EE 478 Lab 2

Designing a High Reliability Microprocessor Based Remote Surgery System (Including Design Trade-offs)

University of Washington - Department of Electrical Engineering

James K. Peckol

Lab Objectives:

The objectives of this lab are to learn the 18F25K22 PIC microprocessor, to utilize our knowledge of the SRAM, to learn simple networking and protocols, and to learn and practice the full product development life cycle while documenting, designing, building, and testing a highly reliable (fail operational), distributed, remotely controlled network based device.

Background Information:

You have just returned from a meeting at Johns Hopkins University with a number of the top researchers in the field of remotely controlled surgery. Anxious to explore some of the ideas and problems discussed, you decide to design and implement a working prototype to study the feasibility of and potential problems affecting such a system.

Supporting Material:

The EE 472 text contains several chapters that are directly relevant to the tasks presented in this lab. You are strongly encouraged to look through those. These include: Chapters 5, 8, 9, 10, 15, 16, and 17. Chapter 12 (corresponding to the text's Chapter 9) is also on line under *documentation* and labeled *Design Cycle*. You will find this particularly relevant during the specification and early design phases.

You can also find a discussion in the class lecture notes, also posted on line.

Cautions and Warnings:

When you are working with the EIA-232 (RS-232) portion of the design, make certain to check all of your connections, signals and voltage levels prior to connecting to the computers. We do not want to risk damaging the equipment.

Contrary to beliefs in some circles, a 4-bit, ripple, binary up counter cannot be converted into the equivalent down counter merely by interchanging power and ground nor by mounting the chips on the opposite side of the PCB. Rather, such an attempt is more likely to release the smoke demon – not a pleasant prospect to say the least.

Observations and Curious Questions:

If the water leaving a flushed toilet circulates one direction north of the equator and the opposite direction to the south, what direction does it go at the equator? At what point does it change directions? What if half of the toilet is to the north and the other half to the south…then what happens? Does the same thing happen to a ballet dancer? How about a Tasmanian devil?

If you hide in the closet with your cell phone, how does a telephone call know exactly where you are when no one else does?

Project:

General Description

The prototype system will implement a portion of a highly reliable remotely controlled system intended for use in critical surgery operations. Ultimately, the remote portion of the system will be triply redundant and include a separate monitoring channel. For the feasibility prototype and current studies, the system will comprise a local node, a remote node, and single instance of the control channel on the remote node. The communication link between the local and remote nodes will bundle the outgoing control channel with the return monitor channel from the remote node.

The local node will provide the interface through which the user will remotely control the operation of the surgical device. Together, the local and remote nodes will form a distributed closed loop control system. Commands will be accepted from the user, interpreted and formatted, then sent over the network to the remote node. On the remote node, the control channel will accept, interpret, and execute the commands. The monitoring channel will track the operation of the end effecter control system, return the state of the end effecter to the local node as an error signal, and take the appropriate action(s) under out of tolerance or failure conditions.

The project is to be developed in two phases. The first phase deliverables will comprise the high-speed control (outgoing) channel, implemented as a portion of a local area I^2C network, and a local EIA-232 terminal user interface that will provide a text based display of commands, responses, and system state. The second phase will support the remote monitor channel (return channel) and the data for the local control, annunciations, warnings, and alarms.

High reliability is the key objective.

General Requirements

You are building a feasibility prototype for a portion of a highly reliable, remotely controlled system intended for use in critical surgical operations. The prototype for the remote portion of the system will implement a single control channel and a return monitor channel.

Local Node

The local node will provide the user interface, the control algorithm, warning and alarm driver, and the local portion of the network.

From the local node, the user will be able issue commands to start and stop the motor controlling the surgical tools on the remote node. In addition, the user will be able to specify a set point and to increment or decrement the speed of the motor in steps of ±0.5%. Such action will set the reference motor stimulus value on the remote node.

Based upon the data returned via the remote monitor channel, the control algorithm will be able to command an increase or decrease in the speed of the motor around the

set point in steps of ±0.5%. The error signal will be algebraically added to the reference motor stimulus value to maintain the commanded speed.

The local node will support an EIA-232 terminal interface to enable the display of the set point, increment, and decrement values as well as any error or alarm information from the remote node.

Remote Node

The remote node will comprise the control and monitoring channels and support for the remote LAN portion of the network.

Control Channel

The control channel will accept set point, increment, and decrement commands from the local node. The control channel will use the set point information to control the pulse width of a PWM output signal that is used to drive a small electric motor. Increment and decrement commands will increase or decrease the pulse width, and thereby the speed of the motor, accordingly.

The PWM output will have a frequency of 20.00 KHz. The control voltage for the motor ranges between 0.0, corresponding to motor OFF, and 5.0 V DC, corresponding to motor FULL ON.

Monitor Channel

The monitor channel will measure the average voltage applied to the motor and return this value to the local node. In addition, based upon the control commands, it will compute the expected average voltage that should be applied to the motor, measure the actual voltage, and compute any error. If there is an error, the appropriate actions should be taken.

Errors are organized into three classes,

- ±5.0% - Level 0 - severe
- ±2.0% - Level 1 - moderate
- ±1.0% - Level 2 – of concern

The Local Area Network

The LAN will be implemented as an I^2C link and will provide the communication path between the local and remote nodes. The link will comprise a bidirectional connection between the local node and the control and monitor channels on the remote side

Communication will be buffered at both the local and remote nodes.

The LAN between the local and remote nodes can be implemented using the built-in I^2C interface on the PIC on both sides of the link

Deliverables

The following are the deliverables for the project,

All Phases

Weekly status report by each team member emailed to the instructor and TAs by Friday afternoon describing his or her efforts and contributions on the project for the previous week. This is to be an individual, not team, report.

High Level Phase 1 Deliverables

- The required documentation,
- A working prototype of the local node,
- A working prototype of the control portion of the remote node,
- A working prototype of the control channel link on the network.

High Level Phase 2 Deliverables

- The full documentation,
- A working prototype of the local node,
- A working prototype of the control and monitor portions of the remote node,
- A working prototype of the control and monitor channel links on the network.

Phase 1

1. The following are due on the date listed on the class web page, prior to the first design review:

   For both the requirements and design specifications, see the EE 472 or on-line text for examples, the expected format, and general content.

   a. A Requirements Specification, a Design Specification, Test Plan, and a preliminary bill of materials reflecting the specifications for the local node and the control portion of the remote node.

   b. A detailed block diagram for the local node and the control node portion of the remote node.

   c. A Failure Modes Analysis based upon the current block diagram.

   d. A preliminary functional decomposition for the software on the local node and the control channel and comms link on the remote node.

   e. A full schedule, presented as a Gant Chart, specifying the major tasks and milestones on the project and the primary person responsible for each task. This is due on the date listed on the class web page.

2. Project design review on the date listed on the class web page. Documentation for the design review should include:
    a. An updated Requirements Specification, a Design Specification, Test Plan, and bill of materials.
    b. An updated schedule reflecting the current state of the project.
    c. An updated block diagram for the local node and the control node portion of the remote node.
    d. A project design review. Be prepared to justify your design decisions and design.
3. Project demo on the date listed on the class web page. The first phase demo shall show the local node functionality and transmission to and control at the remote node.
    a. An updated Requirements Specification, a Design Specification, Test Plan, and bill of materials reflecting the specifications for the monitor portion of the local and remote nodes.
    b. A set of test cases based upon the Test Plan for the current system.
    c. An updated schedule reflecting the current state of the project.
    d. An updated detailed block diagram for the local node and the control and monitor portions of the remote node.
    e. An updated Failure Modes Analysis based upon the full block diagram.
    f. A demo of the Phase 1 deliverables.

Phase 2

4. Project design review. Note that this occurs at the time of the Phase 1 demo. Documentation for the design review should include:
    - Block diagram for the bidirectional local to remote nodes portion of the system.
    - Protocol for the exchange between the local and remote nodes.
    - Updated cost estimate and schedule,
    - Pseudo code for the return channel algorithm,
    - Preliminary timing diagrams.
5. Project demo of the complete working system. The demo is on the date listed on the class web page.
6. One project report. The report is due on the date listed on the class web page.

7. Additional features will be individually evaluated but they only apply if the main portion of the system is fully functional.

Project Report:

Write up your project report following the guideline on the EE 478 web page.

Your final deliverables in your report for this project include,

1. Completed and updated Requirements and Design Specifications.
2. Final detailed system block diagram.
3. System timing diagram as appropriate.
4. System state diagram(s).
5. Logic equations or Verilog listings.
6. Final Failure Modes Analysis.
7. Failure management scheme.
8. Software for your system.
9. Schematic / logic diagrams for your remote surgery system.
10. Logic analyzer printouts and accompanying timing analysis as appropriate.
11. Test Plan.
12. Analysis and discussion of problems encountered in the design and implementation of the system as appropriate.
13. Short technical description of the system.
14. Final factory cost (BOM) for your system.
15. Final updated schedule.
16. Demo to your TA or instructor of a working system.

Appendix

Background Information:

Microcomputer communications is a rapidly growing field with an ever-increasing number of applications, ranging from local PC networks to large-scale communication systems. Central to any communications between electronic devices is a protocol for transmitting and receiving information. Within a computer, data is usually transferred in parallel form, such as on a microprocessor or an I/O bus. While parallel communication is far more efficient than serial at moving large numbers of bits, it is not always as practical. Thus, most communications a computer and other electronic devices that are not in the immediate vicinity is usually done using a serial scheme – Ethernet, USB, WiFi, Bluetooth, Firewire, EIA-232, SPI, I^2C etc….

Of course, with two different formats for exchanging data there will be many occasions in which data have to be converted from one form to the other. There are a number of ICs available to accomplish this task.

Another important element of communication is ensuring that the data given to the user following reception contains no errors arising from such a transmission. Note that we do not guarantee there are no transmission errors, these happen. Rather, at the end of the day, we guarantee the data to be correct. There are a variety of schemes by which this is accomplished: all begin with recognizing that a transmission error has occurred. We'll examine one, simple parity checking.

When debugging digital circuits, it is helpful to understand what each part of the circuit is doing and exactly when certain events are occurring. The amount of information can be huge, and often we would like to filter out as much of the unnecessary data as possible. Also, sometimes we are interested in timing measurements, other times we are more concerned with comparing the different states of various parts of the circuit. Use the logic analyzer to do this.

Serial Communication: Asynchronous vs. Synchronous

Asynchronous communication suggests that there are irregular intervals between the sending of data. Suppose that a serial communication line is set up to transmit ASCII characters as typed by a person at a keyboard. The spacing between the transmission of each character will vary widely, and there may be long periods when no characters are typed (coffee breaks, tea breaks, bio break, naps, etc.). In this situation, the receiving device needs to be told when a character is being sent to prepare it to receive that character and sort out which part is data, which part is the error-checking field, and so on.

This is accomplished by a procedure known as ***framing***, in which a start bit is placed before the first data character, and a stop bit is placed at the end of the transmission of one character. The start bit enables the receiving device to temporarily synchronize with the transmitting device, while the stop bits allow the receiving device time to get ready for the next frame (see Intel supplement).

In contrast, when blocks (usually large ones) of regularly spaced data are transferred over a serial line, the transmitter and receiver can be synchronized, transferring characters at a much higher rate. In this format, known as synchronous transmission, start and stop bits are no

longer needed, since the receiving device knows exactly where each character begins and ends.

Although synchronous transfer requires less overhead and therefore is much more efficient, its uses are more limited than asynchronous data transfer, and thus the latter is more widely used.

Network Overview

Our serial network is designed to exchange information between an operator at a local site and a surgical tool at a remote site. The network implements half-duplex communication, that is, data may be sent in only one direction at a time.

Traditionally, networks comprise a number of layers - each viewed as a virtual machine upon which the layer above operates. OSI uses 7, TCP/IP uses 5, USB uses 3. For our design, we will use 4: the *physical laye*r, the *data link* layer, the *protocol layer*, and the *application layer*. These are described in the following.

The Physical Layer

The physical layer for our network will comprise 5 lines and their associated signals: transmitted data, received data, signal ground, and two control lines. Signaling levels will be those specified for EIA-232 as described above.

The Data Link Layer

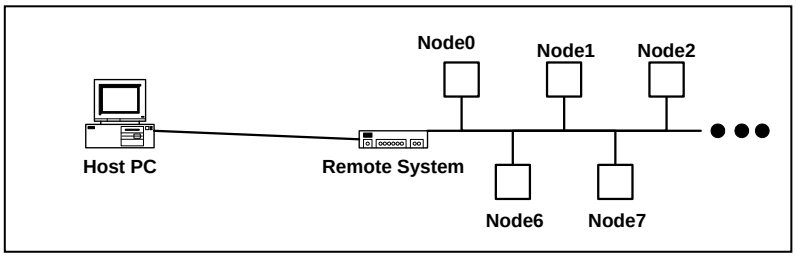
The data link layer will move characters, expressed in the EIA-232 format, from the host to a remote site and from a remote site to the host.

The Protocol Layer

The Protocol layer will move commands and data from the host to a node at the remote site and move data from a node at the remote site to the host. Each node within such system is identified by a 3 bit address.

The Application Layer

The Application layer will provide the link between an application on the host and one on a remote node. We wish to be able to communicate with and execute applications on up to 7 systems on our network as shown in the example block diagram below. For this project it is sufficient to demonstrate one.

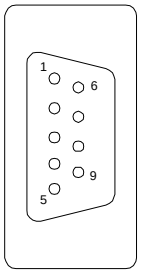


I²C Interface and Protocol

The I²C interface and protocol are described and discussed in the PIC 18F25K22 data sheet. See Section 15.0, Master Synchronous Serial port (MSSP1 and MSSP2) module. Additional information can be found online or in the EE472 text.

EIA-232 Protocol

When two devices are relaying information back and forth, it is common to designate one of the devices as Data Terminal Equipment (DTE) and the other as Data Communication Equipment (DCE). The EIA-232 protocol is a standard that specifies the circuits between the DTE and DCE devices, and the lines that connect them, such as which line data is transmitted over, which line controls handshaking, and so forth. The figure below shows the pin numbers on a DB-9 and a DB-25 connector and the EIA-232 inputs/outputs to which they correspond.



1 Carrier Detect

2 Received Data

3 Transmitted Data

4 Data Terminal Ready

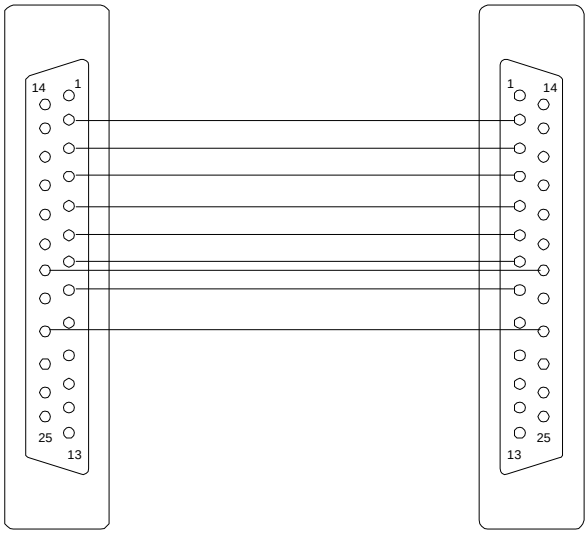
5 Signal Ground

6 Data Set Ready

7 Request to Send

8 Clear to Send

9 Ring Indicator



DTE
Data Terminal Equipment

DCE
Data Communication Equipment

2 TXD Transmitted Data

3 RXD Received Data

4 RTS Request to Send

5 CTS Clear to Send

6 DSR Data Set Ready

7 SG Signal Ground

8 CD Carrier Detect

20 DTR Data Terminal Ready

22 RI Ring Indicator

For this project, a PC will function as the DTE, and our remote system as the DCE. We will be working with only the following EIA-232 lines:

TXD--data transmission line from DTE to DCE

RXD--data transmission line from DTE to DCE

DSR--data set ready primary handshake, from DCE to DTE

CTS--clear to send secondary handshake, from DCE to DTE

SG--signal ground

Handshaking, also known as flow control, is a way for the DCE to co-ordinate actions or the exchange of data with the DTE. For example, if data is sent to a printer at a rate higher than that of the speed of printing, the printer will send a signal to the sending device to stop until it catches up. In the EIA-232 specification, signals such as CTS or DSR are sometimes (mis)used to serve that purpose.

A high level on the DSR pin tells the transmitting device that the communications device (typically the modem) is ready to send data, whereas a low level tells it to stop. The line is asserted following power up (self-tests or other initialization) and should remain in that state.

The signal CTS - Clear to Send - is intended to indicate to the transmitting device that it is now OK to send data - that the physical line is intact, that a carrier has been detected, etc. Often, this signal is incorrectly used for flow control. Why is this a problem…Why will data be lost using such a scheme?

Special voltage levels that correspond to digital high and low characterize EIA-232 data. A logical '1' in the EIA-232 specification is known as a *mark*, and corresponds to a voltage between -3 and -15 volts, whereas a logical '0', or *space*, corresponds to a voltage between +3 and +15 volts. These special voltage levels are used to preserve the signal as it travels through connecting cables.

Notation	**Interchange Voltage**	
	Negative	**Positive**
Binary State	1	0
Signal Condition	Marking	Spacing
Function	OFF	ON

You will need to add a buffer to the input of your device to convert from EIA-232 to TTL levels and vice versa.

Finally, note that the data link remains in the marking state (< -3 V) until the start bit, which is a space (> +3 V), is sent.

Framing

In serial communications, data is grouped into frames, which have already been partially described above. The start of a frame is signaled by the start bit, the end by one or more stop bits. Data and error checking (parity bit) codes are contained within each frame.

In order to perform serial to parallel conversion, our circuit must know what part of the frame is being looked at any one time. For example, we don't want to be looking for a start bit in the middle of the data, nor do we want to be checking parity when the start bit is being received. One way to keep track of the serial input is to use a counting mechanism activated by the start bit of each frame.

Please note that basic ASCII data is based on a 7 bit code (0x00 through 0x7F), so when HyperTerminal is set for 8 bit data, it will add in an extra bit to ASCII values. Normally it will stuff a '0' in the MSB position unless it has a stream of identical characters. In this case, on the third occurrence, it will begin to toggle the MSB. For example, when transmitting a series of D's, it will send out, in order:

0101 0010

0101 0010

1101 0010

0101 0010

1101 0010

etc…

This is not something that we have to worry about for this project.