speed at which the motor will operate. (10 points)

The motor's torque vs. speed characteristic is $T = 0.1 - \frac{S}{100000}$ Set that equal to the load's characteristic, solve for S.

$$\frac{\sqrt{S}}{2500} = 0.1 - \frac{S}{100000}$$

$$40\sqrt{S} = 10000 - S$$

Let $x = \sqrt{S}$

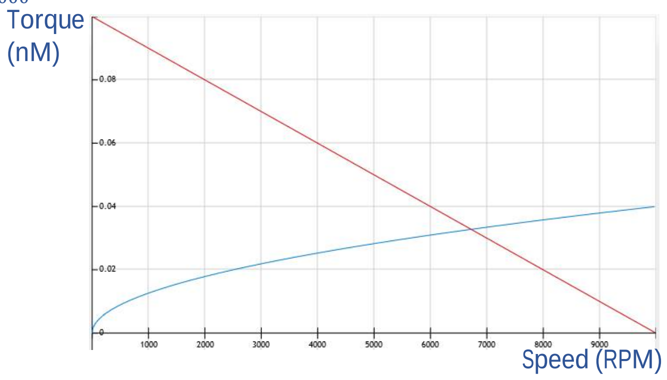
$$x^2 + 40S - 10000 = 0$$

Apply quadratic formula

$$x = 81.98 \; or - 243.96$$

Only 81.96 makes sense here.

$$S = x^2 = 6721 \; RPM$$



4.) Describe in words or pseudo-code how to use an interrupt driven tic-clock system with a microcontroller to generate a pulse of exactly 10 ms each time a certain subroutine is called. Assume the tic-clock interrupt has a frequency of 10 kHz. (10 points)

Tic clock frequency of 10 kHz implies a tic clock period of $t_{ptic} = 1/(10\text{ kHz}) = 0.1$ ms.

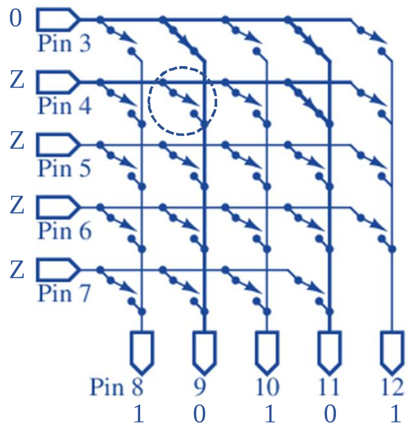
A 10 ms pulse will require $(10\text{ ms})/t_{ptic} = 100$ tic-click periods. I assume here that the tic clock system has a task scheduler. If not, add one.

When the subroutine to start the pulse is called it reads the present tic clock value, then adds a small amount to that value to place the pulse slightly in the future (to give time to load the value into the scheduler). Then the subroutine schedules the start of the pulse. The subroutine then adds 100 tics to the start-value and enters that time into the schedule table to schedule the end of the pulse. (Alternatively, the task that turns the pulse on can add 100 tics to its scheduled time and schedule the pulse's turn off time.)

The task scheduler will then start and stop the pulse at the correct tic-clock interrupts.

5.) Consider creating a keyboard-operated music synthesizer by connecting a keyboard to an Arduino microcontroller. Let the synthesizer have 24 keys (two octaves). Each key will have just one SPST switch. The keyboard has a 48-conductor ribbon cable connected to it, two conductors dedicated to each switch. Could this keyboard be connected directly to an Arduino microcontroller board using no additional parts such as diodes, gates, resistors, or transistors? The Arduino board has 13 digital I/O pins available. If so, describe how and describe any limitations of the connection. If not, give a rationale for your conclusion. (10 points)

Observe that $\sqrt{24} = 4.899$. Rounding up gives 5.
Put the key switches in a 5 x 5 matrix.
Choose ten digital I/O pins on the Arduino, say 3-12.
Connect five of the ten pins to the rows of the matrix. Say pins 3 – 7
Connect the other five to the columns of the matrix. Say pins 8 – 12
Set up the rows as outputs with open-collector drive.
Set up the columns as inputs with internal pull-up resistors enabled.
Connect one keyboard switch at each intersection in the matrix.
Leave one intersection unconnected.
(Because 5 x 5 = 25 but there are only 24 switches.)
A schematic of this set-up is shown at the right.



The rows should be programmed so that one of them is logic-zero at any given time. Designate the row that has the logic-0 as the *selected row*. All the other rows will not be driven (neither to 1 nor 0) due to the open collector drive. Select a row, say the one connected to pin 3. Read the columns (pins 8-12). Assume for the moment that no keys in any non-selected row are pressed down. Any logic-0 detected represents a key that is pressed down. The reading of the columns will correctly represent the state of the entire keyboard.

Now consider pressing additional keys. A keypress on one of the non-selected rows (e.g. row 4, column 11) will connect a column to an open-collector driver. This will not interfere with reading the selected row's keys if it is the only key pressed on a non-selected row. Each non-selected row may have any one key pressed without causing trouble reading the selected row. Finally, consider pressing two keys on a non-selected row. Suppose row 4 is selected and keys at 3-9, 3-11, and 4-11 are pressed. At the intersection of row 4 and column 9, as illustrated by the dotted circle in the schematic It will appear as if the switch is closed due to the connection through row 3 when row 4 is logic 0 (selected). This will cause an error. O**ne can conclude that any two keys down simultaneously can be correctly read by quickly scanning the rows and reading the columns**, but if three or more keys are down there may be errors although as many as five simultaneous key presses can be correctly detected if they are all on one row—all the keys on one row down with no keys down on each of the other rows. By a judicious mapping of keys into the matrix, a few of the most common musical chords can be played. This is a common strategy in toy keyboards.

Using more columns (and fewer rows) than needed is also a valid way to enable more chords to be played. Software that keeps track of past key presses and releases (sequential key state detection) can further improve the performance of this system by mapping changes in scan results to the sequence of key presses needed to achieve the sequence. Finally, the roles of rows and columns may be interchanged (no advantage, just a variation).

6.) What is the difference between vectored interrupts and polled interrupts. (10 points)

Vectored interrupts are supported by dedicated hardware interrupt request signals so that a specific interrupt service routine (ISR) can be called in response to each different type of interrupt request.

With polled interrupts there is a common interrupt request and a common ISR which must perform normal I/O operations to determine the source of the interrupt.

7.) A microcontroller has three sources of interrupts. Any of these can be eliminated by adding additional hardware. To keep costs down it is desirable to have the microcontroller handle as many of the interrupts as possible. Can the microprocessor handle all the interrupts below? If not, suggest which one (or ones) should be eliminated (requiring additional hardware). Try to eliminate as few interrupts as possible. Justify your answer. (15 points)

Listed from highest priority (on top) to lowest (on bottom)

Source 1) The interrupt service routine (ISR) takes 50 µs to run. Minimum time between interrupts is 100 µs. Prompt response to the interrupt request is not needed as long as the interrupt is serviced before it is requested again.

Source 2) ISR takes 10 µs. Minimum time between interrupts is 500 µs but the maximum allowable latency (time from request to end of execution of ISR) is 100 µs.

Source 3) ISR takes 5 µs. Minimum time between interrupts is 70 µs. Prompt response is not needed as long as the interrupt is serviced before it is requested again.

The longest critical region in the main software is 8 µs and the interrupts are disabled during each ISR. The longest instruction takes 1 µs.

(Please write your answer on the next page.)

(This page is provided for the answer to the problem stated on the previous page.)

Summary of the question: Will interrupts work? Eliminate as few as possible if any need to be eliminated. Justify your answer.

Source 1) ISR takes 50 µs to run, 100 µs between interrupts.
Source 2) ISR takes 10 µs to run, 500 µs between interrupts, latency < 100 µs req'd.
Source 3) ISR takes 5 µs to run, 70 µs between interrupts
Critical region is 8 µs, interrupts disabled during ISRs, longest instruction is 1 µs.

$T_{P1} = 100\,\mu s \qquad T_1 = 50\,\mu s \qquad T_{1+} = 10\,\mu s$

$T_{P2} = 100\,\mu s \qquad T_2 = 10\,\mu s \qquad T_{2+} = 8\,\mu s$

$T_{P3} = 70\,\mu s \qquad T_3 = 5\,\mu s \qquad T_{3+} = 8\,\mu s$

CHECK DENSITY

$\frac{50}{100} + \frac{10}{100} + \frac{5}{70} = 0.591 < 1$ OK

INTERRUPT INTERVALS

#1 $T_{1+} + N(1,1)T_1 \overset{?}{\leq} T_{P1}$

$10 + (1)(50) \overset{?}{\leq} 100$

$60 \leq 100$ OK

$N(2,1) = \left\lceil \frac{T_{P2} - T_2}{T_{P1}} \right\rceil = \left\lceil \frac{100-10}{100} \right\rceil = 1$

MUST NOT USE 500 µs HERE!

#2 $T_{2+} + N(2,1)T_1 + N(2,2)T_2 \overset{?}{\leq} T_{P2}$

$8 + (1)(50) + (1)(10) \overset{?}{\leq} 100$

$68 \leq 100$ OK

$N(3,1) = \left\lceil \frac{T_{P3} - T_3}{T_{P1}} \right\rceil = \left\lceil \frac{70-5}{100} \right\rceil = 1$

$N(3,2) = \left\lceil \frac{T_{P3} - T_3}{T_{P2}} \right\rceil = \left\lceil \frac{70-5}{100} \right\rceil = 1$

OR COULD USE 500 µs HERE

#3 $T_{3+} + N(3,1)T_1 + N(3,2)T_2 + N(3,3)T_3 \overset{?}{\leq} T_{P3}$

$8 + (1)50 + (1)10 + (1)(5) \overset{?}{\leq} 70$

$73 \not\leq 70$ FAILURE

TO FIX: COULD ELIMINATE ISR 3 OR, OBSERVE THAT INTERVAL #1 HAS MARGIN DEMOTE ISR #1 TO LOWEST PRIORITY, THEN

SOURCE 1 (FORMERLY 2) $T_{P1} = 100\,\mu s \quad T_1 = 10\,\mu s \quad T_{1+} = 50\,\mu s$

SOURCE 2 (FORMERLY 3) $T_{P2} = 70\,\mu s \quad T_2 = 5\,\mu s \quad T_{2+} = 50\,\mu s$

SOURCE 3 (FORMERLY 1) $T_{P3} = 100\,\mu s \quad T_3 = 50\,\mu s \quad T_{3+} = 8\,\mu s$

$N(2,1) = \left\lceil \frac{T_{P2} - T_2}{T_{P1}} \right\rceil = \left\lceil \frac{100-10}{100} \right\rceil = 1$

$N(3,1) = \left\lceil \frac{T_{P3} - T_3}{T_{P1}} \right\rceil = \left\lceil \frac{70-5}{100} \right\rceil = 1$

$N(3,2) = \left\lceil \frac{T_{P3} - T_3}{T_{P2}} \right\rceil = \left\lceil \frac{70-5}{100} \right\rceil = 1$

INTERRUPT INTERVALS

#1 $T_{1+} + N(1,1)T_1 \overset{?}{\leq} T_{P1} \Rightarrow 50 + (1)10 \leq 100$ OK

#2 $T_{2+} + N(2,1)T_1 + N(2,2)T_2 \overset{?}{\leq} T_{P2} \Rightarrow 50 + (1)10 + 5 \overset{?}{\leq} 70 \Rightarrow 65 \leq 70$ OK

#3 $T_{3+} + N(3,1)T_1 + N(3,2)T_2 + N(3,3)T_3 \overset{?}{\leq} T_{P3} \Rightarrow 8 + (1)10 + (1)5 + (1)50 \overset{?}{\leq} 100 \Rightarrow 73 < 100$ OK

$\therefore$ IT CAN WORK WITHOUT ELIMINATING ANY INTERRUPTS.

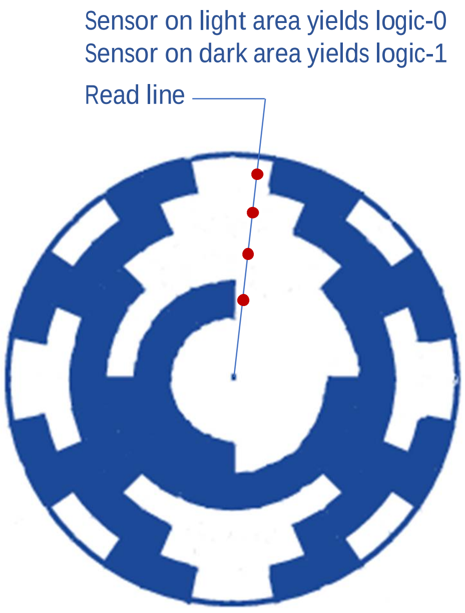
8.) Design a rotational position sensor for sensing ***absolute*** position with a resolution of 11.25 degrees (1/32 of a revolution). The rotational position sensor needs to only keep track of one revolution. If the shaft of the sensor should roll past the home position, the position code can roll over too. Each time the rotational position of a disk changes by 1/32 of a revolution, an interrupt should be produced. You may assume that the interrupt will be serviced before the disk moves enough to generate another interrupt request. You may decide for yourself how much logic to do in hardware and how much to do in the ISR. Your design should be relatively easy to manufacture. (15 points)

a.) Show the design of the disk and number and location of the sensor(s).

b.) Show a flow chart (or equivalent pseudo-code) to describe the processing that needs to be done in the ISR to keep track of ***absolute*** position.

A disk is encoded with gray code as shown at the right. Because of the nature of gray code, only one bit can change at a time. When any sensor data-bit changes logic-level, the new read-line word should be latched into a buffer and a hardware interrupt should be generated. It is given in the problem statement that one may assume the interrupt service routine (ISR) will finish execution before another bit can change.

The ISR will read the buffered value of the read-line data. it will use this to look up the angular position in a table. The table is shown below.


Sensor on light area yields logic-0
Sensor on dark area yields logic-1
Read line

Each red dot represents a sensor.

Inner-to-outer track on disk	*Disk Position*
00000	0
00001	1
00011	2
00010	3
00110	4
00111	5
00101	6
00100	7
01100	8
01101	9
01111	10
01110	11
01010	12
01011	13
01001	14
01000	15
11000	16
11001	17
.	.
.	.
.	.
10000	31

Rotation in degrees is *Disk Position* times 11.25°.

Pseudo code:
Enable ISR on any bit change.
Hardware buffer the state of all 5 sensors immediately after
any bit change.

ISR reads senor buffer.
Look up *Disk Position*
Return angle in degrees as Disk Position x 11.25

Another acceptable solution uses a binary coded disk with V-scan or U-scan sensors. For this case one needs 5 tracks and nine sensors, two sensors on a track except one sensor. on the track for the LSB.

9. The performance of a car's shock absorber needs to be monitored to assist with the design and testing of the shock absorber. The fastest rate at which the shock absorber might effectively move to absorb the energy of bumps is 100 Hz. (100 bumps per second for example.)

The transducer used is a linear analog position sensor. It produces a voltage proportional to the length between the two anchor points of the shock absorber. The maximum voltage is +0.100 V at full extension of the shock absorber and the minimum is –0.100 V at full compression. The transducer has a frequency response up to 10000 Hz. Thus, it is not only is sensitive to the motion of the shock absorber, it is also sensitive to some of the sound of the wheel rolling over the road and conducted into the shock-absorber.

A commercially available analog-to-digital converter is in use. The sample rate of the converter is 1000 Hz. (One sample is taken every millisecond). This analog-to-digital converter has an input range of +10 V to –10 V and provides a 10-bit output in a two's complement format such that +10 V produces the code $01\ 1111\ 1111_{TC}$ which corresponds to $+511_{10}$ and –10 V produces the code $10\ 0000\ 0001_{TC}$ which corresponds to $–511_{TC}$. This converter has no anti-aliasing filter. The user is expected to add their own anti-aliasing filter.

The user figured that the 10000 Hz frequency response of the transducer and the 1000 Hz sample rate are 100 and 10 times (respectively) more than needed since the user is only interested in motions at 100 Hz or less. Thus, these were perceived as very good specifications. (The user may be naive in this determination.) The ±10 V input range is also perceived as 100 times greater than needed so that should also be very good. (The user may be naive in this determination.) The user is getting poor results from this system.

For an anti-aliasing filter the user has resorted to averaging 8 sequential samples (using digital addition of the sampled data, then dividing by 8) in order to get better results. This has reduced their effective sample rate to 1000/8 = 125 Hz. (Every eight digital samples get averaged to produce just one average sample for the interval.) However, the samples still are not accurate.

Assume that averaging digital samples must be abandoned as it is an incorrect method of addressing the problems here. Assume analog filters and amplifiers are available to any specifications you might like. For a filter, specify if it should pass low or high frequencies and what the cutoff frequency should be. For an amplifier, specify the gain. How can you use filters and amplifiers to improve the results? Explain your choices. (10 points)

The Nyquist low-pass sampling theorem tells us that an analog anti-aliasing filter must be used before the sampler and it must cut off all signal above half the sampling rate. In this case the sampling rate is 1000 Hz so the anti-aliasing **low-pass filter** must cut off everything above 500 Hz. The signal only has meaningful content up to 100 Hz, thus **any cut-off frequency between 100 and 500 Hz** will work. The digital anti-aliasing filter the user has described is pointless because the signal is already suffering from aliasing before the filter is applied. The digital filter should be removed from the system and an analog filter should replace it before the signal is sampled.

The signal amplitude of ±0.1 V is 100 times smaller than the range of the A/D converter which is ±10 V. This means that out of the 1024 codes available from the converter, only 1/100 of them, or about 10 of them, will be used! This is a poor use of the available dynamic range of the converter. **An amplifier with a gain of 100** placed before the A/D converter will fix this.