DIGITAL TRANSMISSION OF AN INITIAL FIRE REPORT OVER A VHF RADIO NETWORK

Roy Belak
Dylan Gunn

Project Sponsor: Mike Winder, Technical Director, BCFS

Engineering Physics 459
University of British Columbia
Project No. 0408
March 31, 2004

Executive Summary

Congestion of internal radio channels plagues the British Columbia Forest Service (BCFS) during fire season. High broadcast volumes arise as field personnel vocally relay fire statistics and field reports to regional fire centres over the radio network. One of the most commonly transmitted reports, the Initial Fire Report (IFR), is responsible for a significant portion of this radio traffic.

A prototype system was developed to demonstrate the advantages of transmitting IFRs digitally. The system hardware consists of independent Transmission and Reception Modules that connect to the BCFS radio network via the external port of a standard issue ICOM-F3 radio. The transmission module encodes IFR information entered by field personnel as a voice-band frequency shifted signal which can be broadcast over the BCFS radio network. A reception module located in a regional fire centre decodes the received signal and sends it to the serial port of a PC. Software on the PC processes the received IFR for errors and displays the information on the screen.

Testing of the device showed an average IFR transmission time of 2.7 seconds, which is a substantial improvement over the one minute required for vocal transmission. This suggests that the system can provide a significant reduction in radio traffic. In addition, the accuracy of the transmissions exceeded 95% and with further refinement to transmission protocols, both transmission time and accuracy can be improved considerably.

It is therefore recommended that the BCFS employ the Digital IFR Transmission system to help reduce radio congestion and consider widespread application of the technology.

Table of Contents

List of Figures

1.0 Introduction

During the peak fire season, from July to September, the British Columbia Forest Service (BCFS) radio network becomes clogged with fire suppression related transmissions. This has proven to be a detriment to the safety and overall performance of the organization. As the radio system becomes increasingly congested, the vital communication link between forestry personnel is compromised.

Initial Fire Reports (IFRs) are one of the most common field reports relayed over the network. The Digital IFR Transmission system was developed to improve both the transmission time and accuracy of IFR data being sent from field personnel to regional dispatch centres. The traditional method of relaying an IFR by voice over a radio channel takes approximately one minute and is prone to miscommunication and errors. During busy fire seasons, the radio time being occupied by IFR transmissions often results in substantial periods during which the radio network is inaccessible to other personnel.

Initial investigation by the authors along with information provided by the sponsor indicated that the Very High Frequency (VHF) radio network already used by the BCFS provided a sufficient communication pathway over which digitally encoded information could be transmitted. Based on this investigation, it would be possible to interface a mobile, field-ready device with the existing network to broadcast IFR data in a fraction of the time traditionally required.

This report was written to document the development of the Digital Initial Fire Report (IFR) Transmission system developed for the Engineering Physics 459 Project Lab from

January to April of 2004. The report summarizes the work completed at the time of writing, including a description of the deliverables produced, major problems encountered during development, changes to the original project plan, and recommendations concerning the continuation of the project. An outline of the report structure with a brief description of the major sections is given below.

- **Discussion:** Background information and theory pertaining to the project are followed by an analysis of the project work completed. A detailed description of the solution obtained is compared to the original objectives of the project, and a thorough analysis of the results obtained is provided.

- **Conclusions:** The conclusions section summarizes the results and elaborates on the importance of the results achieved.

- **Recommendations:** Based on information given in the discussion and conclusions sections, recommendations on the future continuation of the project are given in this section.

- **Appendices:** The appendices contain supplementary information on the project. They include data and information sheets on the hardware used in the construction of the device, device schematics, results of tests completed, source code for software written and a user's manual for the device.

2.0 Discussion

2.1 Background

As chief steward for the crown land in BC, the BCFS is responsible for managing wildfires in wooded areas. As fire season heats up, an increasing number of fire suppression related voice transmissions are made on the BCFS radio network. As transmissions become more frequent, it becomes increasingly difficult for Forest Service employees to obtain enough radio airtime to perform their job. Radio congestion has also been identified as a significant safety hazard should personnel need immediate contact with dispatchers at regional fire centres. Any measure that might alleviate radio traffic is being considered seriously by the BCFS.

In summer months, an Initial Fire Report (IFR) is one of the most commonly transferred field reports relayed over the network. The IFR provides critical fire details such as fire location, elevation, fuel type, and fire activity. Currently, broadcasting an IFR by voice takes approximately 1 minute. With an average of more than 2500 fires annually in BC, this amounts to almost 42 hours of radio time during which other personnel cannot use the radio network. This problem is exacerbated during fire “busts”, when many fires are discovered simultaneously. The transmission of these IFRs virtually clogs the radio network, making all other communication between personnel difficult.

The BCFS radio network uses half-duplex VHF Radio Frequency (RF) technology. Field units (personnel) are given handheld or truck mounted radios with pre-programmed frequencies. Communication with fire centres is coordinated through an extensive repeater network established around the province. Messages sent from field units are relayed via the repeater network to a destination repeater that is connected to a fire centre through a traditional telephone cable.

In order to reduce the network congestion during fire season, the BCFS has many options at its disposal. Current innovations in frequency multiplexing and other technologies are providing means to increase the bandwidth of data transmission networks. Unfortunately, most of these solutions require considerable investment in upgrading the existing infrastructure to make it compatible with new hardware and software. Although these upgrades will likely be inevitable in order for the BCFS to maintain the operability of the network, these solutions take time to implement properly. A short-term solution for dealing with increased radio volume is needed. A simple interim resolution involves digitizing IFR data and sending it over the radio network.

2.2 Theory

As with most RF networks, the bandwidth available for communication is restricted to a band just large enough to allow intelligible transmissions. In the case of the BCFS network, this bandwidth is confined to 2.5 KHz. This narrow bandwidth limits the number of available options for implementing a digital over-link. The entire frequency band is located in the audible range, meaning that any digital data encoding will require the transmission of tones. Similar RF telemetry schemes have been implemented for civilian and military purposes in the past. Many of these schemes perform automatic data logging and transmission, some common examples include products for collecting tidal and geological data used by meteorologists.

The fundamentals behind the Digital IFR Transmission system revolve around encoding IFR data fields into digital format and transmitting the information over the network. To use the radio channel as an effective medium, the digital information needs to be encoded into a variable frequency signal.

Previous experimentation conducted by the BCFS has found that data transmission rates on the order of 1200 – 2400 baud are feasible on the existing system (Barry Cowan, Lead Radio Hand, telephone discussion, September 2003). Frequency Shift Keying (FSK) was identified as the most suitable method for converting digital information to analog signals given the relatively small bandwidth. FSK modulation is based on using two analog frequencies to represent binary states. Other digitizing schemes such as Dual Tone Multiple Frequency (DTMF) use a larger number of different tones, and can therefore transmit more information per unit of transmitted data. Adding more tones compresses the frequency bands available to each state and therefore results in increased decoding errors. FSK was selected over these more efficient protocols in order to maintain data integrity. Figure 1 below illustrates the principles behind FSK modulation.

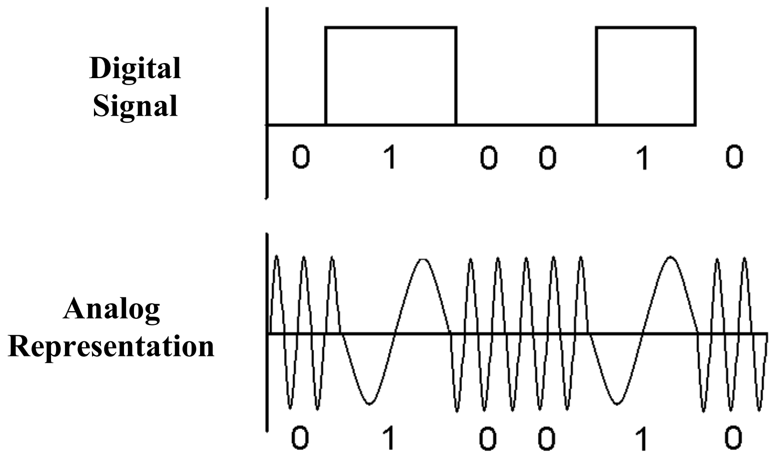


Figure 1 - FSK Modulation

2.3 Statement of the Problem

Excessive radio network traffic has deleterious effects on the efficiency and safety of the BCFS personnel. Frequently occurring transmissions, such as IFRs, are responsible for much of this congestion. Apart from direct consumption of radio time, these transmissions also require dispatchers to transcribe the data being transmitted. Eliminating these requirements would free these resources for other uses.

A more efficient method for getting the initial fire information from the point of fire discovery to central dispatch was needed. More specifically, a device to digitize and relay IFR data over the network as a set of audible tones was desired. As a criterion for proper operation, this device must be compatible with the current BCFS radio network. This compatibility would allow any solution to provide easy integration with existing resources, and therefore, reduce the time needed for full implementation. Furthermore, it is desirable that financial obligations involved with the implementation of any data transmission solution be light.

2.4 The Solution Chosen for the Problem

The solution chosen was designed to satisfy the project objectives given the resources and time available. This required independent Transmission and Reception Modules that could interface with the current BCFS radio network by plugging into the external microphone/speaker ports of standard field radios, and software / protocol that would allow data to be transmitted quickly and efficiently with a low rate of error and strong error detection capabilities. The following discussion is broken into sections corresponding to the components of the system delivered. The first section deals with the hardware and software developed for the Transmission

Module, the second details the solution for the Reception Module, and the last section provides a description of the transmission protocol developed.

2.4.1 IFR Transmission Module

The Transmission Module acts as the data entry interface between personnel in the field and the regional fire centre. Once the user inputs the required data fields, the module encodes the IFR information for transmission over the BCFS radio network via a handheld radio. Figure 2 below shows the current configuration of the Transmission Module.

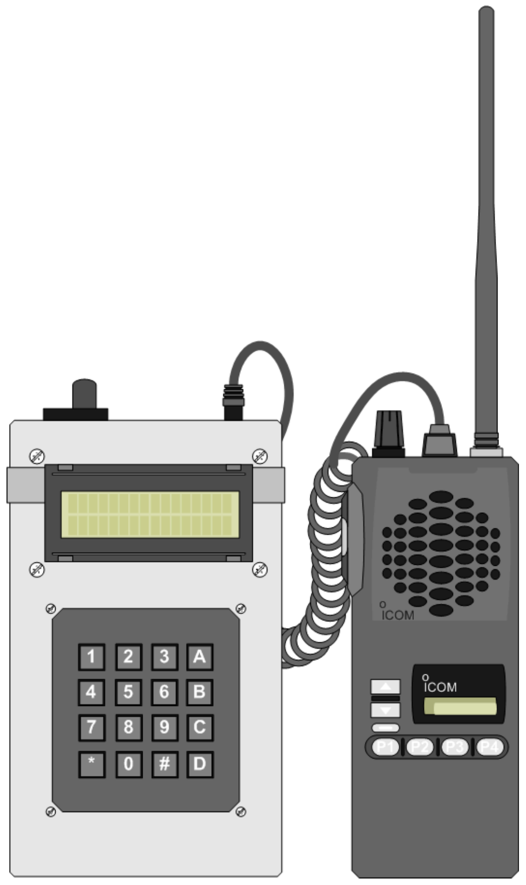


Figure 2 - Transmission Module

2.4.1.1 IFR Transmission Module Hardware

TM3105 Modem: The fundamental component of the transmission hardware is the Texas Instruments TCM3105 1200 Baud Frequency Shift Keying (FSK) modem. The modem converts the digital signal representing encoded IFR information to an analog signal with frequencies in the voice-band range. This signal is then sent to the transmitting radio through the external microphone port. This particular modem was chosen over newer, more powerful modem ICs for several reasons. Modern modem ICs designed for cellular phone applications are equipped with several sophisticated features such as call display and caller ID that were not needed for this project. Furthermore, these modems are designed to transmit at speeds above 9600 baud, which is well beyond the bandwidth available on the radio system.

Although the TCM3105 is a very popular modem among hobbyists, it is no longer manufactured and is only available in small quantities from specialty distributors at relatively high prices. The project proposal originally specified that the Bell202 standard would be used to modulate the digital signal, producing a mark (logic 1) frequency of 1200Hz and a space (logic 0) frequency of 2200Hz. However, once the modem was set up for testing, it was determined that use of the Bell202 standard would require complex timing circuitry to configure the modem. The CCITT V.23 standard, with mark and space frequencies at 1300Hz and 2100Hz respectively, was much easier to implement and therefore chosen to replace the Bell202. The analog output of the modem was coupled to a voltage divider to reduce its amplitude from approximately 3V to 200mV peak-to-peak.

PIC Microcontroller: The Microchip PIC18F452 microcontroller was chosen as the control computer for the Transmission Module as it is equipped with more than sufficient processing power and RAM to handle the simultaneous operation of various parts of the module. In addition, the 18F452 has an embedded Addressable Universal Asynchronous-Synchronous Receiver Transmitter (AUSART) serial communication port that can be configured to output serial data at a variety of baud rates. In order to operate the AUSART at 1200 baud, it was necessary to use a 4MHz oscillator to time the chip, which resulted in an overall processor speed of 1MHz. The amount of RAM available on the 18F452 exceeded the amount required to store IFR information entered by the user. Although the capabilities of the PIC far exceeded the requirements of the project, it was deemed safe to err on the side of caution for the development stage of the project.

COTO-0325 Reed Relay: The external microphone input to which the IFR Transmission Module is connected is sensitive to very small signals, on the order of 5-10mV. Almost any signal input can cause the radio to begin transmitting at the incorrect time. A Coto reed relay controlled by the PIC was used to produce a physical break in the signal line from the transmitting modem to the microphone input, mimicking the configuration of the external speaker/microphones provided with the radios.

Keypad: A 16-digit alphanumerical keypad manufactured by Grayhill (#96BB2-056-R) was added to the Transmission Module to allow the user to enter IFR information. The keypad came without external ICs to handle polling/reading operations and these had to be programmed in the PIC control software.

Liquid Crystal Display: In order for the user to view IFR information as it is entered, a 16X2-line LCD display manufactured by Optrex (DMC-16207) was interfaced with the Transmission Module. The LCD data and power lines were wired straight to port B of the PIC. A library of functions for controlling the LCD was provided with the MPLAB software, and little work was required to make the LCD operational.

Initial tests of the transmission circuitry (PIC and modem) proved successful, but once the LCD was connected to the PIC, data transmission was no longer reliable. A quick analysis of the circuit revealed that the speed at which the PIC was sending data to the modem (1200 baud) decreased substantially when the LCD was connected to the PIC. The first attempted solution involved powering the LCD from a separate connection to the power rail, rather than sharing a power connection with the PIC. This failed to work and required a different approach. The next attempt was to shut off the LCD during transmission of the IFR. In order to achieve this, the power line to the LCD was wired through several reed and transistor switch configurations that were controlled by the PIC. However, none of these were able to shut off the PIC during transmission of the signal, and further analysis showed that the LCD was receiving power from the PIC through the data lines of port B. The PIC was then programmed to ground all ports connected to the LCD during transmission. This solved the problem.

Power Supply: A 9V battery used to power the transmission module was stepped down to 5V with a Fairchild MC7805 voltage regulator. Capacitors were placed across the terminals of the regulator in order to stabilize the signal.

2.4.1.2 IFR Transmission Module Software

Software for the Transmission Module was coded in C to allow easy integration with the MPLab Desktop Environment. A demo compiler available from Microchip, MCC 18, was used to convert the C-code source files to the assembly level instructions needed by the assembler. The MCC 18 compiler is a standard ANSI C compiler that comes with a diverse library of functions. These libraries were used extensively for port addressing, LCD module control, and variable manipulation.

The source code is broken into different files to increase the modularity of the program. These files are detailed in Appendix F. A flow diagram showing the flow of control for this software is shown on the following page.

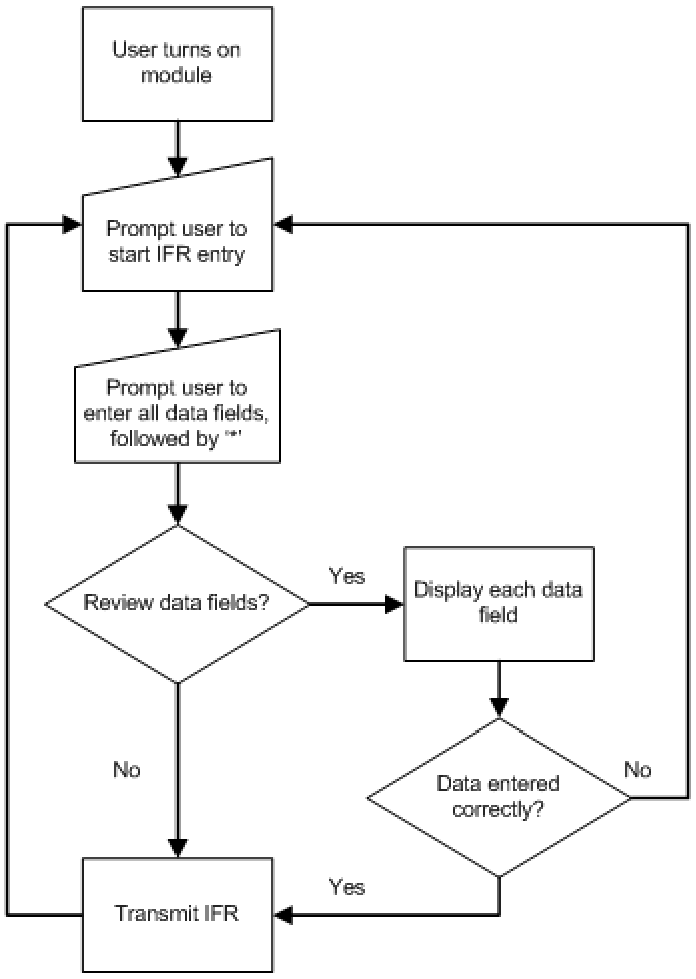


Figure 3 - Transmission Software FlowDiagram

2.4.2 IFR Reception Module

The Reception Module is used to demodulate the signal from the external speaker port of the receiving radio and reconstruct the original data. The Reception Module also converts the TTL logic levels to RS232 for input to the COM port of a standard PC. The delivered module is meant as a temporary solution for preliminary testing with handheld radios and in a full implementation the receiving radio equipment would already be interfaced to a computer. Figure 4 is a diagram of the Reception Module.

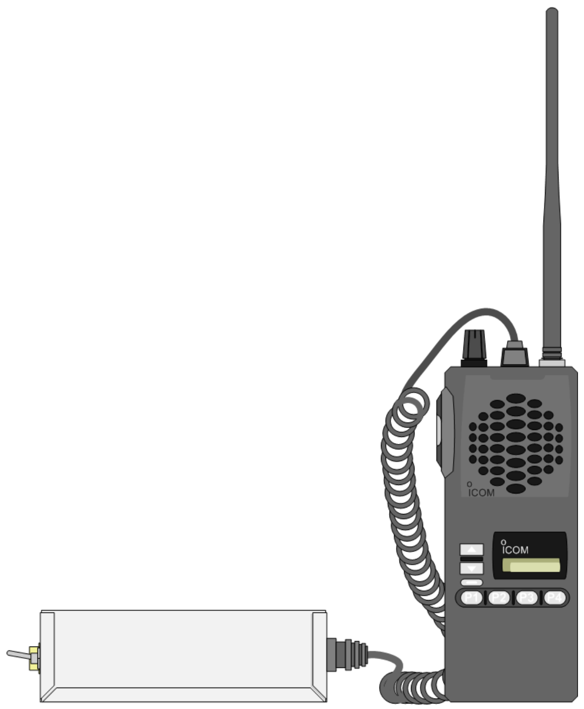


Figure 4 - Signal Reception Module

2.4.2.1 IFR Reception Module Hardware

TCM3105 Modem: The Texas Instruments TCM3105 modem was used in the Reception Module for the same reasons previously mentioned. The external speaker line from the receiving radio was capacitively coupled to the analog input of the modem to remove any DC component from the signal.

Maxim MAX232 Level Converter: The digital output of the receiving modem was wired to the input of a MAX232 level converter to convert the TTL logic levels from the modem to RS232 levels understood by a PC. The output of the MAX232 converter was wired to a DB9 serial connector to allow communication with the COM port of a standard PC.

2.4.2.2 IFR Reception Module Software

LabVIEW 7.0 was chosen as the development platform for the Reception Module software as it contains several built-in functions for controlling the serial ports on a computer. In addition, it is possible to build stand-alone LabVIEW executables which can be run on any computer that has the LabVIEW 7.0 runtime engine installed. The LabVIEW runtime engine can be downloaded free of charge from the National Instruments website (www.ni.com), and there are no restrictions on distribution. Initial plans to develop an executable using C++ or Visual Basic were quickly discarded when it was determined that a custom driver would have to be written to gain access to the serial ports on versions of Microsoft Windows newer than Windows 98.

The front panel of the reception software, shown in Fig 6, indicates the operating status of the program as it receives and processes IFRs. Once an IFR has been fully received and processed, the program displays the information on the screen and prompts the operator to save the IFR in a user-defined file. A debugging version of the program that displays additional data on the program flow was also produced, and a screen shot of its front panel can be found in Appendix G. Figure 5 on the following page illustrates the software flowchart for the Reception Module.

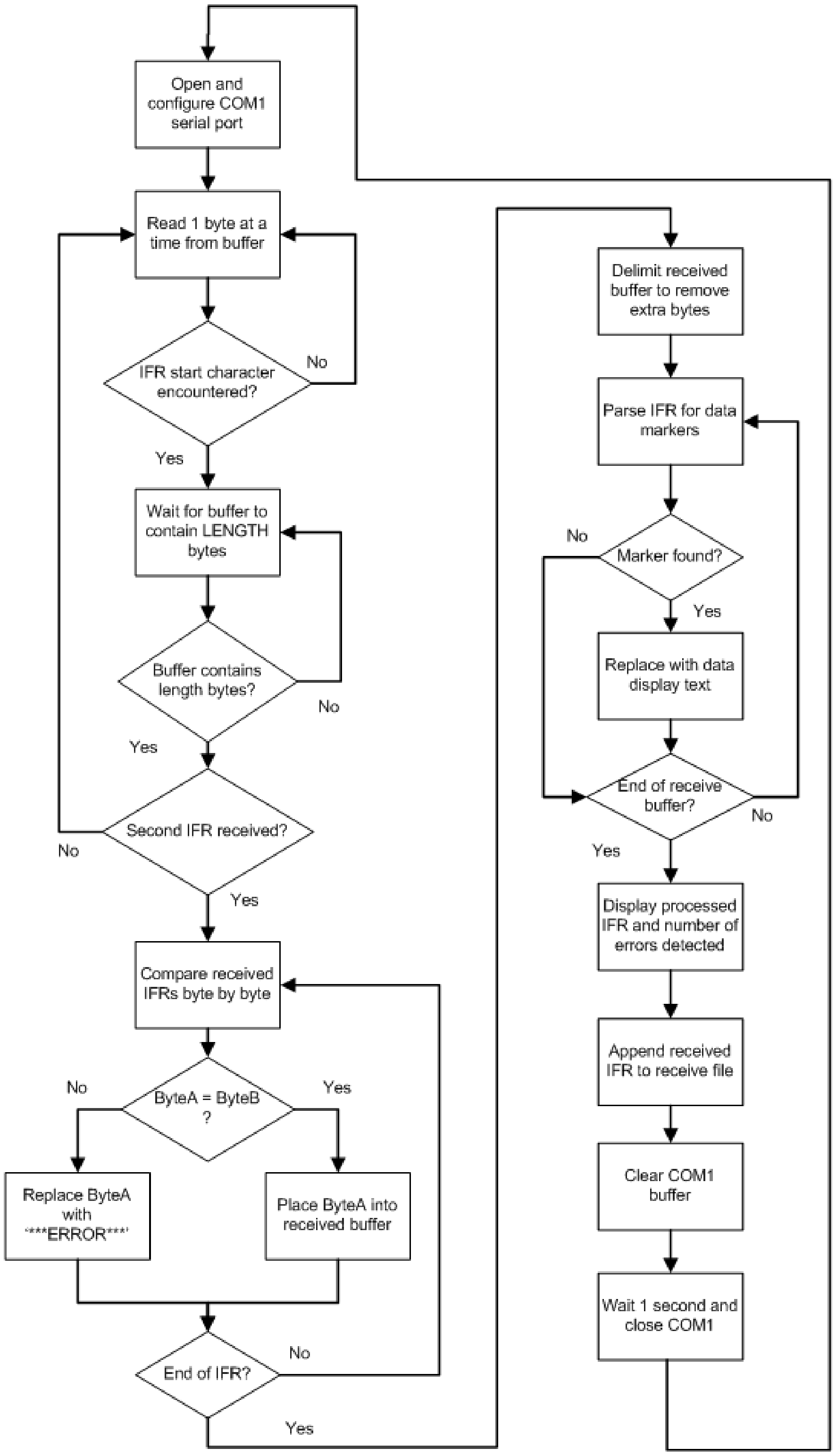
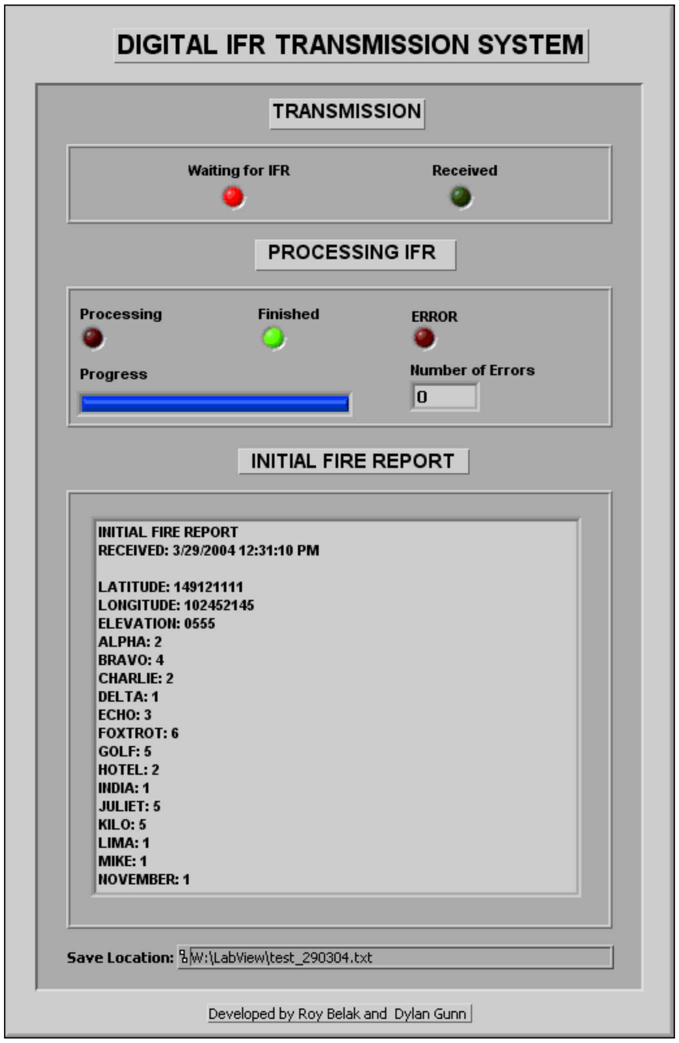

Open and configure COM1 serial port
Read 1 byte at a time from buffer
IFR start character encountered?
No
Yes
Wait for buffer to contain LENGTH bytes
Buffer contains length bytes?
No
Yes
Second IFR received?
No
Yes
Compare received IFRs byte by byte
ByteA = ByteB ?
No
Yes
Replace ByteA with '***ERROR***'
Place ByteA into received buffer
End of IFR?
No
Yes
Delimit received buffer to remove extra bytes
Parse IFR for data markers
Marker found?
No
Yes
Replace with data display text
End of receive buffer?
No
Yes
Display processed IFR and number of errors detected
Append received IFR to receive file
Clear COM1 buffer
Wait 1 second and close COM1

Figure 5 - Reception Module Software Flowchart



```
0
```

```
INITIAL FIRE REPORT
RECEIVED: 3/29/2004 12:31:10 PM

LATITUDE: 149121111
LONGITUDE: 102452145
ELEVATION: 0555
ALPHA: 2
BRAVO: 4
CHARLIE: 2
DELTA: 1
ECHO: 3
FOXTROT: 6
GOLF: 5
HOTEL: 2
INDIA: 1
JULIET: 5
KILO: 5
LIMA: 1
MIKE: 1
NOVEMBER: 1
```

Save Location: W:\LabView\test_290304.txt

Figure 6 - Reception Module Screen Shot

The reception software source code is a 15-frame stacked sequence structure. Each frame is executed in sequential order and program flow moves from one frame to the next according to specific conditions. The process is repeated after the last frame in the sequence has been executed. The program continually monitors the COM1 serial port read buffer and waits for an IFR transmission to appear. Once an IFR appears, the program scans the transmission character by character, looking for errors and replacing them with an error flag. The received IFR information is then decoded into a more convenient format and displayed on the screen. The user is then prompted to save the information to a file of his choice. A full printout of the 15 frames of the block diagram is given in Appendix G.

2.4.3 IFR Transmission Protocol

Seventeen distinct pieces of information corresponding to latitude, longitude, elevation, and field codes Alpha through November are transmitted in a single IFR. Nine numbers are required for longitude and latitude each, four numbers are required for elevation in meters, and one number each is required for the remaining field codes. This corresponds to a total of 36 characters of information that need to be transmitted for each IFR.

Speed and accuracy were the primary objectives for designing the communication protocol. The inability of LabVIEW to perform bitwise manipulation of data eliminated using parity check bits, polynomial cyclic redundancy checking, and Reed-Solomon encoding for error detection. It was decided to forego the correction of errors in a transmitted IFR and focus on their detection while keeping transmission time to a minimum.

The design of the transmission protocol then focused on arranging the IFR data in a way that could be sorted by LabVIEW. LabVIEW's ability to break up a string of data between user-defined delimiters meant that by adding a start delimiter at the front of the IFR string and appending another delimiter to the end of the string, data representing the IFR could be sorted from other non-IFR characters. LabVIEW also contains a pre-defined function for parsing a string and replacing selected characters with a replacement string. Marker characters were added between the individual segments of IFR data so that LabVIEW could separate the IFR information and replace the markers with display information corresponding to each piece of data. By programming LabVIEW to replace the letter ‘s’ with the string ‘ELEVATION: ‘, the following encoded IFR string

```
XXXXXXXs1200XXXXXX
```

(where ‘X’ represents arbitrary characters) would be replaced by

```
XXXXXXXELEVATION: 1200XXXXXX.
```

By adding line feed characters to the replacement strings, and repeating this operation for every piece of IFR data, a continuous string of IFR information could be parsed into a vertical list displaying the IFR data in an easy to read format. For example, the string below

```
XXXXXq104667002r049078997s2500XXXXX
```

could be processed to produce the following portion of a report:

```
Latitude: 104667002
Longitude: 049078997
Elevation: 2500
```

Without the ability to perform bitwise error correction it was decided that the most effective solution was to simply transmit the IFR information twice and then have LabVIEW compare the received strings character-by-character to determine any discrepancies. Although this solution would not provide any means of correcting errors, the possibility of an error being received undetected was extremely low, and the exact location of any errors within the received IFR string could be easily determined.

The last consideration in the protocol development was the timing needed to open the radio channel. It became apparent that the delay between attempting to open the channel, and the channel opening varied by several hundred milliseconds. Without accounting for this variable delay, the first portion of an IFR was often truncated. To compensate, several 'X' characters were added as padding at the start of the IFR transmission and the final form of the transmitted string is given below:

```
XXXXXXXXXXpq123456789r987654321s2000A1B2C3D4E5F
6G7H8I9J0K1L2M3N4opq123456789r987654321s2000A1B
2C3D4E5F6G7H8I9J0K1L2M3N4oXXXXX
```

The 'p' and 'o' characters mark the beginning and end delimiters of the transmitted stream respectively, and the other lower case letters all correspond to the data markers to be replaced.

2.5 Testing Protocol

Testing of the modules was needed to verify that they met the objectives. Under field conditions, the modules would utilize the repeater network to establish a data path to regional dispatch. This option was not available as the ICOM F3's provided from the BCFS were not programmed with repeater frequencies. Instead, testing was done with direct radio-to-radio transmission, which allowed the authors to test the modules thoroughly without broadcasting on the BCFS repeater channels. As a consequence, the Digital IFR Transmission device is still in need of testing through these types of channels. It is anticipated that the results obtained from transmission through the repeater network will be representative of those from radio-to-radio transmission.

Testing consisted of sending 20 IFRs through the device. Data was entered manually via the keypad and verified on the LCD screen, as would be performed under field conditions. The Reception Module was connected to a 2.4GHz PC running the custom executable that was designed for decoding the IFR. As the data was received it was presented on the screen, and subsequently verified for accuracy. Half of the IFRs were transmitted at moderate range (~1 km), and the other half were transmitted with the radios in close proximity.

2.6 Results

Testing of the Transmission and Reception Modules was an ongoing process, and although the final testing protocol consisted of only 20 IFR transmissions, hundreds were performed prior to this for thorough debugging. Following debugging, data was collected as described in Section 2.5.

Table 1 below summarizes the results from the 20 IFR transmissions. Failed transmissions included any IFR that had one or more errors. Two transmission distances were used: Bench Test (adjacent radios) and Moderate Range (~ 1 km line of sight). Transmission time constitutes the average amount of radio time that was needed to send the IFR. Details of the testing are shown in Appendix D.

Test Type	IFRs Transmitted	Perfect Transmission	Failed Transmission	Perfect Trans. Ratio	Transmission Time (s)
Bench Test	10	10	0	100%	2.7
Moderate Range	10	9	1	90%	2.7
Total	**20**	**20**	**1**	**95%**	**2.7**

Table 1 - Transmission Results

2.7 Discussion of Results

Initial test results for the Digital IFR transmission system show a strong potential for full application of the technology. The delivered prototype represents a marked improvement over the current practice of relaying IFRs vocally, decreasing transmission time by a factor of 20. Given additional time, the authors strongly believe that the transmission reliability of the system could approach 100%. Although reliability statistics for vocal IFR transmission are not readily available, mistakes during transcription of poorly enunciated IFRs are not uncommon.

As previously discussed, data integrity could be further enhanced by utilizing more sophisticated error detection and correction algorithms. The method chosen, relaying data twice and checking for inconsistencies between transmissions, is inefficient. Although this strategy has a very high probability of identifying incorrect data, it cannot reconstruct the correct value. Once an error has been detected, the data field represented by the corrupt transmission must be discarded. Forward error correction algorithms could both reduce the number of detection bits required and correct a moderate number of transmission errors, allowing faster transmission times with higher transmission reliability.

The ICOM F3 showed inconsistent delays between channel initialization (microphone cueing) and opening of the radio channel. As a result, a delay was needed between microphone cueing and data transmission that would allow all the data to be transmitted without truncation. A delay of one second was used to provide a large factor of safety and it is expected that this time could be reduced substantially with further development.

Ergonomics of the transmission module were thoroughly scrutinized. The physical dimensions of the Module are much larger than ideal, but not unreasonable considering the current stage of development. The input / output functions of the keypad and LCD screen were adequate, but could also benefit from some refinement. The most notable deficiencies of the Transmission Module include:

- Large outer dimensions
- Poor LCD contrast under low-light conditions
- Difficult keypad access with gloved hands
- Occasional lock-up of microcontroller (requires resetting)
- Slow data entry
- Questionable durability

The above shortcomings could easily be corrected on subsequent prototypes.

Useful insight into the functionality of the Reception Module was also gained during final testing. Due to the temporary nature of this module, intermediate circuitry linking the radio and computer was built on a solderless bread-board. More robust circuit construction was avoided because this unit was not intended for use as a final prototype. Linking a handheld radio to a computer was only done to facilitate preliminary testing of the Transmission Module. A finished Reception Module would likely utilize some data acquisition hardware connected directly to the BCFS radio network at a regional fire centre. All software for the Reception Module was written in LabVIEW as a stand-alone executable to allow an easy transition to this type of radio interface.

The software responsible for decoding the IFR showed reliable performance. Some minor problems identified with the current version of the reception software include:

- Unintuitive presentation of data fields
- Inconsistent program recovery after severe transmission errors
- Arbitrary choice of output file format

In general, performance of the Digital IFR Device met the objectives of the project. Connecting the Transmission and Reception Modules with their associated peripherals is straightforward, as is entering the required data. Although small improvements to the functionality and user interface of both modules would increase their user-friendliness, they are currently adequate for preliminary field testing.

2.0 Conclusions

The Digital IFR Transmission System provides a very efficient method for the BCFS to relay initial fire reports from remote field locations to regional fire centers. By digitizing data that is traditionally relayed by voice, the transmission time and error rate are reduced substantially. Automatic capture and logging of incoming IFRs provides further benefits by eliminating the need for dispatchers to transcribe IFR data.

System hardware consists of two independent battery-powered modules that interface with the BCFS radio network by plugging in to the external microphone/speaker ports of ICOM-F3 handheld radios. The Transmission Module is a field-capable prototype into which the user can enter and review IFR data and it is encased in a sturdy fire-resistant PVC case. The Reception Module is a stand-alone unit that decodes the incoming IFR information and passes it to the serial port of a PC via a DB9 serial cable. The reception software is a stand-alone LabVIEW executable that can be run on any computer on which the LabVIEW 7 runtime engine is installed. The software scans a received IFR stream for possible errors, processes and displays the IFR information to the user, and then prompts the user to save it in a user-defined log file.

The system quickly and reliably transmitted IFRs; of 20 IFRs that were transmitted with an average transmission time of 2.7 seconds, only one contained errors. When comparing broadcast times between voice and digital transmission, the IFR device showed substantial timesavings. With oral transmission of an IFR taking approximately one minute, and the IFR device needing only 2.7 seconds, the result is nearly 57 seconds of freed network airtime.

Tests of the system show that the objectives have been satisfied and potential exists for further development of the project. Large improvements could be achieved by using more elaborate error correction / detection protocols. These types of protocols could reduce the rate of failed transmissions substantially, while further decreasing transmission times. Other minor changes to the ergonomics and user-friendliness of the modules could also enhance the functionality of the system. Future considerations include two-way communication, automatic retransmission of corrupted data, and multiplexed communication over voice channels. Although these types of improvements would enhance the utility of the Digital IFR Transmission system, this device already provides a superior alternative to current methods of data communication.

3.0 Recommendations

Development of the Digital IFR Transmission system resulted in the following recommendations:

i. **A Digital IFR Transmission System should be implemented by the BCFS.**

   Based on successful preliminary testing of the IFR device, it is clear that the BCFS could realize substantial benefits from its implementation, particularly reduced radio usage and increased transmission efficiency.

ii. **The Reception and Transmission Modules should be refined to increase user-friendliness.**

   Although both modules are adequate for field-testing in their current configuration, their ergonomics are crude. The Reception Module needs a more intuitive user display and the Transmission Module needs an improved menu system and smaller casing.

iii. **A more sophisticated error handling protocol should be developed.**

   Transmission accuracy and efficiency would be improved substantially by employing better error correction / detection protocols. Algorithms such as Cyclical Redundancy Checking, Hamming codes, or Reed-Solomon codes all provide error correcting capabilities that would further enhance the utility of the digital transmission system.

iv. **The digitization of other field reports should be explored by the BCFS.**

The improvements demonstrated by the digital IFR transmission system could be applied to other field reports used by the BCFS to improve communication in several areas.

v. **The Reception Module should be integrated with BCFS fire centre hardware.**

The current form of the Reception Module was designed for the purpose of testing the Transmission Module. As such, it was not intended as a final solution for interfacing with the radio network. A more permanent solution might include mating a Data Acquisition Card (DAC) with the radio network at fire centres.

vi. **The design of the Transmission Module should be reconsidered to allow more flexible data entry.**

The current Transmission Module works adequately for inputting IFR data. In order to allow further gains from digitizing data, a new user interface would be necessary. A PDA or other such device might be able to interface with a Transmission Module and provide for a more flexible means of data entry.

Appendix A – Comparison of Deliverables with Proposal

Although the objectives stated in the project proposal were met, there were several deviations from the project plan. Most of these deviations were related to underestimates of the time required to perform tasks or achieve milestones. Relatively few changes were made to the design specified in the proposal, and for the most part changes were effected in order to improve the design, as opposed to removing a specification that could not be achieved.

The schedule specified in the proposal had a very sequential structure, where one task followed directly after another and a set of completed tasks culminated in a milestone. The actual project schedule followed a much different order; however, as it turned out many of the tasks were very inter-related and had to be completed simultaneously. For example, the IFR decoding software could not be written without a complete understanding of the transmission protocol, and the transmission protocol could not be specified without first knowing the transmission characteristics of the modems and radios.

The proposal also gave clear instructions as to which group member was to perform a specific task. However, once the schedule of the tasks began to change, it became clear that the allocation of tasks was going to shift as well. This more flexible arrangement proved to be an improvement as it reduced the dependence of one group member's work on that of another.

The most significant deviations from the proposed schedule were related to the development of the transmission protocol, design of the Transmission Module, and the programming of the decoding software. February 7th was the proposed completion date for selecting the communication protocol. Although the most important characteristics of the protocol, such as baud rate and mark and space frequencies, were determined at this time, the exact form of the IFR transmission sequence could not be determined until the capabilities of the reception software were understood. This was not achieved until mid March.

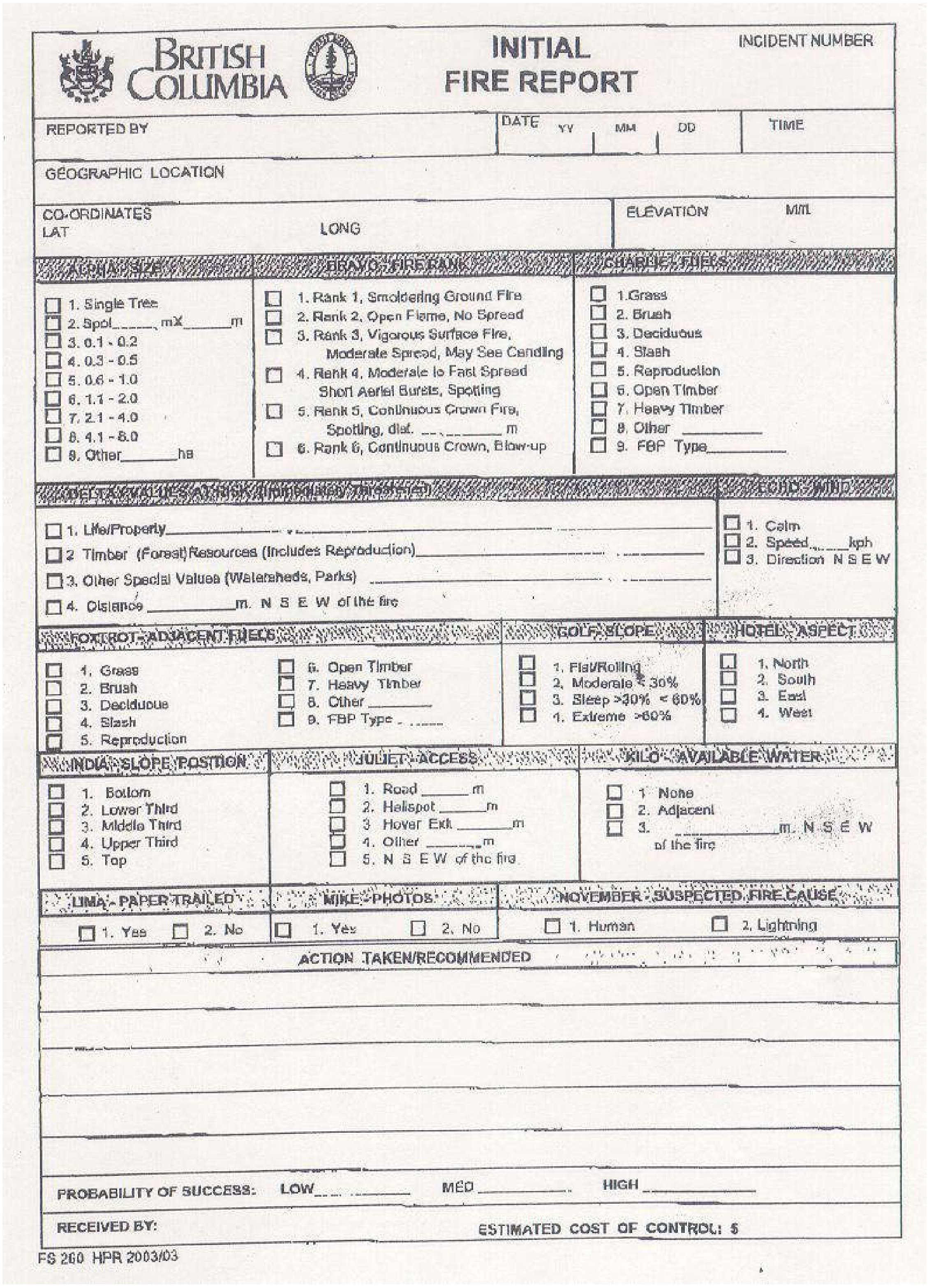
With regard to the Transmission Module, initial setup and programming of the PIC18F452 proved to be much more time-intensive than originally forecast. Five hours were allocated to programming the PIC, and this proved to be a gross underestimate. Difficulties compiling and linking C files on the networked computers in the Project Lab slowed programming considerably, and several days were lost to configuring compiler libraries and linker scripts. When the LCD display was connected to the PIC, the transmission speed of the AUSART (serial port) decreased considerably from 1200 baud. Two days were lost trying to find a method for shutting off the LCD during transmission of the data. Control of the transmitting radio also presented an unforeseen problem. Small voltage levels appearing on the external microphone line caused the radio to enter the transmit mode and approximately ten hours were spent to find a solution. These difficulties added approximately 50 hours to the development of the Transmission Module.

Several hours were spent investigating the design of the Reception Module software before it was realized that using Microsoft Visual Basic or C++ to gain access to the computer's serial ports would require advanced driver programming to bypass Windows control of hardware. Approximately three full days were then required to develop a program capable of receiving, checking, and processing a received IFR in LabVIEW 7.0 and several subsequent hours were spent debugging and enhancing the program.

The only significant negative change to the planned design was the elimination of the ability to correct errors in the received signal. Given more time, a program capable of performing bitwise operations for error checking and correcting could be developed in C++ or Visual Basic. Access to PC serial ports could be gained by purchasing a kernel driver to bypass Windows control of the hardware, or by developing a driver using Microsoft Foundation Classes and Windows dynamically linked libraries. Encasing the Transmission module in a durable, hand-sized ABS case was performed in addition to the objectives stated for the prototype, and helped to move the system beyond a simple proof of concept to a field-ready prototype capable of demonstrating its performance in a real working environment.

These deviations from the project plan were offset by a substantial increase in work performed by the authors to avoid sacrificing the objectives of the project. In hindsight, the discrepancies between the proposed project development and the actual project development could have been avoided by making more realistic estimates of the time required. This may have led to a reduced set of objectives. Many of the tasks involved learning new programming techniques, interfacing new hardware, and developing new solutions, all of which require considerable time investment before progress can be made.

Appendix B – Initial Fire Report



BRITISH COLUMBIA

INITIAL FIRE REPORT

INCIDENT NUMBER

REPORTED BY | DATE YY MM DD | TIME

GEOGRAPHIC LOCATION

CO-ORDINATES LAT LONG | ELEVATION M/m

ALPHA - SIZE

- ☐ 1. Single Tree
- ☐ 2. Spot______ mX______ m
- ☐ 3. 0.1 - 0.2
- ☐ 4. 0.3 - 0.5
- ☐ 5. 0.6 - 1.0
- ☐ 6. 1.1 - 2.0
- ☐ 7. 2.1 - 4.0
- ☐ 8. 4.1 - 8.0
- ☐ 9. Other______ ha

BRAVO - FIRE RANK

- ☐ 1. Rank 1, Smoldering Ground Fire
- ☐ 2. Rank 2, Open Flame, No Spread
- ☐ 3. Rank 3, Vigorous Surface Fire, Moderate Spread, May See Candling
- ☐ 4. Rank 4, Moderate to Fast Spread Short Aerial Bursts, Spotting
- ☐ 5. Rank 5, Continuous Crown Fire, Spotting, dist. ______ m
- ☐ 6. Rank 6, Continuous Crown, Blow-up

CHARLIE - FUELS

- ☐ 1. Grass
- ☐ 2. Brush
- ☐ 3. Deciduous
- ☐ 4. Slash
- ☐ 5. Reproduction
- ☐ 6. Open Timber
- ☐ 7. Heavy Timber
- ☐ 8. Other ______
- ☐ 9. FBP Type______

DELTA - VALUES AT RISK (Immediate Threat)

- ☐ 1. Life/Property______
- ☐ 2 Timber (Forest) Resources (Includes Reproduction)______
- ☐ 3. Other Special Values (Watersheds, Parks) ______
- ☐ 4. Distance ______ m. N S E W of the fire

ECHO - WIND

- ☐ 1. Calm
- ☐ 2. Speed______ kph
- ☐ 3. Direction N S E W

FOXTROT - ADJACENT FUELS

- ☐ 1. Grass
- ☐ 2. Brush
- ☐ 3. Deciduous
- ☐ 4. Slash
- ☐ 5. Reproduction
- ☐ 6. Open Timber
- ☐ 7. Heavy Timber
- ☐ 8. Other ______
- ☐ 9. FBP Type ______

GOLF - SLOPE

- ☐ 1. Flat/Rolling
- ☐ 2. Moderate < 30%
- ☐ 3. Steep >30% < 60%
- ☐ 4. Extreme >60%

HOTEL - ASPECT

- ☐ 1. North
- ☐ 2. South
- ☐ 3. East
- ☐ 4. West

INDIA - SLOPE POSITION

- ☐ 1. Bottom
- ☐ 2. Lower Third
- ☐ 3. Middle Third
- ☐ 4. Upper Third
- ☐ 5. Top

JULIET - ACCESS

- ☐ 1. Road ______ m
- ☐ 2. Helispot ______ m
- ☐ 3 Hover Exit ______ m
- ☐ 4. Other ______ m
- ☐ 5. N S E W of the fire

KILO - AVAILABLE WATER

- ☐ 1 None
- ☐ 2. Adjacent
- ☐ 3. ______ m. N S E W of the fire

LIMA - PAPER TRAILED ☐ 1. Yes ☐ 2. No

MIKE - PHOTOS ☐ 1. Yes ☐ 2. No

NOVEMBER - SUSPECTED FIRE CAUSE ☐ 1. Human ☐ 2. Lightning

ACTION TAKEN/RECOMMENDED

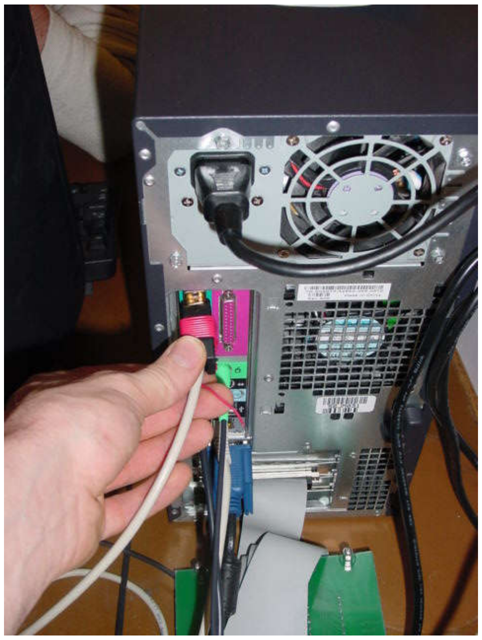
PROBABILITY OF SUCCESS: LOW______ MED ______ HIGH ______

RECEIVED BY: ESTIMATED COST OF CONTROL: $

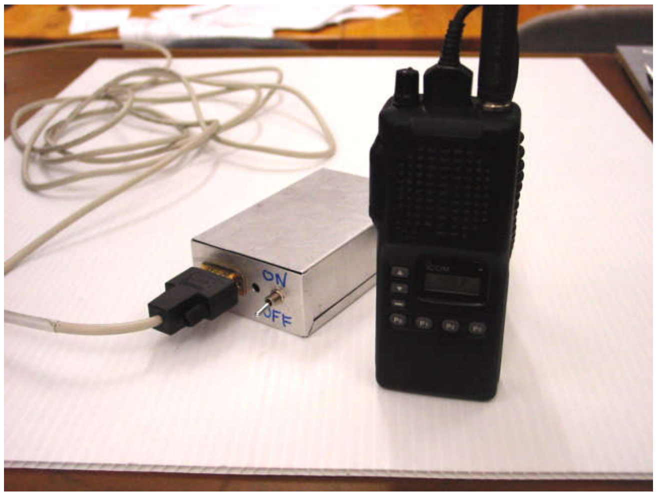
FS 260 HPR 2003/03

Appendix C – Operation Manual

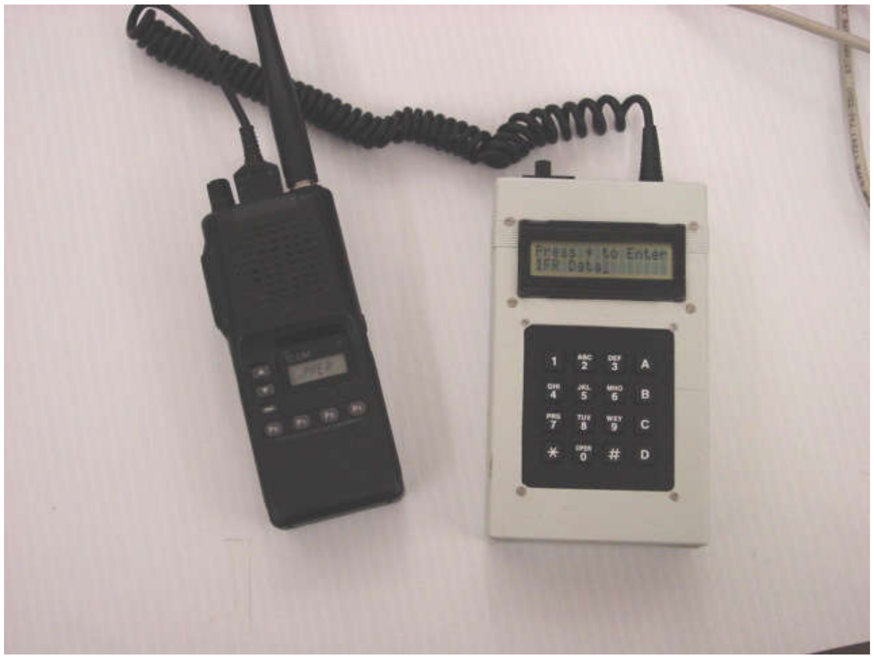
1) Install Reception Module software onto a PC with a functioning COM1 Port.
2) Hook up the RS232 cable to the Reception Module and computer. **Note:** The end that is wrapped with red wire needs to be connected to the computer.



3) Connect the radio to the Reception Module and turn on the radio to HALF VOLUME. Turn on the Reception Module switch.



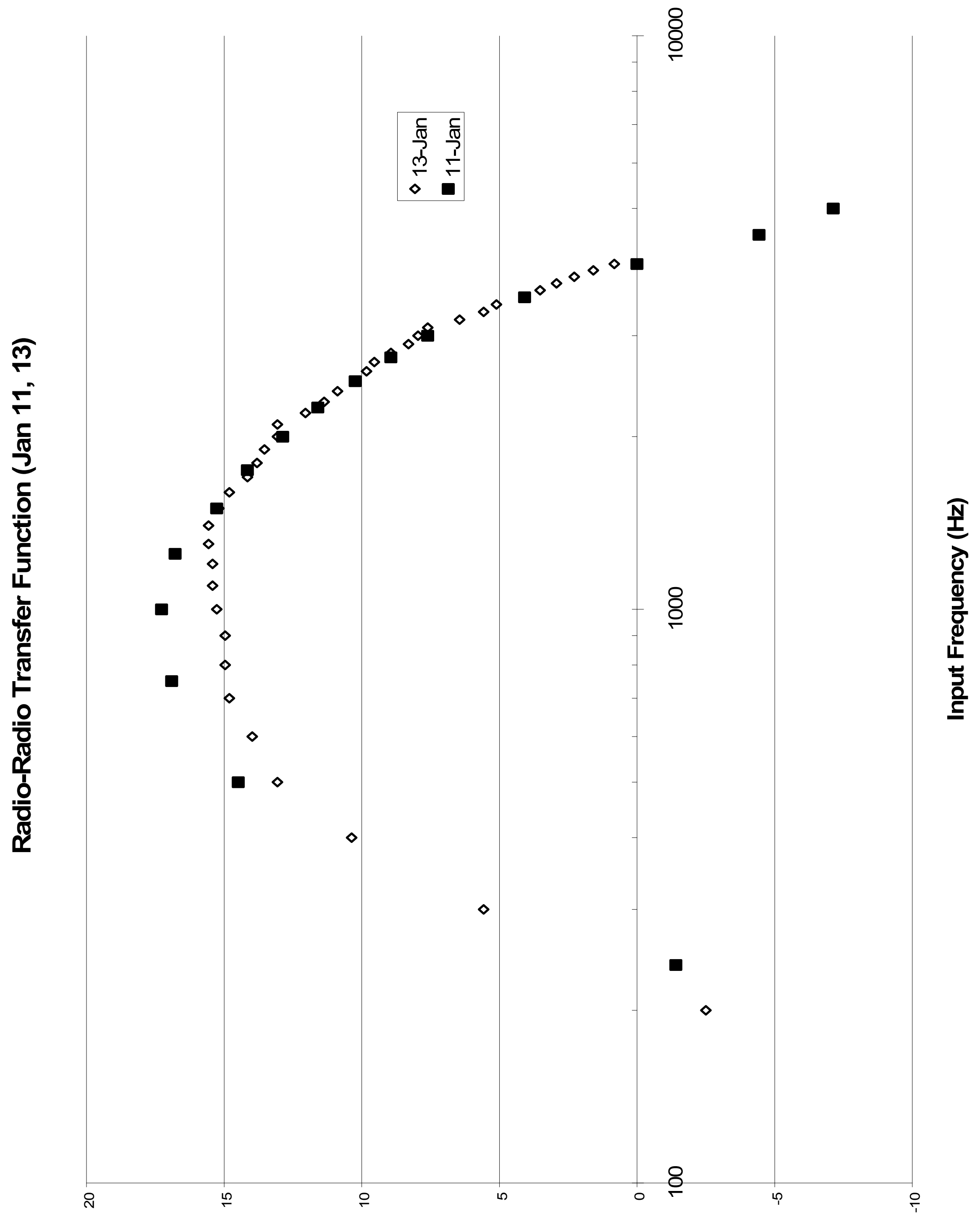
4) Attach the transmitting radio to the Transmission Module. Turn them both on.



5) Start the IFR program in **continuous run mode** (depicted as a set of circular arrows on the top left portion of the screen).
6) Enter IFRs – some notes on the various data fields:
   i. * button is used to accept data
   ii. D button is used to delete data
   iii. Latitude and Longitude require 9 characters before entering
   iv. Elevation requires 4 characters before entering
   v. Letter Data Fields only accept one character before entering

Appendix D – Testing Results

Data Transmission Test Completed 02/24/2004	
Input test stream	40......................................80......................................120.....................................160......................................200......................................240
Receiving radio volume	**Received data stream**
3600mV – 100%	40......................................80......................................120.....................................160......................................200......................................240
1800mV – 50%	40......................................80......................................120......................................160......................................200.......................................240
1200mV – 33%	......................................40......................................80......................................120......................................160......................................200..240
900mV – 25%	....40...┐ rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr...............................40...................
720mV – 20% min.	É2rrrrrb0rpr"r2rrr2rrrrrrrrrbprprrrrrbrrrr2rrrpprrrbr20rrRrrrr`r÷ ÷ ÷ r0rbpRrrprr2rrrrr↨ Rrprrpbr2rrr2Â,prRrpbbrpp`bpb2prrbrrRrrrrr2rrr*É ‰ÁÁÉÁÉ‰ÉÁÁÉÉÉpp2pprrrrrprr2rrprrpprrrrRrbrrprrrrrrrrrpprrrr2prrrr`rr'¢rbbrrprrrrrp2rrrrrrrrpprrprprprprrrprprr,€rprRrr2rpr0`rrrpr2rbrrrÀ,rrrbppbrrrrrÁIÉÉÁÉÉÉ....(.&p

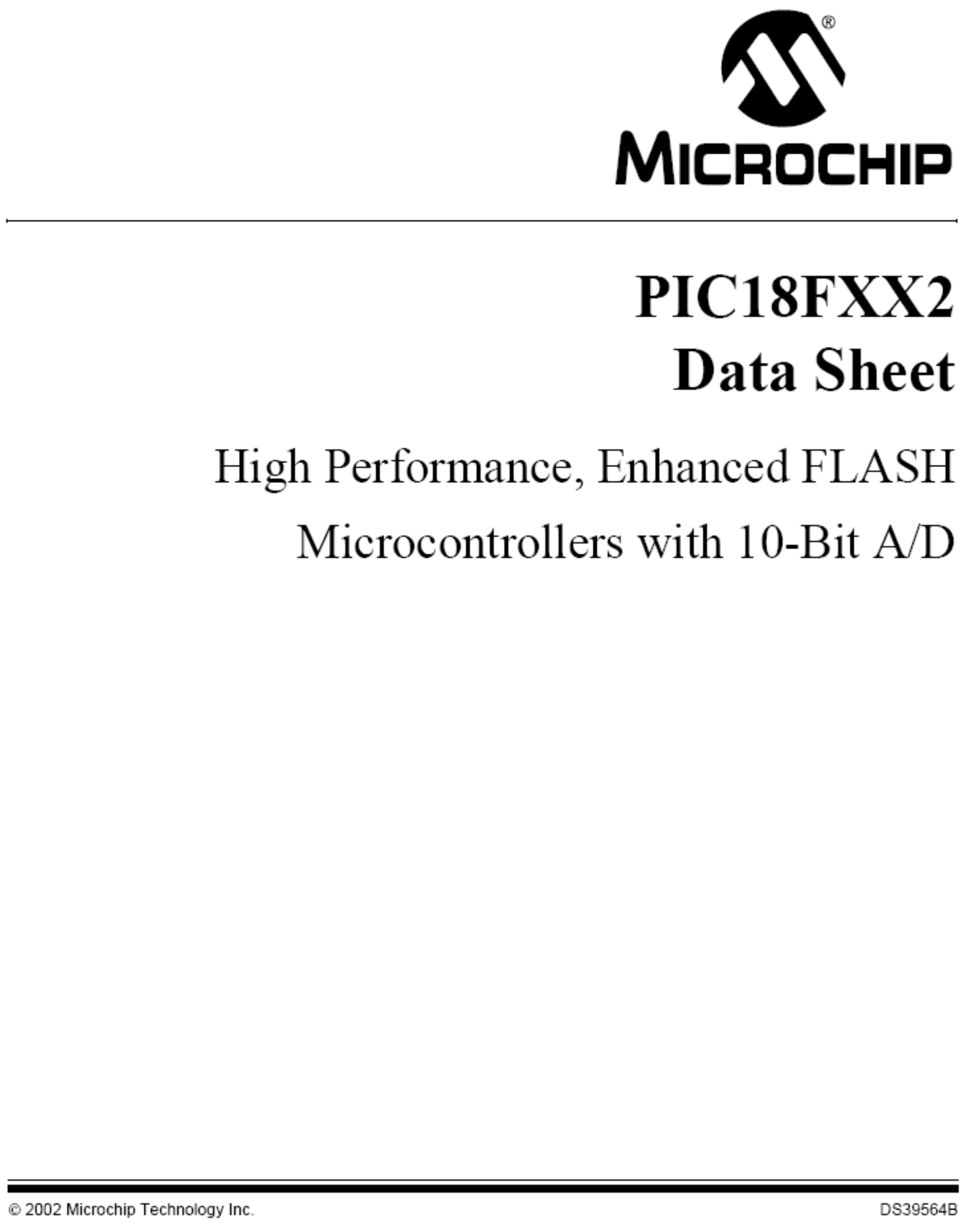

Radio-Radio Transfer Function (Jan 11, 13)
Output Voltage (dB)
20
15
10
5
0
-5
-10
100
1000
10000
Input Frequency (Hz)
13-Jan
11-Jan

IFR Transmission test – 03/25/2004

```
****************************
INITIAL FIRE REPORT
RECEIVED: 3/25/2004 10:11:34 AM

LATITUDE: 217443564
LONGITUDE: 012416744
ELEVATION: 0850
ALPHA: 1
BRAVO: 3
CHARLIE: 2
DELTA: 5
ECHO: 4
FOXTROT: 6
GOLF: 2
HOTEL: 6
INDIA: 4
JULIET: 4
KILO: 8
LIMA: 1
MIKE: 9
NOVEMBER: 3

****************************
INITIAL FIRE REPORT
RECEIVED: 3/25/2004 10:12:16 AM

LATITUDE: 210653187
LONGITUDE: 002664754
ELEVATION: 1540
ALPHA: 2
BRAVO: 6
CHARLIE: 4
DELTA: 2
ECHO: 3
FOXTROT: 8
GOLF: 1
HOTEL: 6
INDIA: 2
JULIET: 4
KILO: 4
LIMA: 2
MIKE: 5
NOVEMBER: 3

****************************
INITIAL FIRE REPORT
RECEIVED: 3/25/2004 1:49:09 PM

LATITUDE: 122454344
LONGITUDE: 306995417
ELEVATION: 1890
ALPHA: 3
BRAVO: 9
CHARLIE: 5
DELTA: 1
ECHO: 3
FOXTROT: B
GOLF: 4
HOTEL: 5
INDIA: 6
JULIET: 2
KILO: 1
LIMA: 6
MIKE: 3
NOVEMBER: 6

****************************
INITIAL FIRE REPORT
RECEIVED: 3/25/2004 2:51:47 PM

LATITUDE: 225241865
LONGITUDE: 2152**ERROR**5355
ELEVATION: 2155
ALPHA: 2
BRAVO: 3
CHARLIE: 4
DELTA: 1
ECHO: 6
FOXTROT: **ERROR**
GOLF: 1
HOTEL: 2
INDIA: 3
JULIET: 4
KILO: 6
LIMA: 1
MIKE: 3
NOVEMBER: 5
```

Final Test Data					
Transmission #	Range	Pass	Fail	Time (s)	Comments
1	Close	✓		2.6	screen blanked out after transmission
2	Close	✓		2.7	
3	Close	✓		2.7	
4	Close	✓		2.8	
5	Close	✓		2.5	
6	Close	✓		2.7	
7	Close	✓		2.7	
8	Close	✓		2.6	
9	Close	✓		2.9	
10	Close	✓		2.7	
11	Moderate	✓		2.5	
12	Moderate	✓		2.7	
13	Moderate	✓		2.8	
14	Moderate	✓		2.7	
15	Moderate		✓	2.7	54 errors, unknown cause
16	Moderate	✓		3	
17	Moderate	✓		2.9	
18	Moderate	✓		2.5	
19	Moderate	✓		2.6	
20	Moderate	✓		2.6	
Total		**19**	**1**	**2.695**	

Appendix E – Abbreviated Data Sheets



MICROCHIP

PIC18FXX2 Data Sheet

High Performance, Enhanced FLASH Microcontrollers with 10-Bit A/D

DS39564B

Excerpt from PIC18F452 Data Sheet

MICROCHIP

PIC18FXX2

28/40-pin High Performance, Enhanced FLASH Microcontrollers with 10-Bit A/D

High Performance RISC CPU:

- C compiler optimized architecture/instruction set
  - Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 32 Kbytes
- Linear data memory addressing to 1.5 Kbytes

Device	On-Chip Program Memory		On-Chip RAM (bytes)	Data EEPROM (bytes)
	FLASH (bytes)	# Single Word Instructions		
PIC18F242	16K	8192	768	256
PIC18F252	32K	16384	1536	256
PIC18F442	16K	8192	768	256
PIC18F452	32K	16384	1536	256

- Up to 10 MIPs operation:
  - DC - 40 MHz osc./clock input
  - 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules. CCP pins that can be configured as:
  - Capture input: capture is 16-bit, max. resolution 6.25 ns ($T_{CY}/16$)
  - Compare is 16-bit, max. resolution 100 ns (T_{CY})
  - PWM output: PWM resolution is 1- to 10-bit, max. PWM freq. @: 8-bit resolution = 156 kHz
    10-bit resolution = 39 kHz
- Master Synchronous Serial Port (MSSP) module, Two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I^2C™ Master and Slave mode

Peripheral Features (Continued):

- Addressable USART module:
  - Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

Analog Features:

- Compatible 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
  - Conversion available during SLEEP
  - Linearity $\leq$ 1 LSb
- Programmable Low Voltage Detection (PLVD)
  - Supports interrupt on-Low Voltage Detection
- Programmable Brown-out Reset (BOR)

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory
- FLASH/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- Single supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins

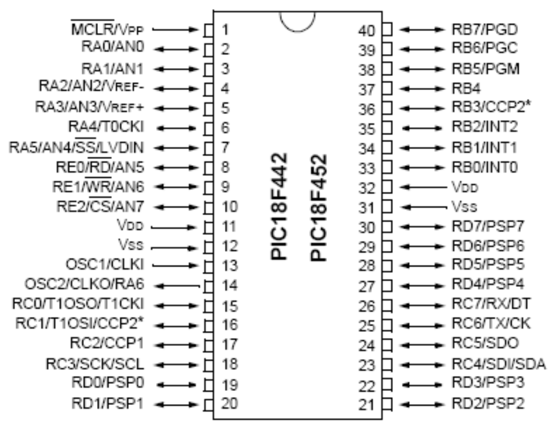
CMOS Technology:

- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption:
  - < 1.6 mA typical @ 5V, 4 MHz
  - 25 μA typical @ 3V, 32 kHz
  - < 0.2 μA typical standby current

Excerpt from PIC18F452 Data Sheet

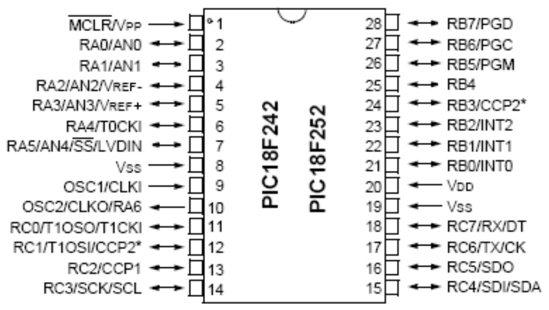
Pin Diagrams (Cont.'d)

DIP



Note: Pin compatible with 40-pin PIC16C7X devices.

DIP, SOIC

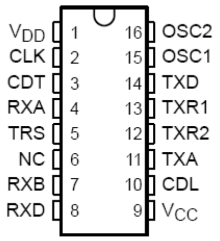


* RB3 is the alternate pin for the CCP2 pin multiplexing.

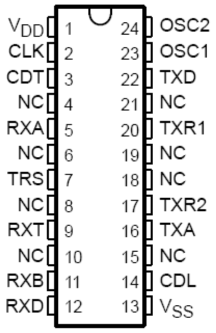
Excerpt from PIC18F452 Data Sheet

- **Single-Chip Frequency-Shift-Keying (FSK) Modem**
- **Meet Both Bell 202 and CCITT V23 Specifications**
- **Transmit Modulation at 75, 150, 600, and 1200 Baud**
- **Receive Demodulation at 5, 75, 150, 600, and 1200 Baud**
- **Half-Duplex Operation Up to 1200 Baud Transmit and Receive**
- **Full-Duplex Operation Up to 1200 Baud Transmit and 150 Baud Receive**
- **On-Chip Group Equalization and Transmit/Receive Filtering**
- **Carrier-Detect-Level Adjustment and Carrier-Fail Output**
- **Single 5-V Power Supply**
- **Low Power Consumption**
- **Reliable CMOS Silicon-Gate Technology**

J OR N PACKAGE
(TOP VIEW)



DW PACKAGE
(TOP VIEW)



NC – No internal connection

D package are available taped and reeled. Add the R suffix to device type (e.g., YCM3105DWLR).

description

The TCM3105 is a single-chip asynchronous frequency-shift-keying (FSK) voice-band modem that uses silicon-gate CMOS technology to implement a switched-capacitor architecture. It is pin selectable (TXR1, TXR2, and TRS) for a wide range of transmit/receive baud rates and is compatible with the applicable BELL 202 or CCITT V23 standards. Operation is fully reversible, thereby allowing both forward and backward channels to be used simultaneously.

The transmitter is a programmable frequency synthesizer that provides two output frequencies (on TXA), representing the marks and spaces of the digital signal present on TXD.

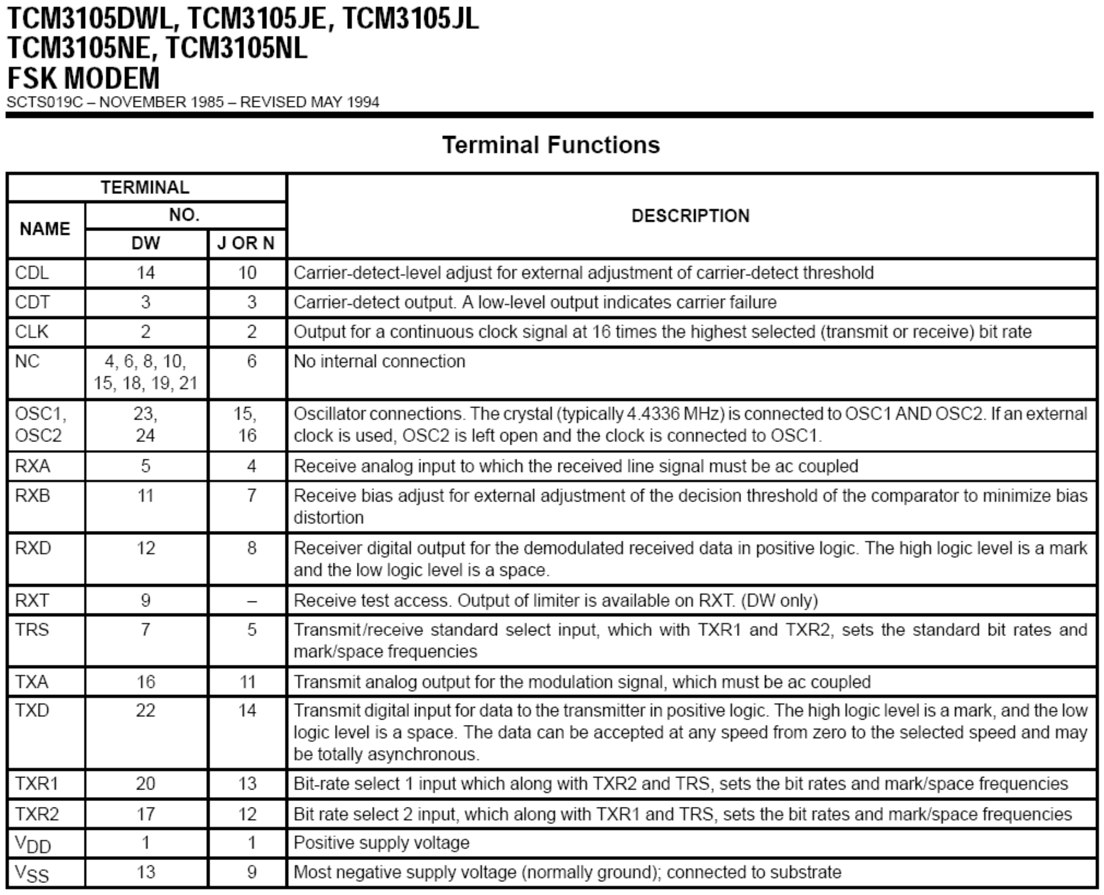
The receive section is responsible for the demodulation of the analog signal appearing at the RXA input and is based on the principle of frequency-to-voltage conversion. This section contains a group delay equalizer (to correct phase distortion), automatic gain control, carrier-detect-level adjustment, and bias-distortion adjustment, thereby optimizing performance and giving the lowest possible bit error rate.

Carrier-detect information is given to the system by means of the carrier-detect circuits, which set a flag on the CDT output if the level of received in-band energy falls below a value set on the CDL input for a specified minimum duration.

The TCM3105JE and TCM3105NE are characterized for operation from −40°C to 85°C.The TCM3105DWL, TCM3105JL, and TCM3105NL are characterized for operation from 0°C to 70°C.

Caution. These devices have limited built-in protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

Excerpt from TCM 3105 Data Sheet

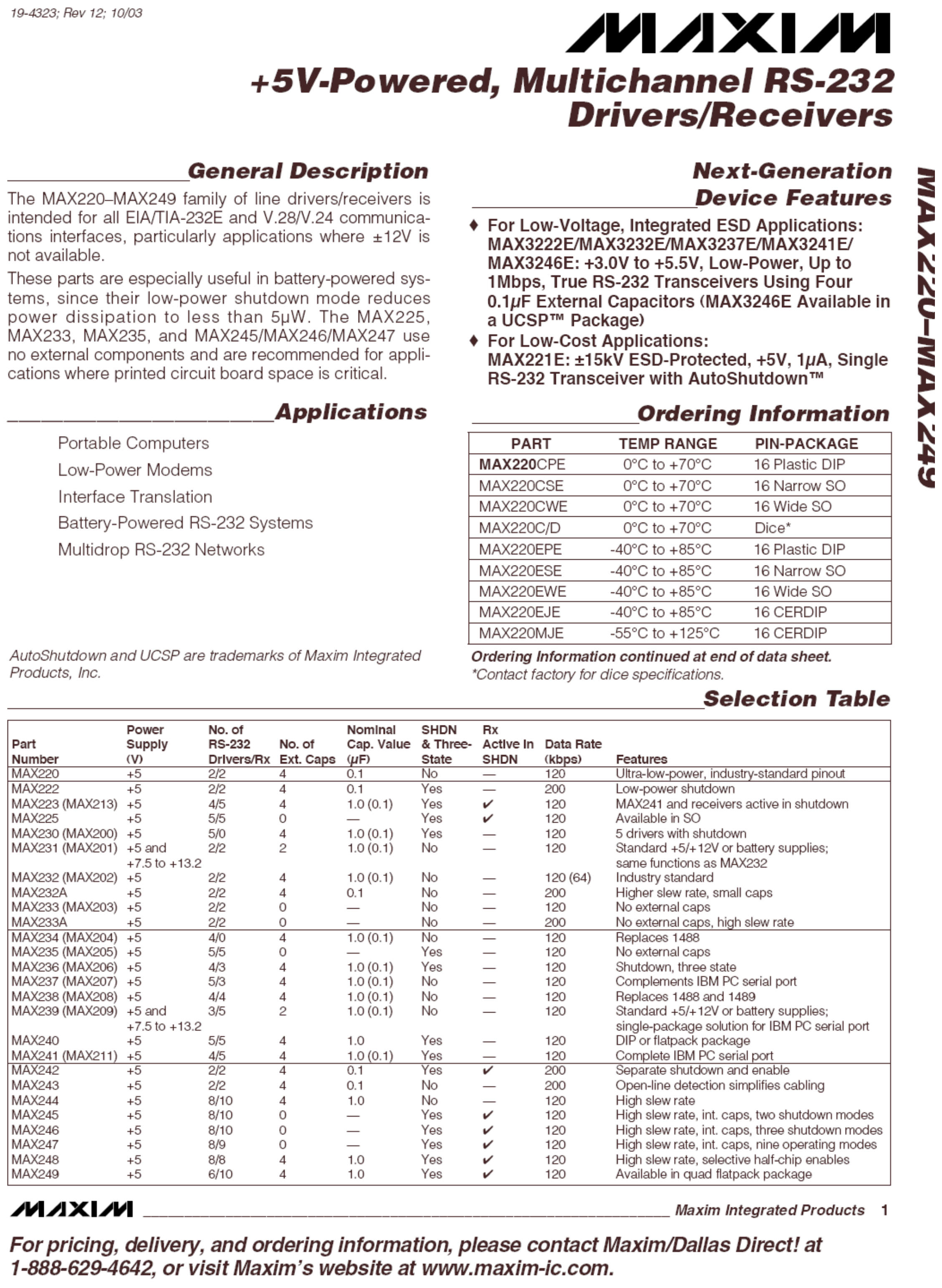


TCM3105DWL, TCM3105JE, TCM3105JL
TCM3105NE, TCM3105NL
FSK MODEM
SCTS019C – NOVEMBER 1985 – REVISED MAY 1994

Terminal Functions

TERMINAL			DESCRIPTION
NAME	NO.		
	DW	J OR N	
CDL	14	10	Carrier-detect-level adjust for external adjustment of carrier-detect threshold
CDT	3	3	Carrier-detect output. A low-level output indicates carrier failure
CLK	2	2	Output for a continuous clock signal at 16 times the highest selected (transmit or receive) bit rate
NC	4, 6, 8, 10, 15, 18, 19, 21	6	No internal connection
OSC1, OSC2	23, 24	15, 16	Oscillator connections. The crystal (typically 4.4336 MHz) is connected to OSC1 AND OSC2. If an external clock is used, OSC2 is left open and the clock is connected to OSC1.
RXA	5	4	Receive analog input to which the received line signal must be ac coupled
RXB	11	7	Receive bias adjust for external adjustment of the decision threshold of the comparator to minimize bias distortion
RXD	12	8	Receiver digital output for the demodulated received data in positive logic. The high logic level is a mark and the low logic level is a space.
RXT	9	–	Receive test access. Output of limiter is available on RXT. (DW only)
TRS	7	5	Transmit/receive standard select input, which with TXR1 and TXR2, sets the standard bit rates and mark/space frequencies
TXA	16	11	Transmit analog output for the modulation signal, which must be ac coupled
TXD	22	14	Transmit digital input for data to the transmitter in positive logic. The high logic level is a mark, and the low logic level is a space. The data can be accepted at any speed from zero to the selected speed and may be totally asynchronous.
TXR1	20	13	Bit-rate select 1 input which along with TXR2 and TRS, sets the bit rates and mark/space frequencies
TXR2	17	12	Bit rate select 2 input, which along with TXR1 and TRS, sets the bit rates and mark/space frequencies
V_{DD}	1	1	Positive supply voltage
V_{SS}	13	9	Most negative supply voltage (normally ground); connected to substrate

Excerpt from TCM 3105 Data Sheet



19-4323; Rev 12; 10/03

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

MAX220–MAX249

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5μW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop RS-232 Networks

AutoShutdown and UCSP are trademarks of Maxim Integrated Products, Inc.

Next-Generation Device Features

- ♦ **For Low-Voltage, Integrated ESD Applications: MAX3222E/MAX3232E/MAX3237E/MAX3241E/ MAX3246E: +3.0V to +5.5V, Low-Power, Up to 1Mbps, True RS-232 Transceivers Using Four 0.1μF External Capacitors (MAX3246E Available in a UCSP™ Package)**
- ♦ **For Low-Cost Applications: MAX221E: ±15kV ESD-Protected, +5V, 1μA, Single RS-232 Transceiver with AutoShutdown™**

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.
**Contact factory for dice specifications.*

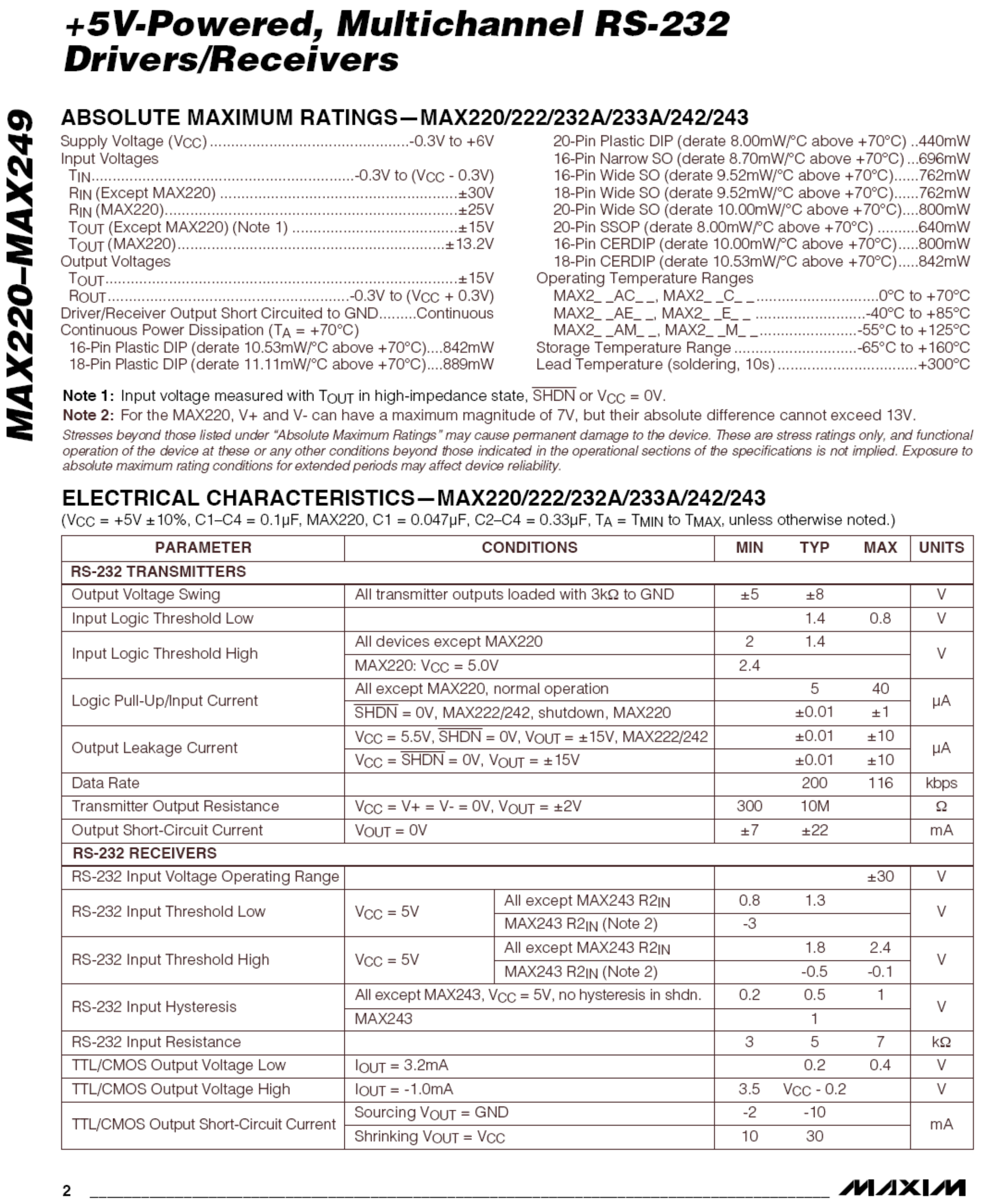
Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (μF)	SHDN & Three-State	Rx Active In SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✓	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✓	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✓	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✓	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✓	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✓	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✓	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✓	120	Available in quad flatpack package

MAXIM ***Maxim Integrated Products*** **1**

For pricing, delivery, and ordering information, please contact Maxim/Dallas Direct! at 1-888-629-4642, or visit Maxim's website at www.maxim-ic.com.

Excerpt from MAX232 Data Sheet



MAX220–MAX249

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ....-0.3V to +6V
Input Voltages
T_{IN}..-0.3V to (V_{CC} - 0.3V)
R_{IN} (Except MAX220) ....±30V
R_{IN} (MAX220)...±25V
T_{OUT} (Except MAX220) (Note 1) ....±15V
T_{OUT} (MAX220)..±13.2V
Output Voltages
T_{OUT}...±15V
R_{OUT}..-0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND.........Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C)....842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C)....889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) ..440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ...696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C)......762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C)......762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C)....800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C).....800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C).....842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ....0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ....-40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _.......................-55°C to +125°C
Storage Temperature Range ....-65°C to +160°C
Lead Temperature (soldering, 10s) ....+300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

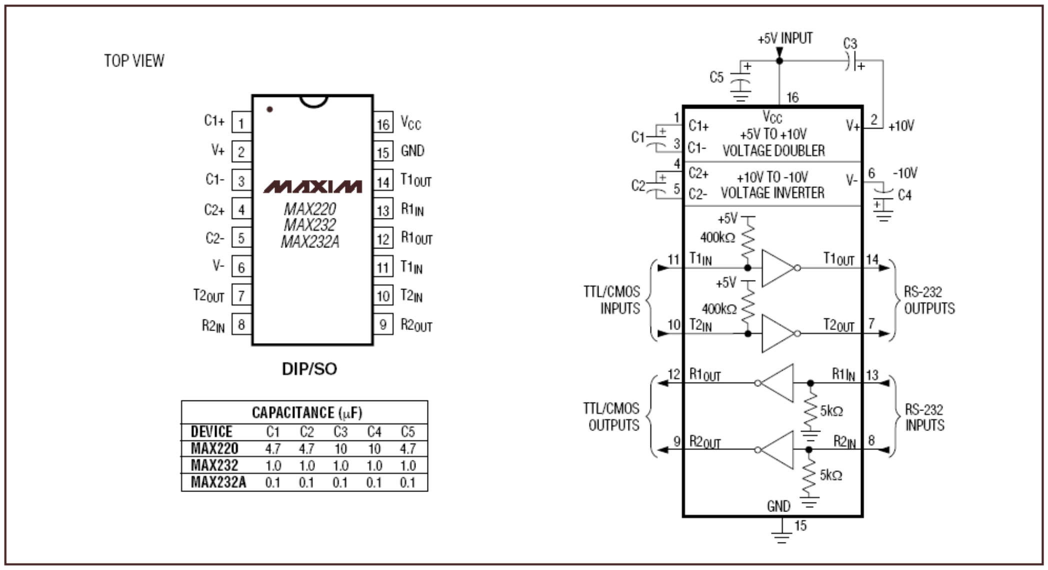
ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1μF, MAX220, C1 = 0.047μF, C2–C4 = 0.33μF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	μA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	μA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kbps
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

2 MAXIM

Excerpt from MAX232 Data Sheet



CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

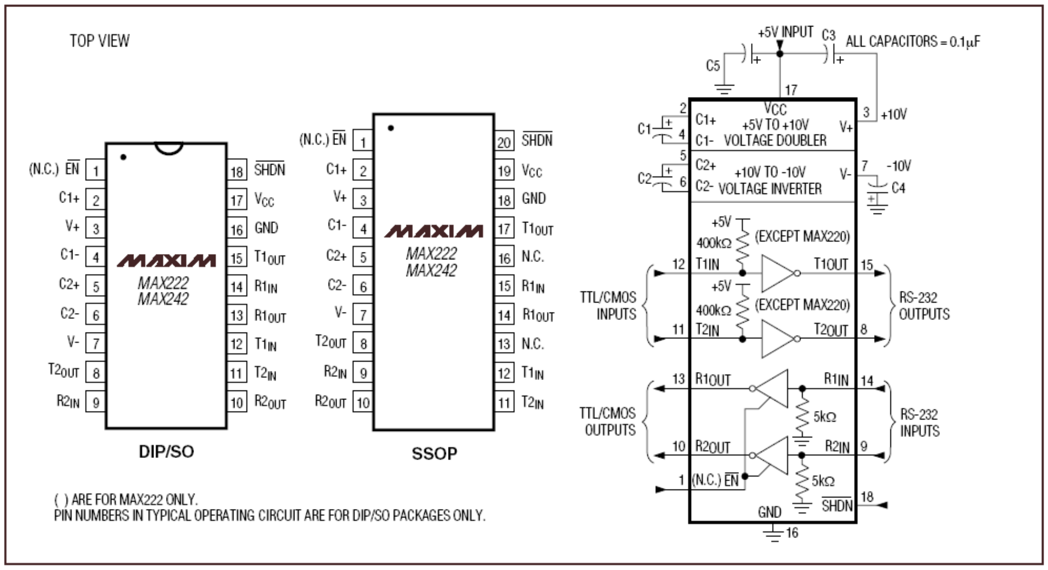


Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit

Excerpt from MAX232 Data Sheet

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

MC78XX/LM78XX/MC78XXA

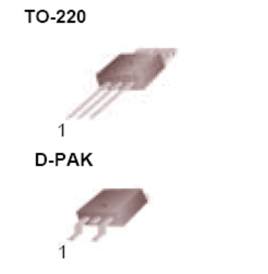
3-Terminal 1A Positive Voltage Regulator

Features

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

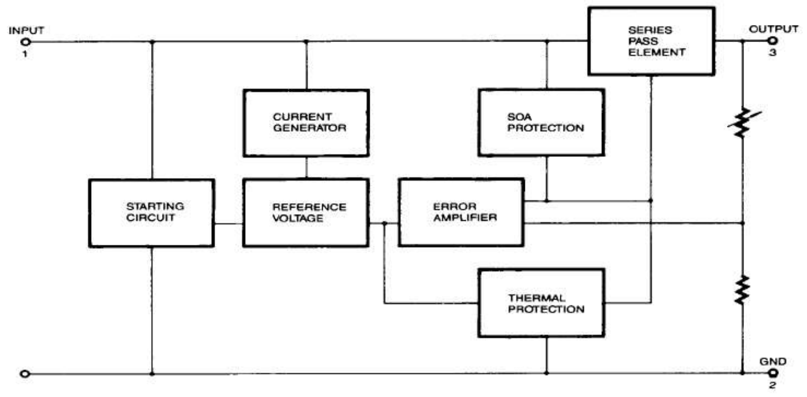
Description

The MC78XX/LM78XX/MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.



1. Input 2. GND 3. Output

Internal Block Digram



Rev. 1.0.1

Excerpt from LM7805 Data Sheet

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input Voltage (for V_O = 5V to 18V) (for V_O = 24V)	V_I V_I	35 40	V V
Thermal Resistance Junction-Cases (TO-220)	$R_{\theta JC}$	5	$^oC/W$
Thermal Resistance Junction-Air (TO-220)	$R_{\theta JA}$	65	$^oC/W$
Operating Temperature Range	T_{OPR}	0 ~ +125	^oC
Storage Temperature Range	T_{STG}	-65 ~ +150	^oC

Electrical Characteristics (MC7805/LM7805)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I = 10V, C_I= 0.33µF, C_O= 0.1µF, unless otherwise specified)

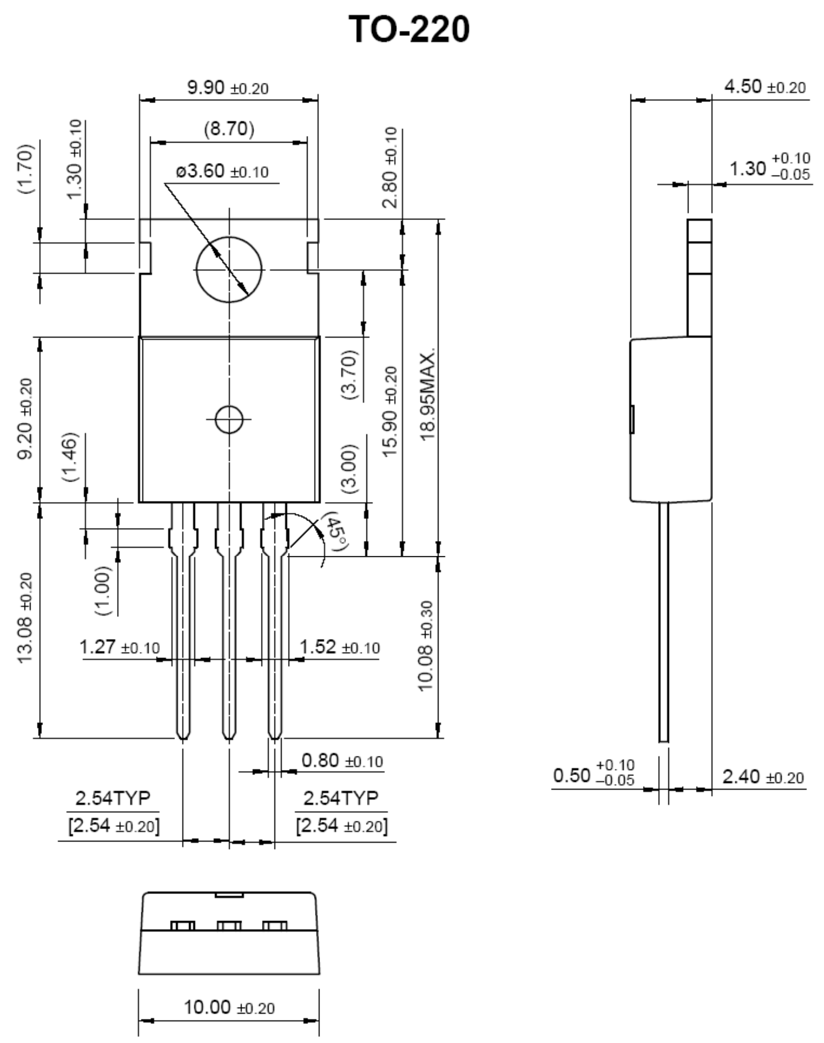
Parameter	Symbol	Conditions		MC7805/LM7805			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25 ^oC		4.8	5.0	5.2	V
		5.0mA ≤ Io ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 ^oC	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 ^oC	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25 ^oC		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/ ^oC
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25 ^oC		-	42	-	µV/Vo
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25 ^oC		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25 ^oC		-	230	-	mA
Peak Current	I_{PK}	T_J =+25 ^oC		-	2.2	-	A

Note:
1. Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty is used.

Excerpt from LM7805 Data Sheet

Mechanical Dimensions

Package



Excerpt from LM7805 Data Sheet

1.5 Pin Assignments

Table 1.2

Pin Number	Symbol
1	V_{ss}
2	V_{cc}
3	V_{ee}
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7

Pin Descriptions:

Table 1.3 List of terminal functions

Signal name	No. of Lines	Input/Output	Connected to	Function
DB4 ~ DB7	4	Input/Output	MPU	4 lines of high order data bus. Bi-directional transfer of data between MPU and module is done through these lines. Also DB_7 can be used as a busy flag. These lines are used as data in 4 bit operation.
DB0 ~ DB3	4	Input/Output	MPU	4 lines of low order data bus. Bi-directional transfer of data between MPU and module is done through these lines. In 4 bit operation, these are not used and should be grounded.
E	1	Input	MPU	Enable - Operation start signal for data read/write.
R/W	1	Input	MPU	Signal to select Read or Write "0": Write "1": Read
RS	1	Input	MPU	Register Select "0": Instruction register (Write) : Busy flag; Address counter (Read) "1": Data register (Write, Read)
Vee	1		Power Supply	Terminal for LCD drive power source.
Vcc	1		Power Supply	+5V
Vss	1		Power Supply	0V (GND)
E1	1	Input	MPU	Enable 1 - Operation start signal for data Read/Write of upper 2

Excerpt from Optrex DMC-16207 Data Sheet

Table 5.3

Standard Character Font Table

High order bit / Low order bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
X X X X0000	CG RAM (1)		0	@	P	`	p		ー	タ	ミ	α	p
X X X X0001	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ä	q
X X X X0010	(3)	"	2	B	R	b	r	「	イ	ツ	メ	β	θ
X X X X0011	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
X X X X0100	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
X X X X0101	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
X X X X0110	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
X X X X0111	(8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
X X X X1000	(1)	(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
X X X X1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
X X X X1010	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
X X X X1011	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	ˣ	万
X X X X1100	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷

Excerpt from Optrex DMC-16207 Data Sheet

Appendix F – Transmission Module Source Code

```
/****************************************************
FILE: Radio2.c
DATE: March 30, 2004
AUTHOR:  Roy Belak
This file provides the main program
****************************************************/



#include "Radio.h"
#include <Keypad.h>
#include <xlcd.h>
#include <delays.h> //for delay functions
#include <stdlib.h> //for data conversion functions for debugging
#include <usart.h>  //for usart (serial port) functions






const rom char null = (char) 0x00;
int j,y,t,p,k,q;                                        //Characters for the for loops
char Longitude [10];// = {0,0,0,0,0,0,0,0,0,null};
char Latitude [10];// = {0,0,0,0,0,0,0,0,0,null};
char Elevation [5];// = {0,0,0,0,null};
char data_output [76];
char review;
char review_done;
char alpha;
char bravo;
char charlie;
char delta;
char echo;
char foxtrot;
char golf;
char hotel;
char india;
char juliet;
char kilo;
char lima;
char mike;
char november;






void main (void)
 {

  char space[4]= {' ','=',' ',null};


  Delay10KTCYx(10);
                //Initialize sequence
                Initialize_Ports();
                Initialize_LCD ();
                Initialize_Keypad ();
                Initialize_USART();

                Longitude [9] = null;
                Latitude [9]=           null;
                Elevation [4]=          null;




                putrsXLCD ("Press * to Enter");     //Prompt user to begin IFR program
                skip_characters(16);
```

```
putrsXLCD ("IFR Data");
Wait_For_Enter();




putrsXLCD("Latitude");                            //Prompt user for Latitude
skip_characters(8);
Get_Keypad_String(Latitude,9,1);
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);


putrsXLCD("Longitude");                                   //Prompt user for Longitude
skip_characters(9);
Get_Keypad_String(Longitude,9,1);
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);


putrsXLCD("Elevation (m)");                       //Prompt user for Elevation
skip_characters(13);
Get_Keypad_String(Elevation,4,0);
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);



putrsXLCD ("ALPHA");                              //Prompt user for fire size
skip_characters (5);
alpha = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);


putrsXLCD ("BRAVO");                                      //Prompt user for fire rank
skip_characters (5);
bravo = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);


putrsXLCD ("CHARLIE");                                    //Prompt user for fuels
skip_characters (7);
charlie = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("DELTA");                              //Prompt user for Values at risk
skip_characters (5);
delta = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("ECHO");                                       //Prompt user for Wind
skip_characters (4);
echo = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("FOXTROT");                                    //Prompt user for Adjacent Fuels
skip_characters (7);
foxtrot = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("GOLF");                                       //Prompt user for Slope
skip_characters (4);
golf = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);


putrsXLCD ("HOTEL");                              //Prompt user for Aspect
skip_characters (5);
hotel = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);
```

```
putrsXLCD ("INDIA");                                        //Prompt user for Slope Position
skip_characters (5);
india = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("JULIET");                                       //Prompt user for Access
skip_characters (6);
juliet = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("KILO");                                                 //Prompt user for Available water
skip_characters (4);
kilo = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("LIMA");                                                 //Prompt user for paper trailed
skip_characters (4);
lima = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("MIKE");                                                 //Prompt user for photos
skip_characters (4);
mike = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);

putrsXLCD ("NOVEMBER");                                             //Prompt user for Suspected cause of Fire
skip_characters (8);
november = Get_Data_Field();
WriteCmdXLCD(CLEAR_DISP);
Delay1KTCYx(10);



putrsXLCD("Review Data?");                                  //Prompt user to review the entered data
skip_characters(12);
putrsXLCD("* = Yes   D = No");


review = review_input ();

if (review == '*')
{
        putrsXLCD("Press * to Scrol");
        Wait_For_Enter();
        WriteCmdXLCD(CLEAR_DISP);


        putrsXLCD("Longitude");
        skip_characters(9);
        putsXLCD(Longitude);
        Delay1KTCYx(10);
        Wait_For_Enter();
        WriteCmdXLCD(CLEAR_DISP);

        putrsXLCD("Latitude");
        skip_characters(8);
        putsXLCD(Latitude);
        Delay1KTCYx(10);
        Wait_For_Enter();
        WriteCmdXLCD(CLEAR_DISP);


        putrsXLCD("Elevation (m)");
        skip_characters(13);
        putsXLCD(Elevation);
        Delay1KTCYx(10);
        Wait_For_Enter();
        WriteCmdXLCD(CLEAR_DISP);



        putrsXLCD ("ALPHA");
        putsXLCD (space);
        WriteDataXLCD(alpha);
```

```
skip_characters(9);


putrsXLCD ("BRAVO");
putsXLCD (space);
WriteDataXLCD(bravo);
Delay1KTCYx(10);
Wait_For_Enter();
WriteCmdXLCD(CLEAR_DISP);


putrsXLCD ("CHARLIE");
putsXLCD (space);
WriteDataXLCD(charlie);
skip_characters(11);


putrsXLCD ("DELTA");
putsXLCD (space);
WriteDataXLCD(delta);
Delay1KTCYx(10);
Wait_For_Enter();
WriteCmdXLCD(CLEAR_DISP);


putrsXLCD ("ECHO");
putsXLCD (space);
WriteDataXLCD(echo);
skip_characters(8);


putrsXLCD ("FOXTROT");
putsXLCD (space);
WriteDataXLCD(foxtrot);
Delay1KTCYx(10);
Wait_For_Enter();
WriteCmdXLCD(CLEAR_DISP);


putrsXLCD ("GOLF");
putsXLCD (space);
WriteDataXLCD(golf);
skip_characters(8);


putrsXLCD ("HOTEL");
putsXLCD (space);
WriteDataXLCD(hotel);
Delay1KTCYx(10);
Wait_For_Enter();
WriteCmdXLCD(CLEAR_DISP);


putrsXLCD ("INDIA");
putsXLCD (space);
WriteDataXLCD(india);
skip_characters(9);


putrsXLCD ("JULIET");
putsXLCD (space);
WriteDataXLCD(juliet);
Delay1KTCYx(11);
Wait_For_Enter();
WriteCmdXLCD(CLEAR_DISP);


putrsXLCD ("KILO");
putsXLCD (space);
WriteDataXLCD(kilo);
skip_characters(8);


putrsXLCD ("LIMA");
putsXLCD (space);
WriteDataXLCD(lima);
Delay1KTCYx(9);
Wait_For_Enter();
WriteCmdXLCD(CLEAR_DISP);
```

```
		putrsXLCD ("MIKE");
		putsXLCD (space);
		WriteDataXLCD(mike);
		skip_characters(8);


		putrsXLCD ("NOVEMBER");
		putsXLCD (space);
		WriteDataXLCD(hotel);
		Delay1KTCYx(13);
		Wait_For_Enter();
		WriteCmdXLCD(CLEAR_DISP);



		putrsXLCD("Data Correct?");
		skip_characters(13);
		putrsXLCD("* = Yes   D = No");
		review_done = review_input ();

		if (review == 'D')				//if the data is incorrect, the user must re-enter all the data
		main();

	}
	data_output[0] = 'X';
	data_output[1] = 'X';
	data_output[2] = 'X';
	data_output[3] = 'X';
	data_output[4] = 'X';
	data_output[5] = 'X';
	data_output[6] = 'X';
	data_output[7] = 'X';
	data_output[8] = 'X';
	data_output[9] = 'X';				//construct the transmission array
	data_output[10] = 'X';
	data_output[11] = 'X';
	data_output[12] = 'X';
	data_output[13] = 'X';
	data_output[14] = 'X';
	data_output[15] = 'p';
	data_output[16] = 'q';
	data_output[17] = Latitude[0];
	data_output[18] = Latitude[1];
	data_output[19] = Latitude[2];
	data_output[20] = Latitude[3];
	data_output[21] = Latitude[4];
	data_output[22] = Latitude[5];
	data_output[23] = Latitude[6];
	data_output[24] = Latitude[7];
	data_output[25] = Latitude[8];
	data_output[26] = 'r';
	data_output[27] = Longitude[0];
	data_output[28] = Longitude[1];
	data_output[29] = Longitude[2];
	data_output[30] = Longitude[3];
	data_output[31] = Longitude[4];
	data_output[32] = Longitude[5];
	data_output[33] = Longitude[6];
	data_output[34] = Longitude[7];
	data_output[35] = Longitude[8];
	data_output[36] = 's';
	data_output[37] = Elevation[0];
	data_output[38] = Elevation[1];
	data_output[39] = Elevation[2];
	data_output[40] = Elevation[3];
	data_output[41] = 'a';
	data_output[42] = alpha;
	data_output[43] = 'b';
	data_output[44] = bravo;
	data_output[45] = 'c';
	data_output[46] = charlie;
	data_output[47] = 'd';
	data_output[48] = delta;
	data_output[49] = 'e';
	data_output[50] = echo;
	data_output[51] = 'f';
	data_output[52] = foxtrot;
	data_output[53] = 'g';
	data_output[54] = golf;
```

```
        data_output[55] = 'h';
        data_output[56] = hotel;
        data_output[57] = 'i';
        data_output[58] = india;
        data_output[59] = 'j';
        data_output[60] = juliet;
        data_output[61] = 'k';
        data_output[62] = kilo;
        data_output[63] = 'l';
        data_output[64] = lima;
        data_output[65] = 'm';
        data_output[66] = mike;
        data_output[67] = 'n';
        data_output[68] = november;
        data_output[69] = 'o';
        data_output[70] = 'o';
        data_output[71] = 'o';
        data_output[72] = 'X';
        data_output[73] = 'X';
        data_output[74] = null;
        data_output[75] = null;


        putrsXLCD("Press * to");
        skip_characters(10);
        putrsXLCD("Transmit");
        Wait_For_Enter();                       //Prompt user for to press * to initialize transmission


        Transmit (data_output);
        main();

}
```

```
/*****************************************************
FILE: KEYPAD.C
DATE: March 30, 2004
AUTHOR:  Roy Belak
This file provides all the functions needed to interface
the keypad PORT A of the microcontroller.
*****************************************************/



#include <Keypad.h>
#include <p18f452.h>
#include <delays.h>
#include <xlcd.h>


const rom char blank = (char) 0b10100000;
const rom char degree = (char) 0b11011111;
const rom char minute = (char) 0b00100111;
int tag1 = 0;
int tag2 = 0;
int tag3 = 0;




//Keypad initialization
void Initialize_Keypad (void)
{
        ADCON1 = 0x07;                  //all pins on PORT A and E are digital I/O
        TRISA = 0x30;                   //set all the lower nibble as output, upper nibble (RA4 and RA5) as input
        TRISE = 0x03;                   //set pins RE1, RE2 as input, RE3 as output
        PORTEbits.RE2 = 0;          //ensure that pin RE3 is not high
}





//function to return the character that is entered into the keypad
void Get_Keypad_String (char *buffer, int length, int data_type)
{
		char data = 'X';
		char debounced_data = 'X';
		int i = 0;
		tag1 = 0;
		tag2 = 0;
		tag3 = 0;


		while (1)
		{

			if (data_type == 1 && (i == 3 || i == 6 || i == 9) && (tag1 == 0 || tag2 == 0 || tag3 == 0))  //this case is for the latitude /
longitude
			{
				if (i == 3 && tag1 == 0)
				{
				tag1 = 1;
					WriteDataXLCD(degree);
					WriteDataXLCD(blank);
				}
				else if (i == 6 && tag2 == 0)
				{
				tag2 = 1;
					WriteDataXLCD(minute);
					WriteDataXLCD(blank);
				}

				else if (i == 9 && tag3 == 0)
				{
				tag3 = 1;
					WriteDataXLCD('"');
					WriteDataXLCD(blank);
				}
			}



			data  = locate_key();	//set data to the value of the button being pushed

			if (data != 'X')							//case data has been entered by the user
			{
				debounced_data = debounce(data);

				if (debounced_data == data)
				{
					if (data == '*' && i == length)			//case where the string has been fully entered
					{
					Delay10KTCYx(30);
					return;
					}

					else if (data == '*' && i != length)
					{}

					else if (data  == 'D' && i != 0)
					{
						if(data_type == 1 && (i == 3 || i == 6 || i == 9))
						{
							WriteCmdXLCD(SHIFT_CUR_LEFT);
							WriteCmdXLCD(SHIFT_CUR_LEFT);
							WriteDataXLCD(blank);
							WriteCmdXLCD (SHIFT_CUR_LEFT);
							if (i == 3 )
								tag1 = 0;
							else if (i == 6)
								tag2 = 0;
							else if (i == 9)
								tag3 = 0;
						}


						WriteCmdXLCD (SHIFT_CUR_LEFT);
						WriteDataXLCD(blank);
						WriteCmdXLCD (SHIFT_CUR_LEFT);
						Delay10KTCYx(30);
						i = i - 1;
					}

					else if (data == 'D' && i == 0)
```

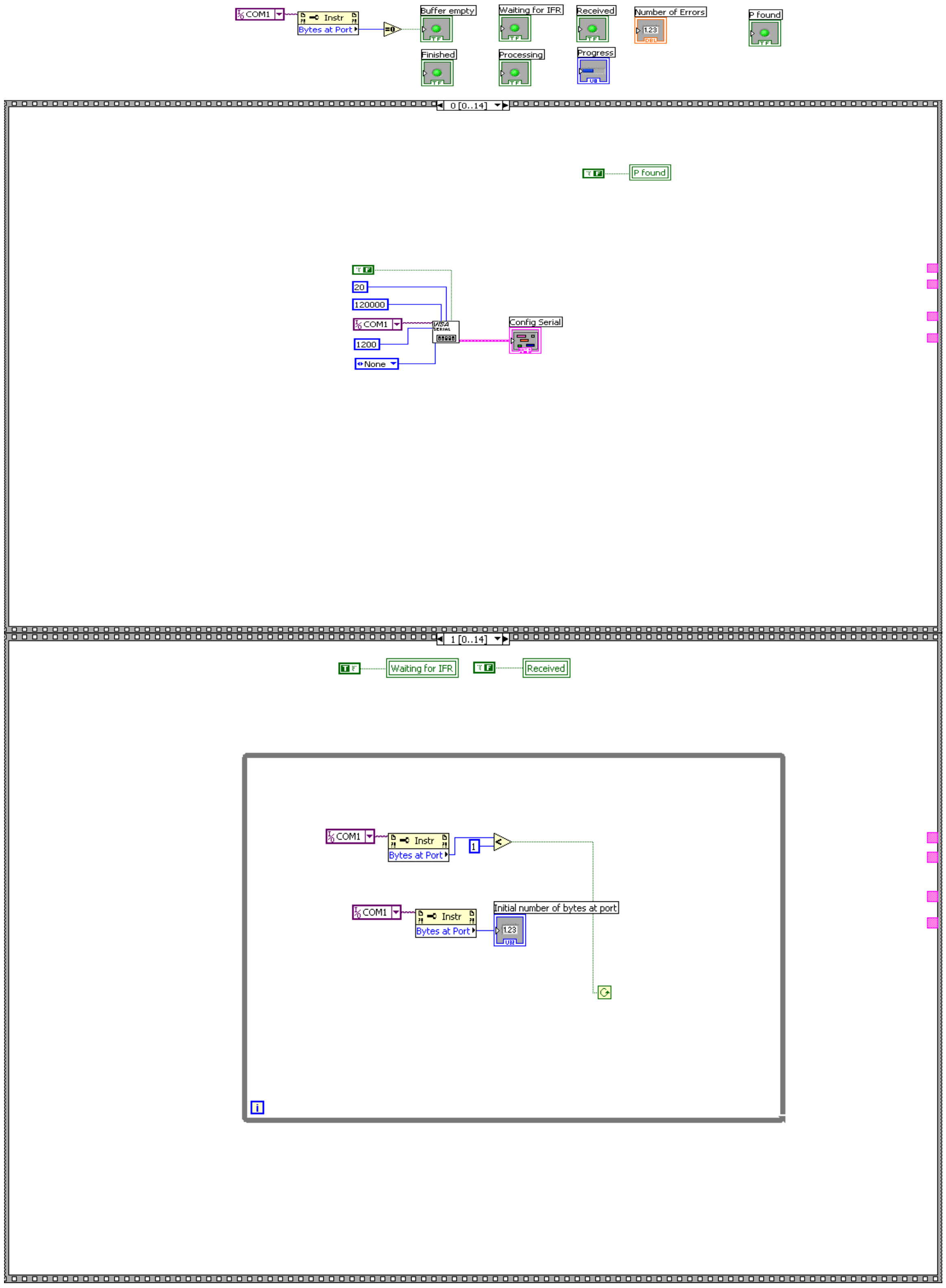
```
                                        {}


                                        else if (i == length )
                                        {}

                                        else
                                        {
                                                WriteDataXLCD(data);
                                                buffer[i] = data;
                                                Delay10KTCYx(30);
                                                i = i + 1;
                                        }
                                }
                                else
                                data = 'X';

                        }

                }
}





char Get_Next_Character(void)
{
                char data2;
                char debounced_data2;


                while (1)
                {
                        data2  = locate_key();  //set data to the value of the button being pushed

                        if (data2 != 'X')                                               //case data has been entered by the user
                        {
                                debounced_data2 = debounce(data2);

                                if (data2 == debounced_data2)                   //case where the key is confirmed
                                {
                                        Delay10KTCYx(30);
                                        return data2;
                                }

                                else
                                data2 = 'X';
                        }
                }
}





char Get_Data_Field(void)
{
                char new_char = 'X';
                char debounced_new_char = 'X';
                int i = 0;
                char data_char;

                while (1)
                {
                        new_char  = locate_key();               //set data to the value of the button being pushed

                        if (new_char != 'X')                                            //case data has been entered by the user
                        {
                                debounced_new_char = debounce(new_char);

                                if (new_char == debounced_new_char)                     //case where the key is confirmed
                                {
                                        if (new_char == '*' && i == 1 )
                                        {
                                                Delay10KTCYx(30);
                                                return data_char;
                                        }
```

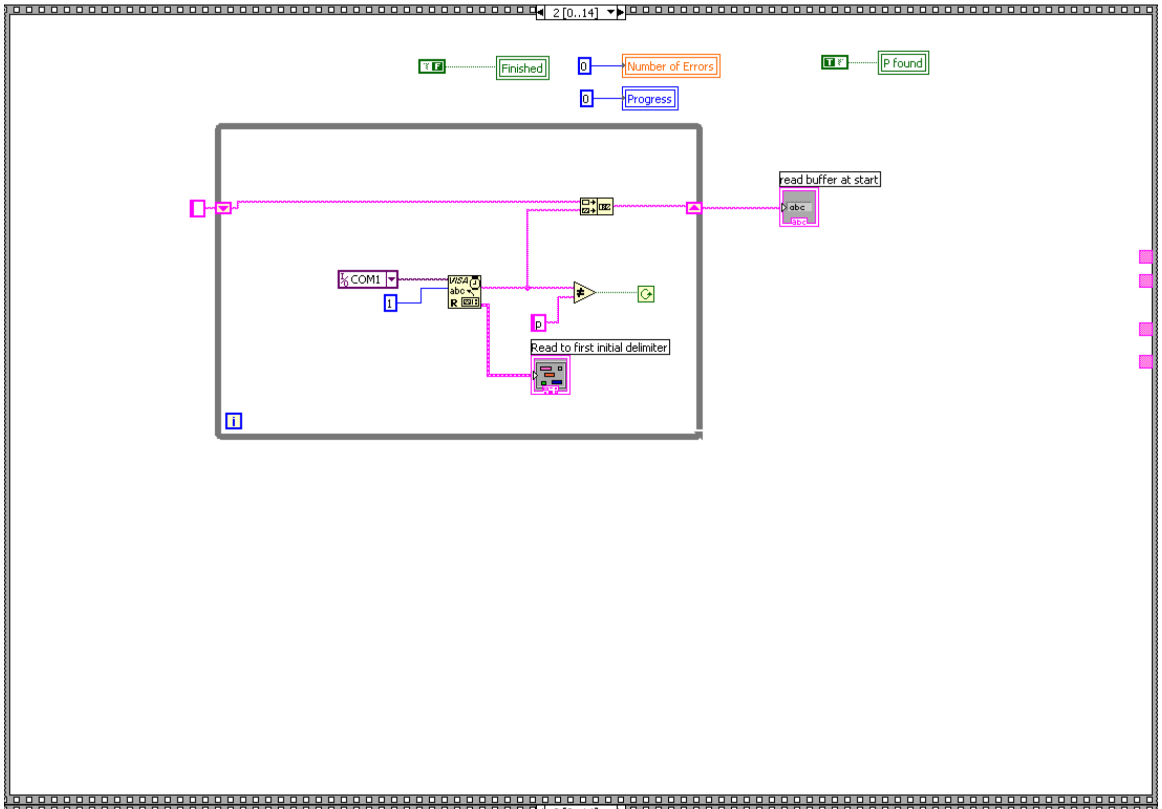
```
                                                else if (new_char == '*' && i == 0)
                                                {
                                                }


                                                else if (new_char == 'D' && i == 0)
                                                {
                                                }

                                                else if (new_char  == 'D' && i == 1)
                                                {
                                                                i = 0;
                                                                WriteCmdXLCD (SHIFT_CUR_LEFT);
                                                                Delay10KTCYx(30);
                                                }


                                                else
                                                {
                                                                if (i == 0)
                                                                {
                                                                i = 1;
                                                                data_char = new_char;
                                                                WriteDataXLCD(data_char);
                                                                Delay10KTCYx(30);
                                                                }

                                                                else
                                                                {}
                                                }
                                }

                        }
                }
}




//function to locate the neccessary key - lookup table
char locate_key (void)
{



                PORTA = 0x01;                           //raise the first row to 5 V
                if ( Pin5 == 1 && Pin6 == 0 && Pin7 == 0 && Pin8 == 0 )
                {
                                PORTA = 0;
                                return          '1';
                }
                if ( Pin5 == 0 && Pin6 == 1 && Pin7 == 0 && Pin8 == 0 )
                {
                                PORTA = 0;
                                return          '2';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 1 && Pin8 == 0 )
                {
                                PORTA = 0;
                                return          '3';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 0 && Pin8 == 1 )
                {
                                PORTA = 0;
                                return          'A';
                }



                PORTA = 0x02;                           //raise the second row to 5 V
                if ( Pin5 == 1 && Pin6 == 0 && Pin7 == 0 && Pin8 == 0 )
                {
                                PORTA = 0;
                                return          '4';
                }
                if ( Pin5 == 0 && Pin6 == 1 && Pin7 == 0 && Pin8 == 0 )
                {
                                PORTA = 0;
                        return          '5';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 1 && Pin8 == 0 )
                {
                        PORTA = 0;
                        return          '6';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 0 && Pin8 == 1 )
                {
                        PORTA = 0;
                        return          'B';
                }



                PORTA = 0x04;                           //raise the third row to 5 V
                if ( Pin5 == 1 && Pin6 == 0 && Pin7 == 0 && Pin8 == 0 )
                {
                        PORTA = 0;
                        return          '7';
                }
                if ( Pin5 == 0 && Pin6 == 1 && Pin7 == 0 && Pin8 == 0 )
                {
                        PORTA = 0;
                        return          '8';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 1 && Pin8 == 0 )
                {
                        PORTA = 0;
                        return          '9';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 0 && Pin8 == 1 )
                {
                        PORTA = 0;
                        return          'C';
                }

                PORTA = 0x08;                           //raise the fourth row to 5 V
                if ( Pin5 == 1 && Pin6 == 0 && Pin7 == 0 && Pin8 == 0 )
                {
                        PORTA = 0;
                        return          '*';
                }
                if ( Pin5 == 0 && Pin6 == 1 && Pin7 == 0 && Pin8 == 0 )
                {
                        PORTA = 0;
                        return          '0';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 1 && Pin8 == 0 )
                {
                        PORTA = 0;
                        return          '#';
                }
                if ( Pin5 == 0 && Pin6 == 0 && Pin7 == 0 && Pin8 == 1 )
                {
                        PORTA = 0;
                        return          'D';
                }


                else            //case no key found , or an illegal combination
                {
                        PORTA = 0;
                        return 'X';
                }
}





//primative debouncing function
char debounce(char temp_char)
{

                Delay1KTCYx(25);        //25 ms delay to allow key to debounce

                temp_char = locate_key();
```

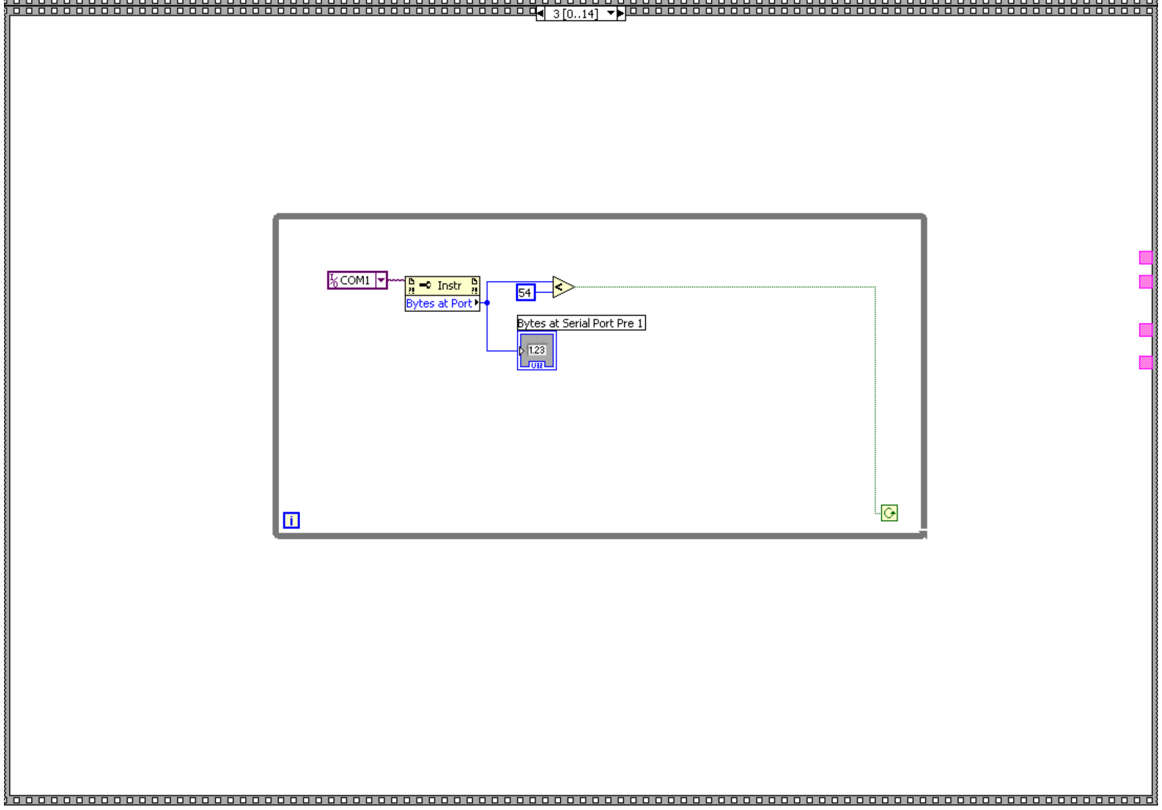
```
		return temp_char;
}
```

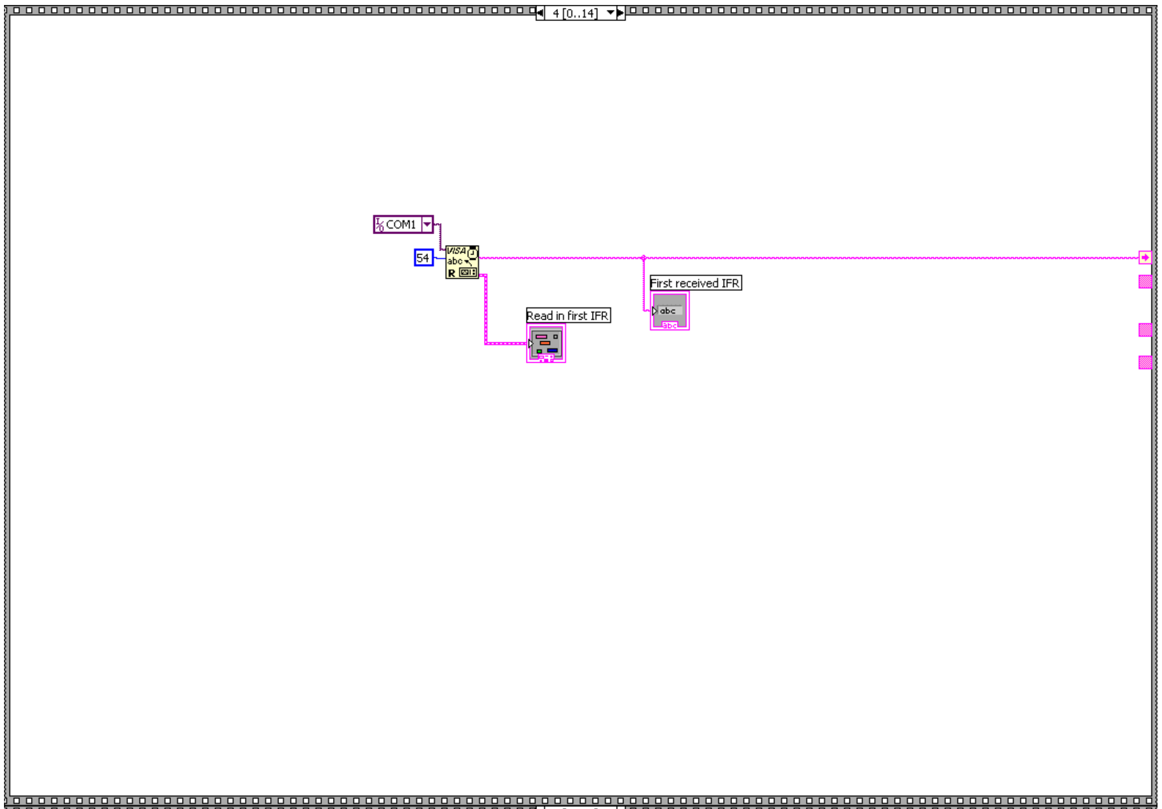
```
/*****************************************************
FILE: radio_functions.c
DATE: March 30, 2004
AUTHOR:  Roy Belak
This file provides all the peripheral functions needed for
radio2.c
*****************************************************/



#include "radio.h"
#include "keypad.h"
#include <xlcd.h>
#include <float.h> //for floating point operations
#include <math.h> //for math operations: floor
#include <usart.h> //for serial communications
#include <delays.h> //for dealy functions (0.1us per instruction @ 40MHz)
#include <stdlib.h> //data conversion for debugging


char control = 'X';
char control2 = 'X';




void Initialize_Ports (void)
{
		INTCONbits.GIEH = 0; //Disable all interupts
		ADCON1 = 0x07;		//Set port A for digital output


		TRISC = 0x00;		//Set Port C for output
		PORTCbits.RC4 = 0;  //Ensure that Transmission Relay is closed

		TRISD = 0x00;	//intitialize all pins on port D as output
		PORTDbits.RD2 = 1;  //turn on the LCD


}




void Initialize_LCD (void)
{
		OpenXLCD( EIGHT_BIT & LINES_5X7 );
}




void Initialize_USART(void)
{
		OpenUSART( USART_TX_INT_OFF &
					USART_RX_INT_OFF &
					USART_ASYNCH_MODE &
					USART_EIGHT_BIT &
					USART_CONT_RX &
					USART_BRGH_HIGH,
					207 );
}




//Functions for the operation of the LCD
void DelayFor18TCY(void)
{
		Delay10TCYx(4); //Delay 20 Instruction Cycles
}

void DelayPORXLCD(void)
{
		Delay1KTCYx(20); //20 ms delay (20 for 4 MHZ)
}

void DelayXLCD(void)
{
		Delay1KTCYx(10); //10 ms delay (10 for 4 MHZ)
}




void Wait_For_Enter(void)
{
		control = 'X';
		while (control != '*')						//case when desired character has not been pushed
		{
				control = Get_Next_Character ();
		}
		WriteCmdXLCD(CLEAR_DISP);
		Delay1KTCYx(10);
}




char review_input()
{
		control2 = 'X';
		while (control2 != '*'  && control2 != 'D')					//case when desired character has not been
pushed
		{
				control2 = Get_Next_Character ();
		}
		WriteCmdXLCD(CLEAR_DISP);
		Delay1KTCYx(10);


		if (control2 == '*')
		return '*';

		else if (control2 == 'D')
		return 'D';
}



void Transmit(char *data_buffer)
{
		PORTDbits.RD2 = 0;	//Turn off the power to the screen.
		PORTDbits.RD7 = 0;	//Turn off R/W bit on LCD
		PORTDbits.RD6 = 0;	//Turn off E bit on LCD
		PORTDbits.RD5 = 0;	//Turn off R/S bit on LCD
		PORTB = 0;		//turn off the data bits to the LCD

  Delay10KTCYx(10);
		PORTCbits.RC4 = 1;		//Control signal to activate relay
  PORTCbits.RC4 = 1;

		Delay10KTCYx(100);		//600 millisecond wait prior to transmission	This should be increased if the signal
									//is clipped at the beggining
		putsUSART(data_buffer);

		Delay10KTCYx(10);

		putsUSART(data_buffer);

		Delay10KTCYx(10);

		PORTCbits.RC4 = 0;  //Close transmission relay
		PORTDbits.RD2 = 1;  //Power up Screen
		Initialize_LCD();		//Re-initialize the LCD
  Delay10KTCYx(10);
  Initialize_LCD();
}
```

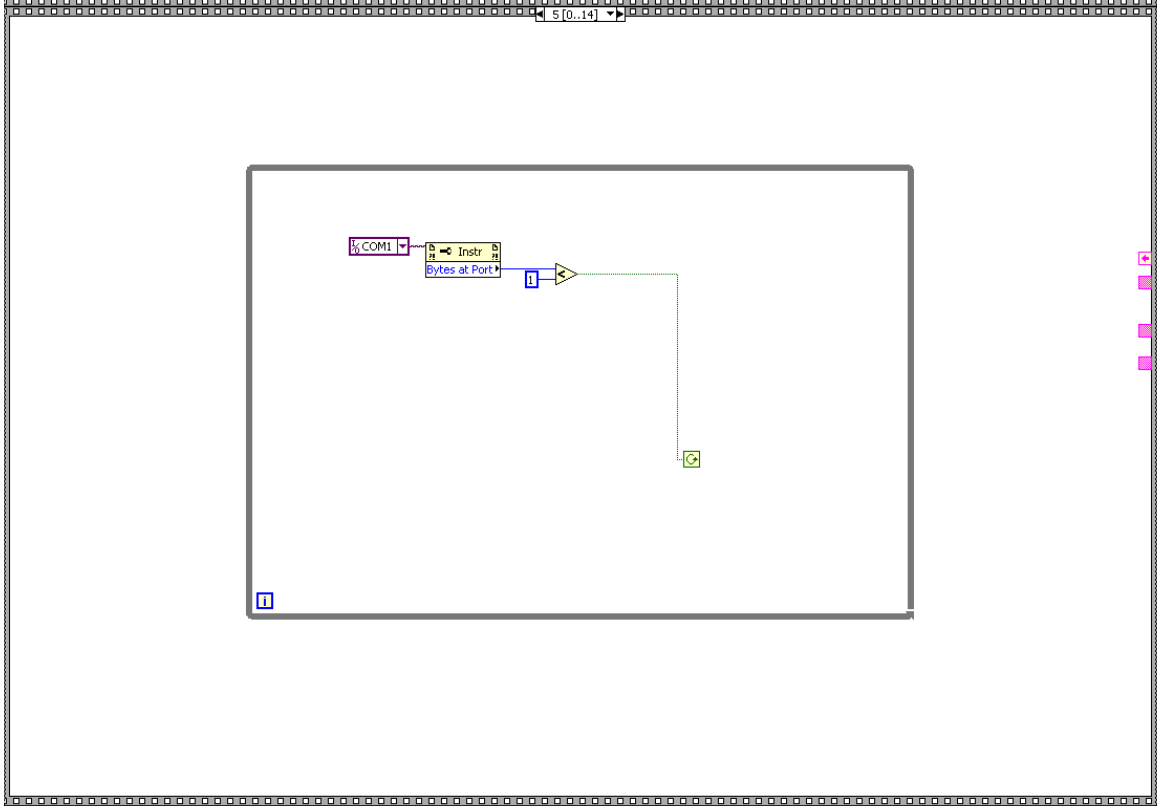
Appendix G – Reception Module Source Code

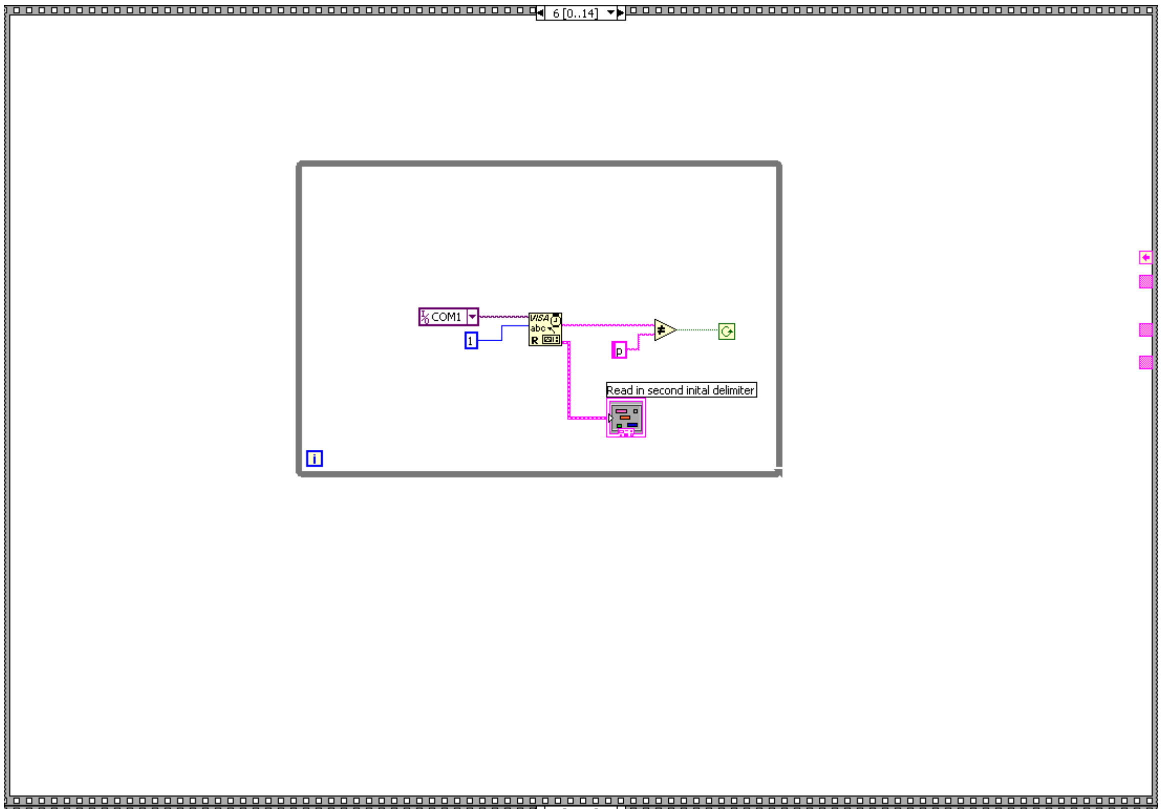


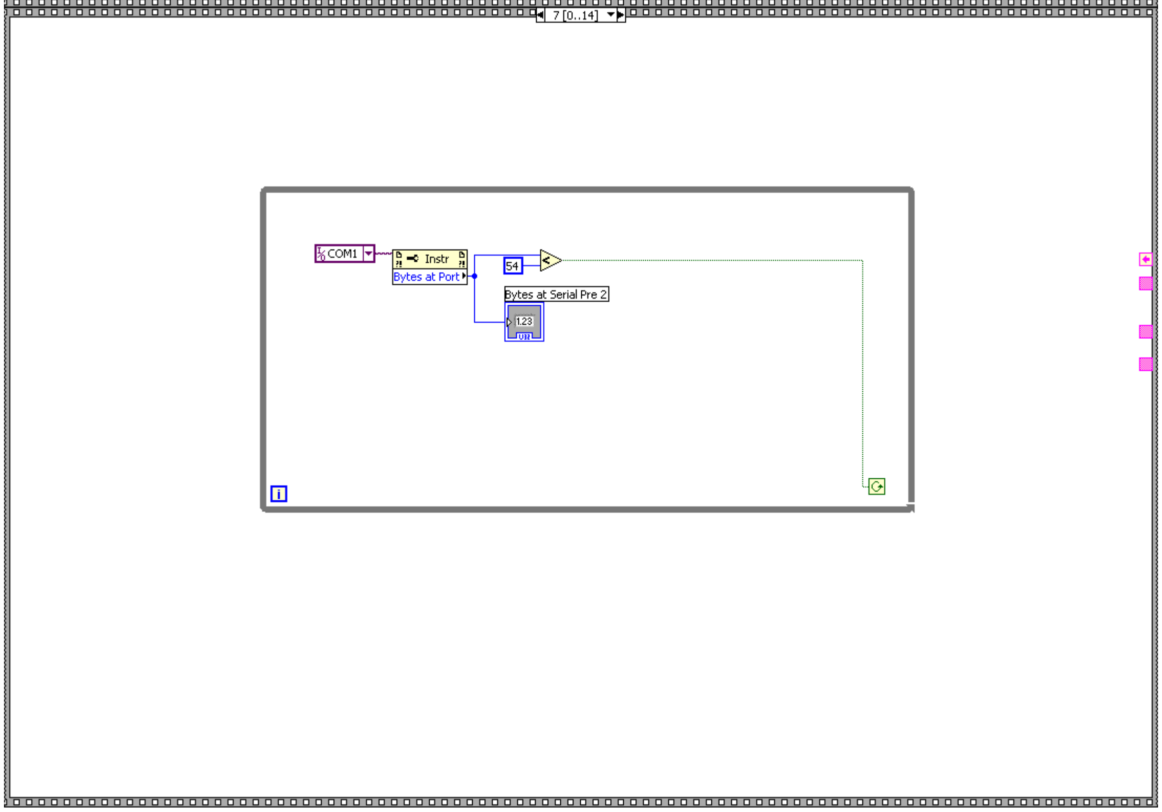

2 [0..14]
Finished
Number of Errors
P found
Progress
0
0
read buffer at start
COM1
1
p
Read to first initial delimiter
i

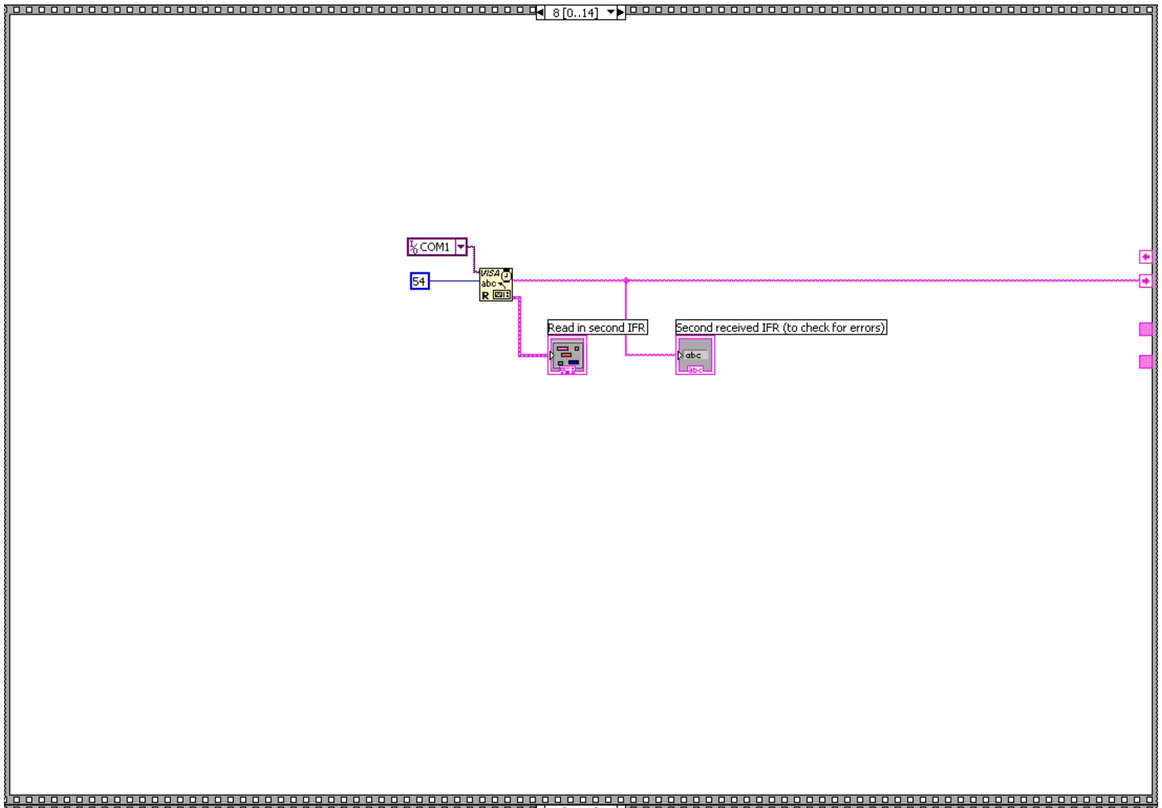

3 [0..14]
COM1
Instr
Bytes at Port
54
Bytes at Serial Port Pre 1
i

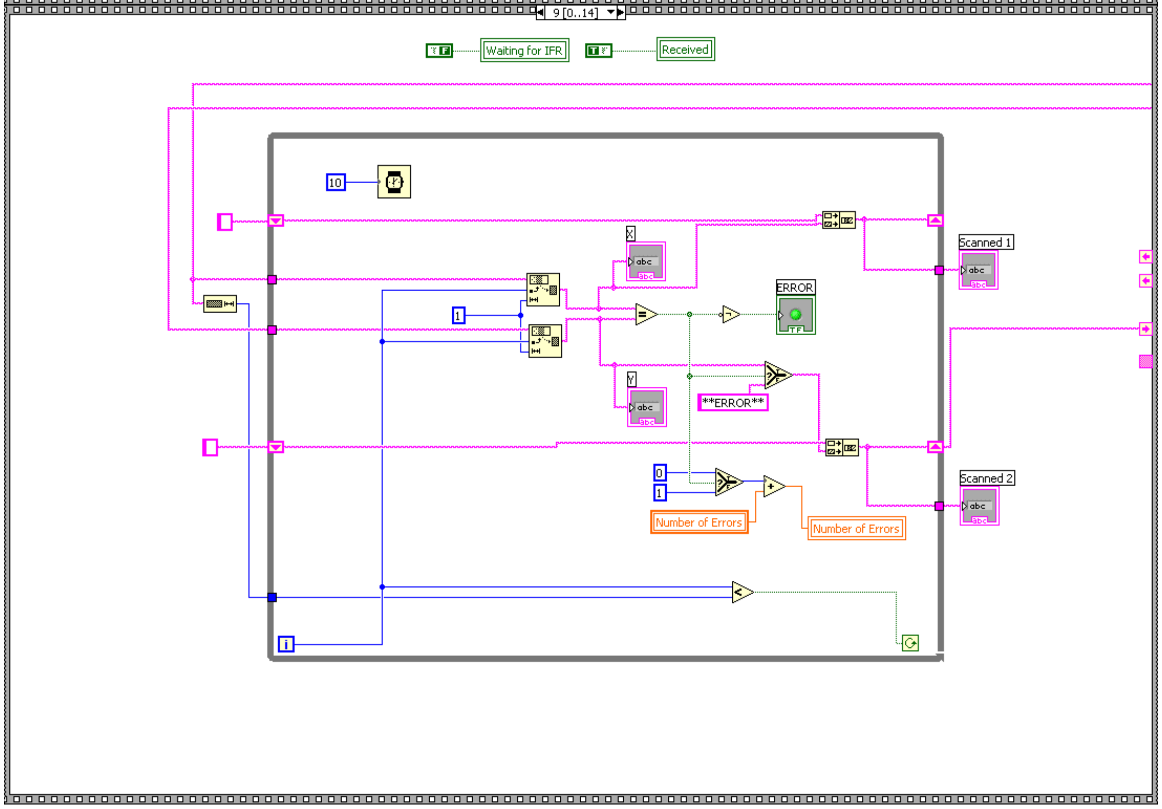

4 [0..14]
COM1
54
VISA abc R
First received IFR
Read in first IFR
abc

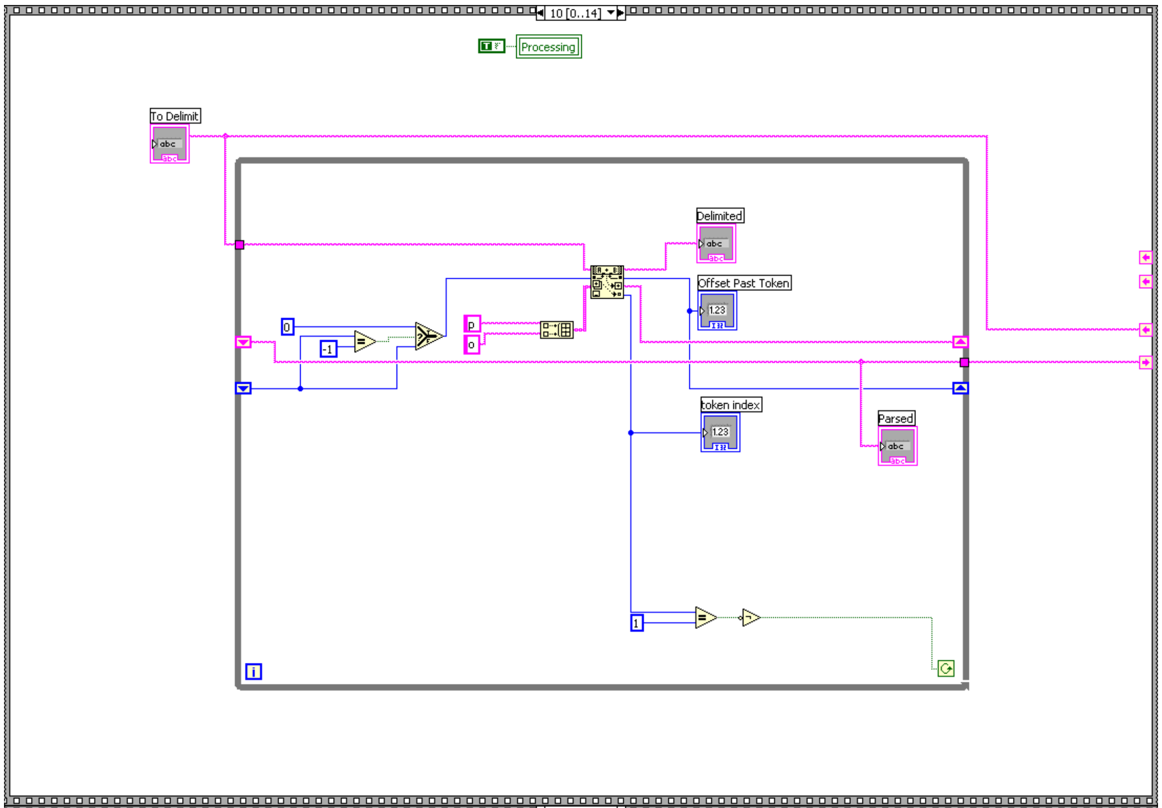

5 [0..14]
COM1
Instr
Bytes at Port
1
<
i


6 [0..14]
COM1
1
p
Read in second inital delimiter

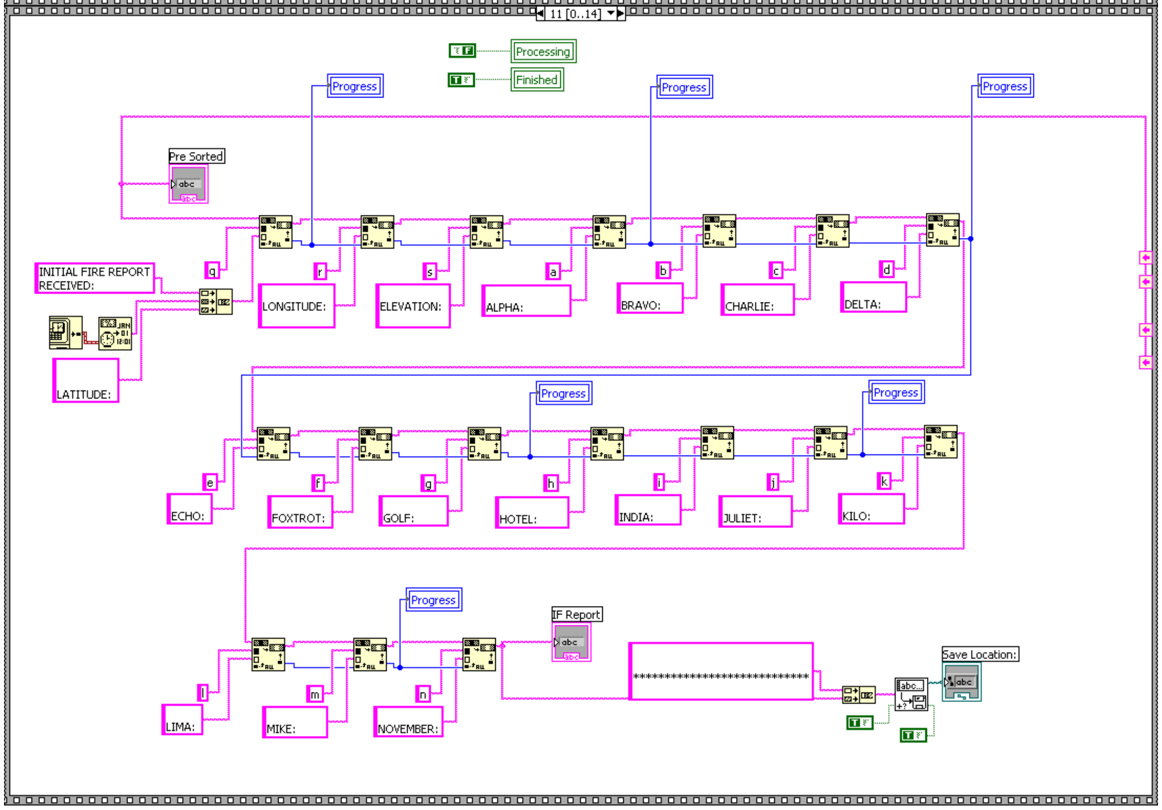

7 [0..14]
COM1
Instr
Bytes at Port
54
Bytes at Serial Pre 2
123


8 [0..14]
COM1
54
Read in second IFR
Second received IFR (to check for errors)

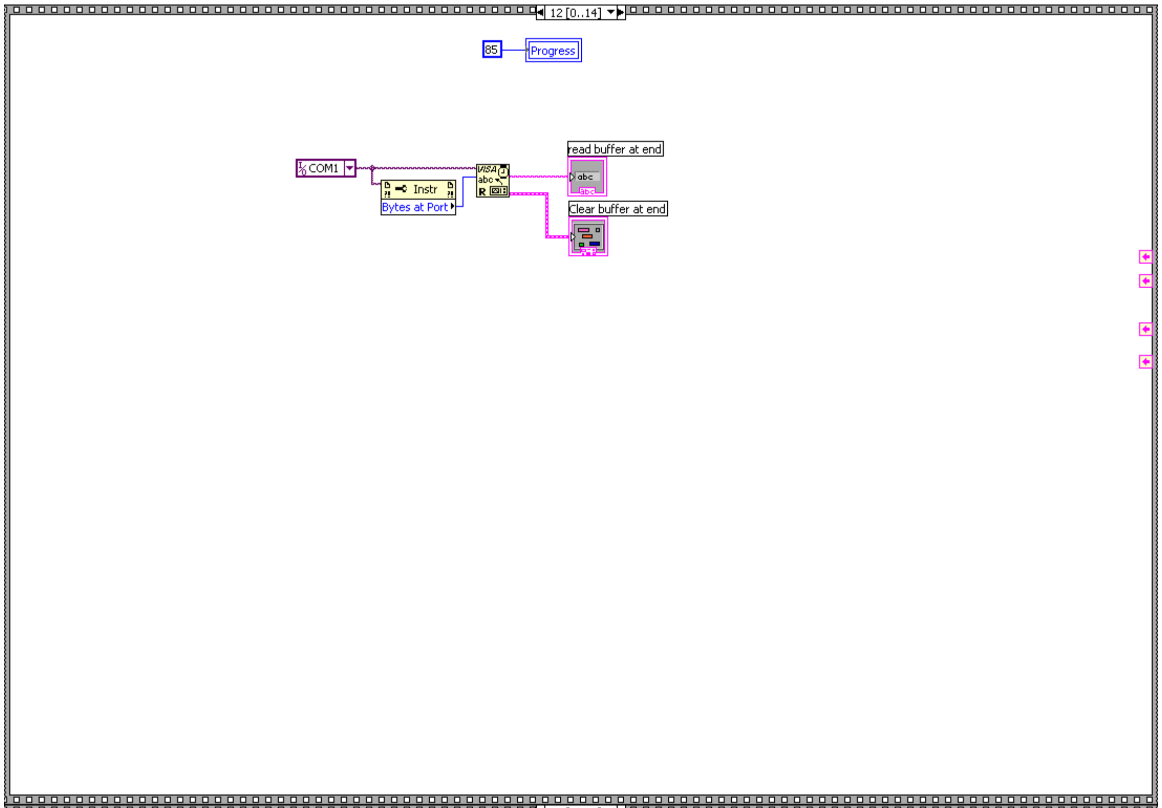

9 [0..14]
Waiting for IFR
Received
10
X
ERROR
1
Y
ERROR
0
1
Number of Errors
Number of Errors
Scanned 1
Scanned 2
i

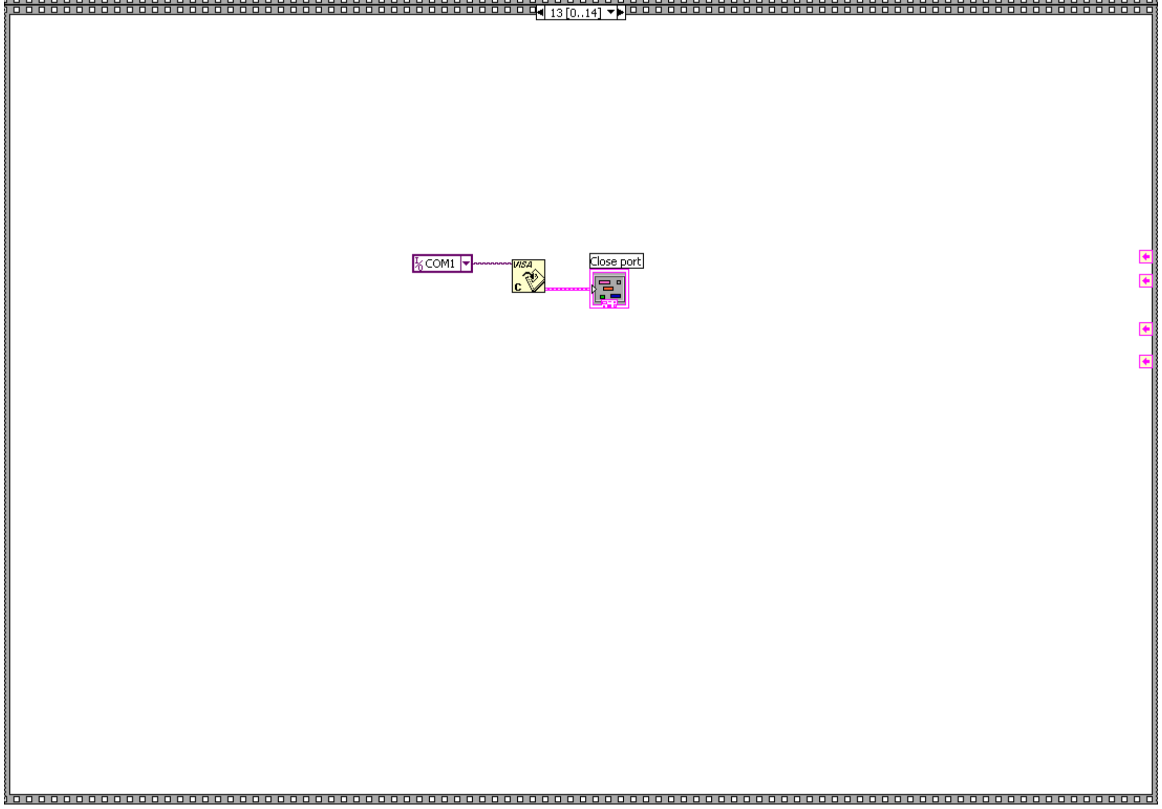


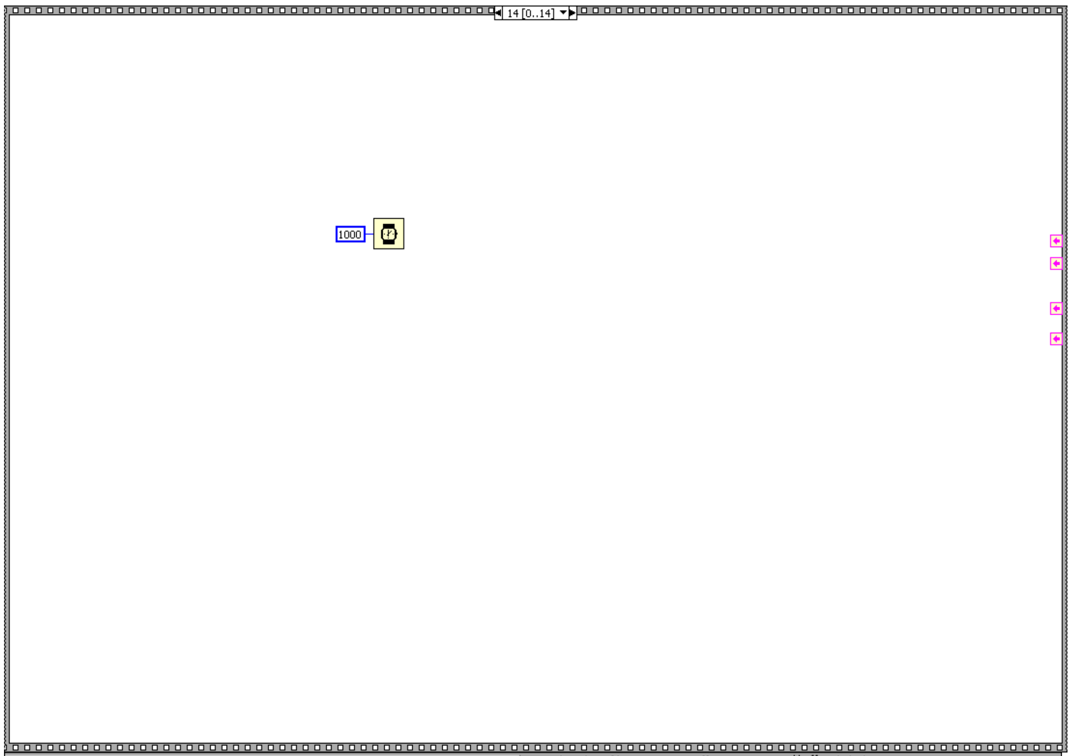
10 [0..14]
Processing
To Delimit
Delimited
Offset Past Token
token index
Parsed
0
-1
p
o
1

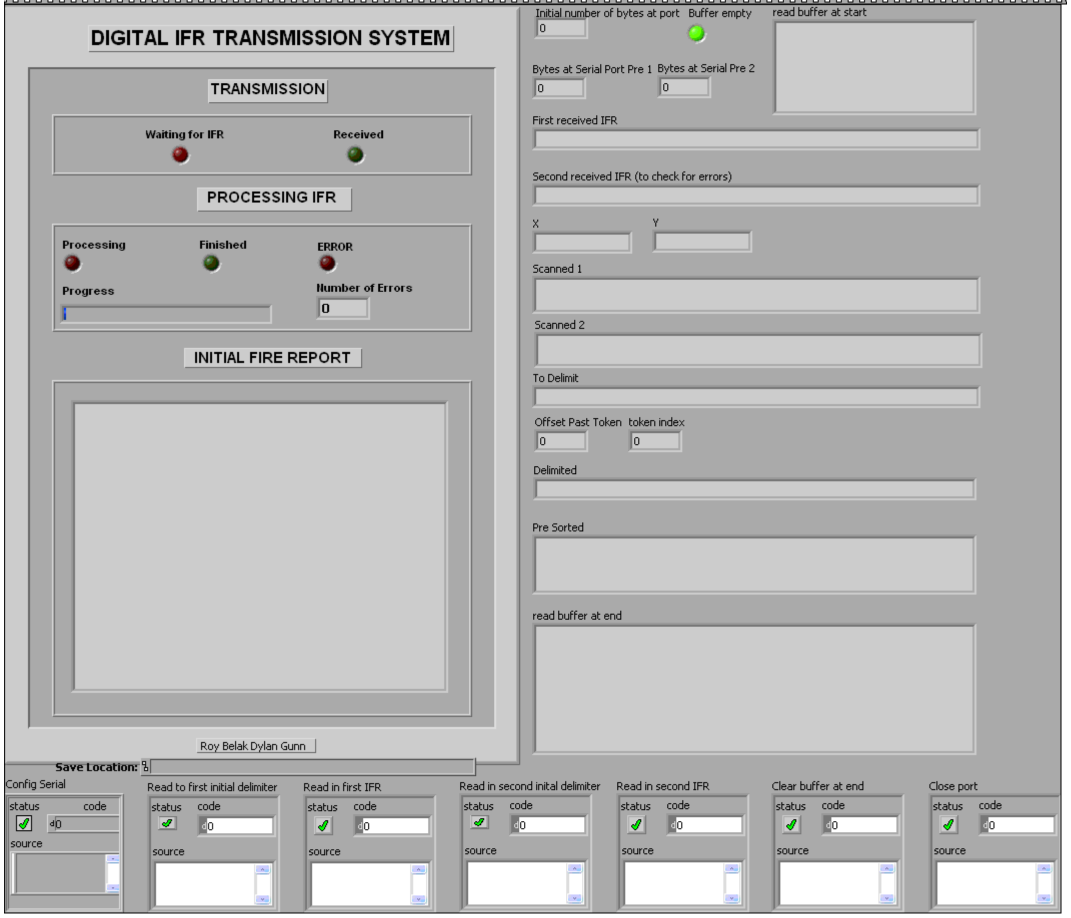


11 [0..14]
Processing
Finished
Progress
Pre Sorted
INITIAL FIRE REPORT RECEIVED:
LATITUDE:
LONGITUDE:
ELEVATION:
ALPHA:
BRAVO:
CHARLIE:
DELTA:
ECHO:
FOXTROT:
GOLF:
HOTEL:
INDIA:
JULIET:
KILO:
LIMA:
MIKE:
NOVEMBER:
IF Report
Save Location:
****


12 [0..14]
85
Progress
COM1
Instr
Bytes at Port
VISA
read buffer at end
Clear buffer at end


13 [0..14]
COM1
VISA
Close port




DIGITAL IFR TRANSMISSION SYSTEM

TRANSMISSION

Waiting for IFR ●
Received ●

PROCESSING IFR

Processing ●
Finished ●
ERROR ●
Progress
Number of Errors: 0

INITIAL FIRE REPORT

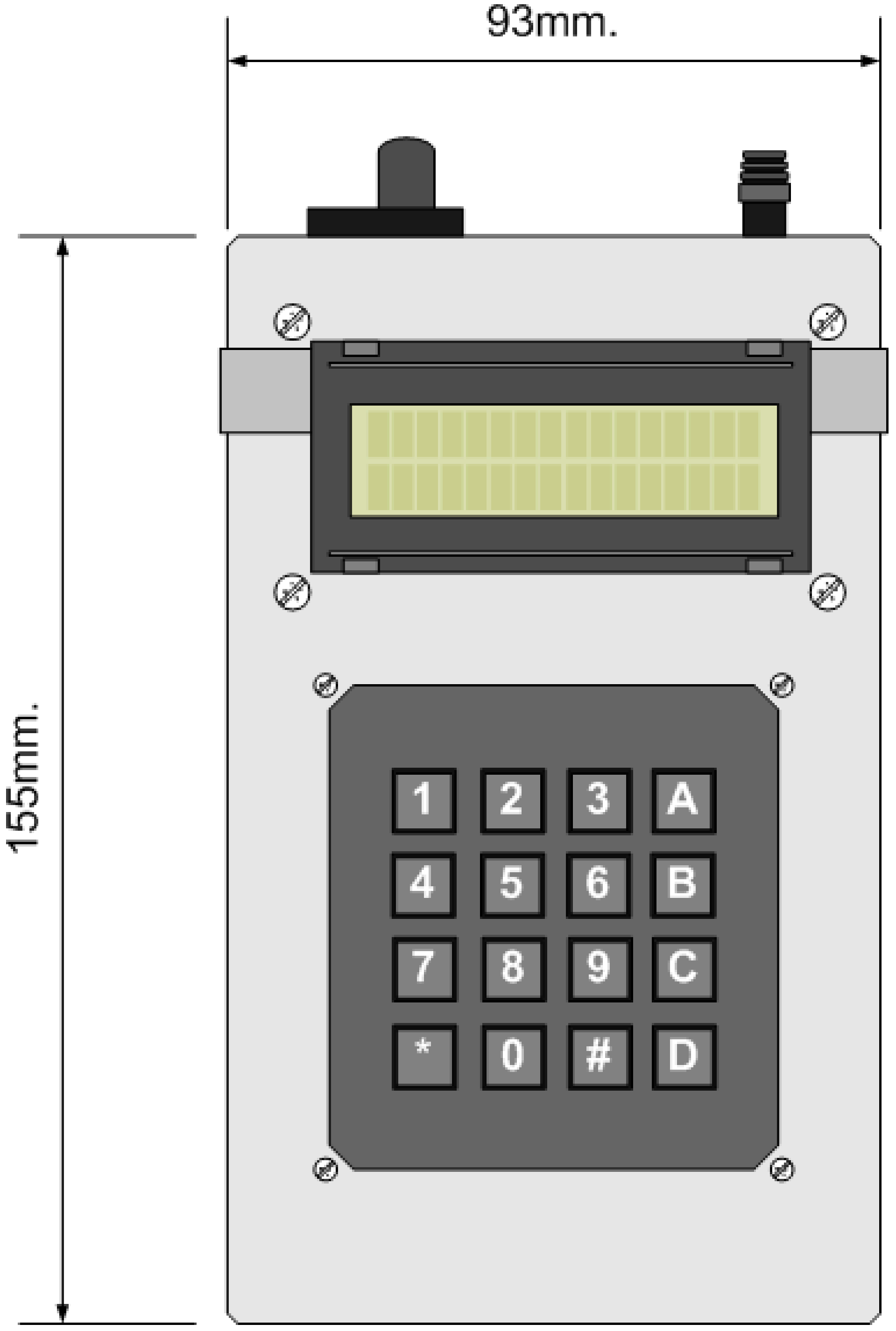
Roy Belak Dylan Gunn

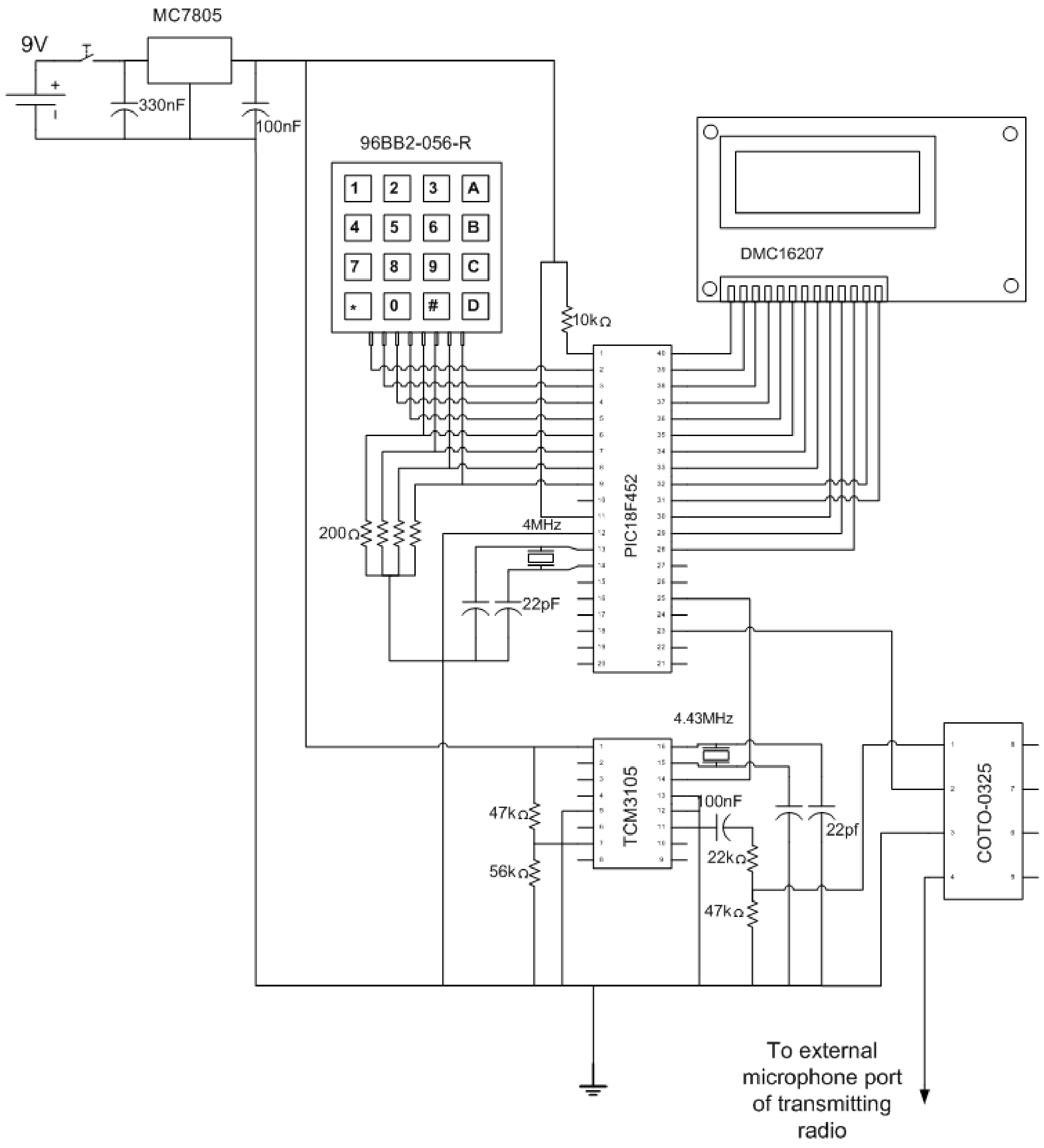
Save Location:

Initial number of bytes at port: 0
Buffer empty ●
read buffer at start
Bytes at Serial Port Pre 1: 0
Bytes at Serial Pre 2: 0
First received IFR
Second received IFR (to check for errors)
X
Y
Scanned 1
Scanned 2
To Delimit
Offset Past Token: 0
token index: 0
Delimited
Pre Sorted
read buffer at end

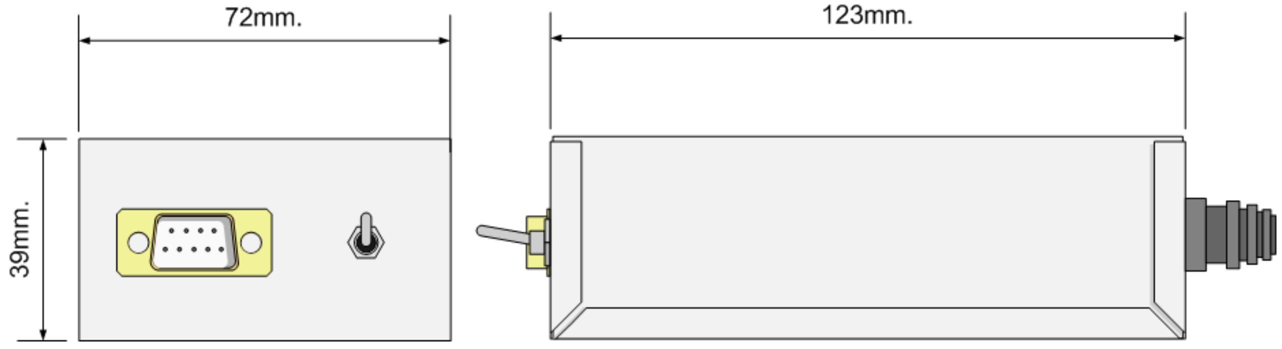
	Config Serial	Read to first initial delimiter	Read in first IFR	Read in second inital delimiter	Read in second IFR	Clear buffer at end	Close port
status	✓	✓	✓	✓	✓	✓	✓
code	0	0	0	0	0	0	0
source							

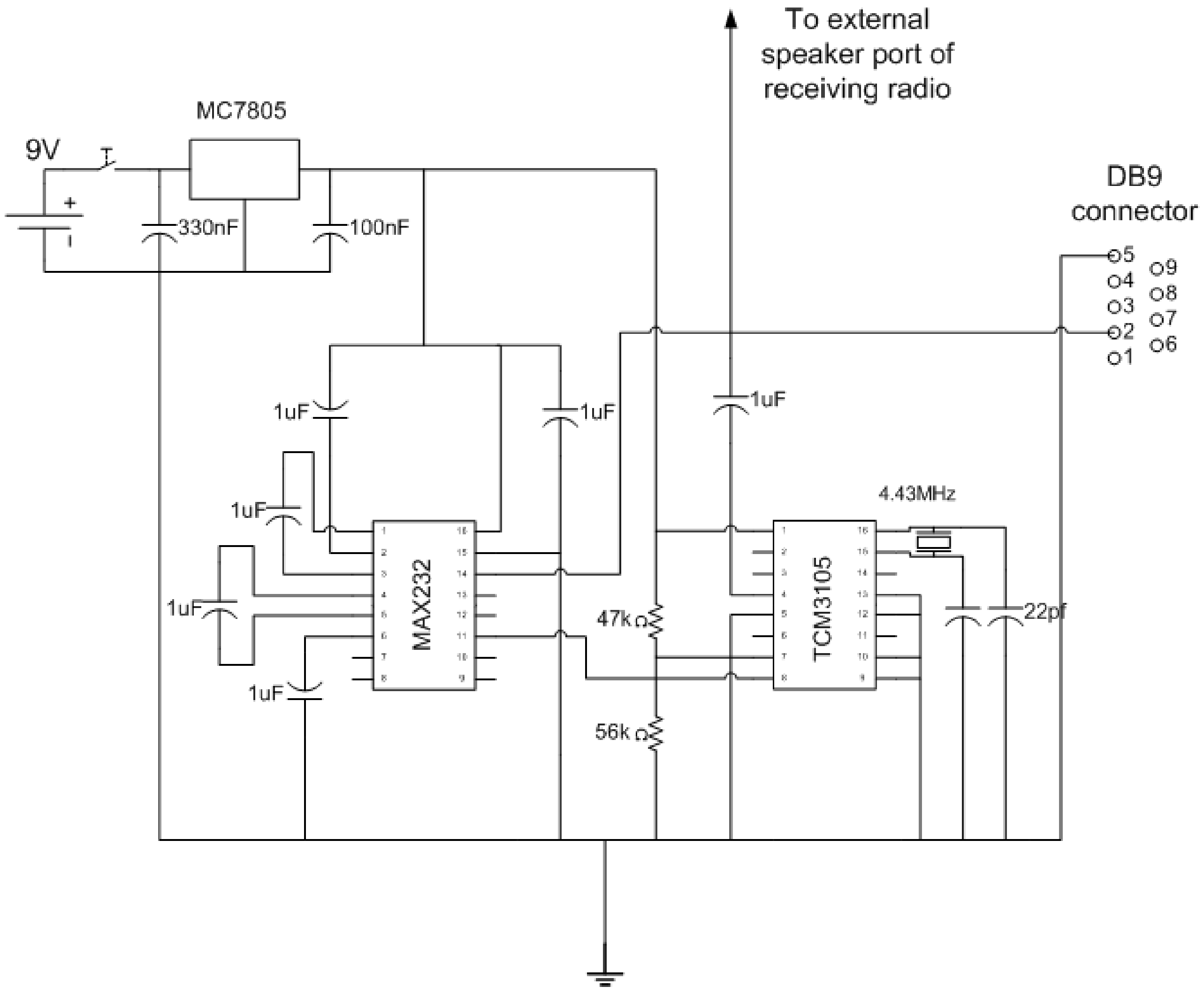
Appendix H – Transmission Module Schematics



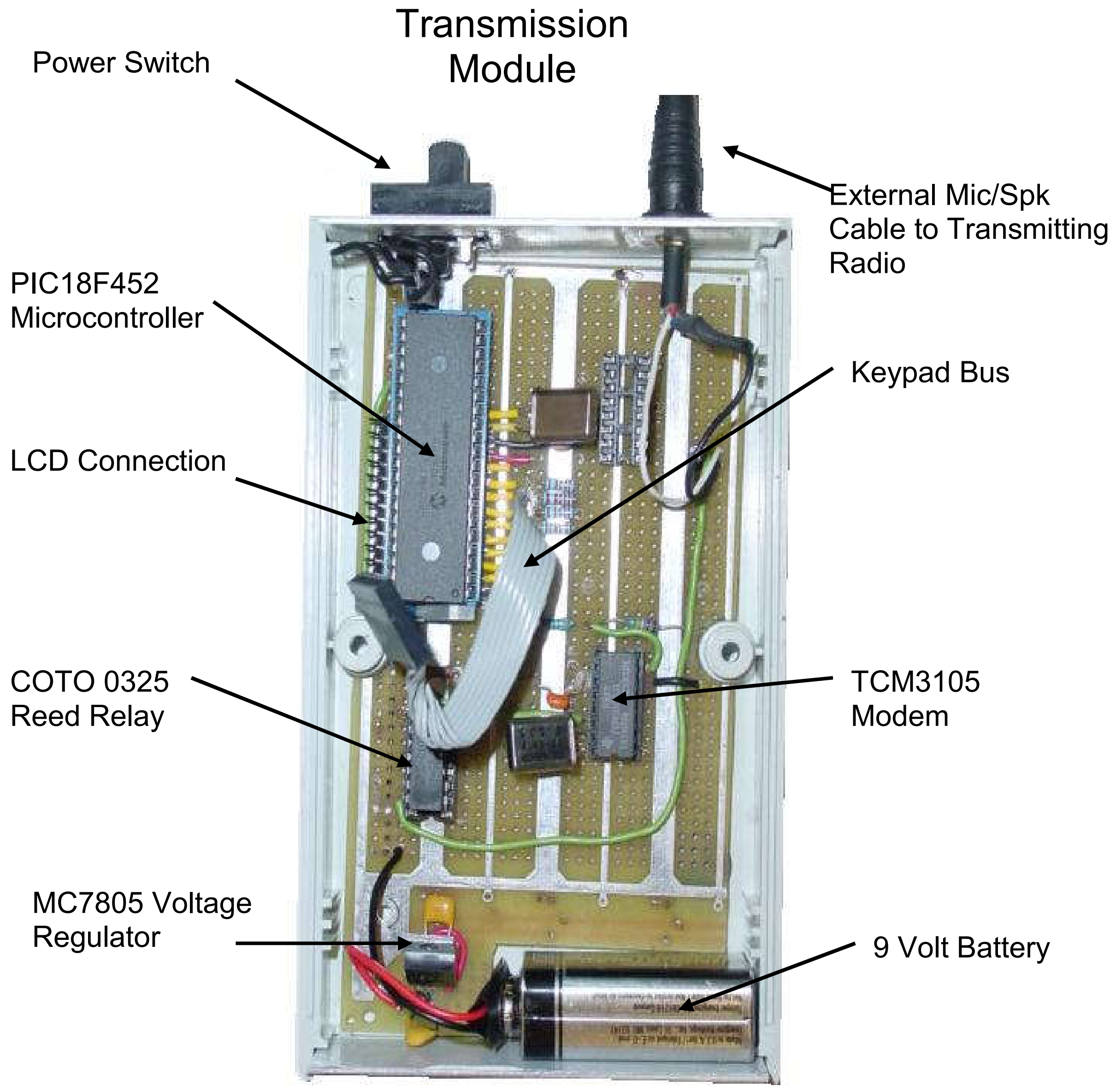

MC7805
9V
330nF
100nF
96BB2-056-R
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D
10kΩ
DMC16207
PIC18F452
200Ω
4MHz
22pF
4.43MHz
TCM3105
47kΩ
56kΩ
100nF
22kΩ
47kΩ
22pf
COTO-0325
To external microphone port of transmitting radio

Appendix I – Reception Module Schematics

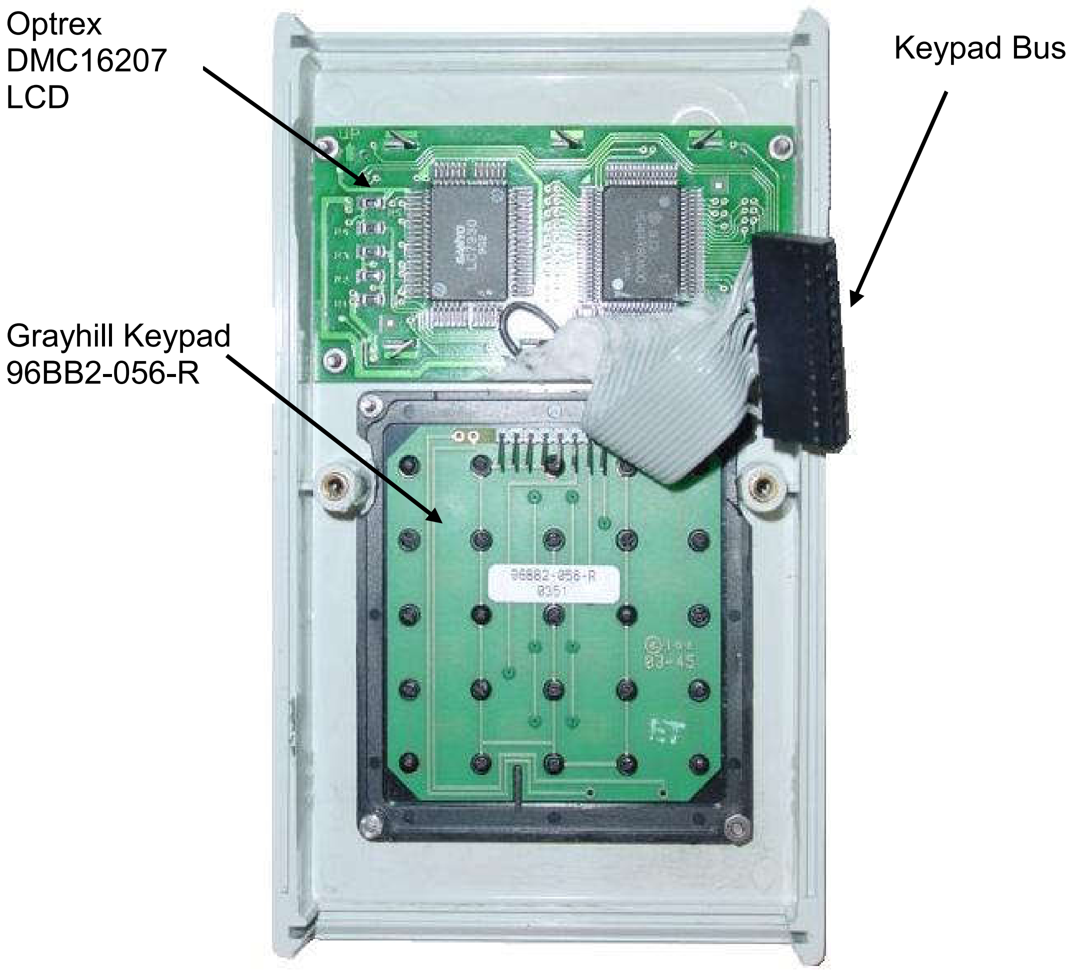


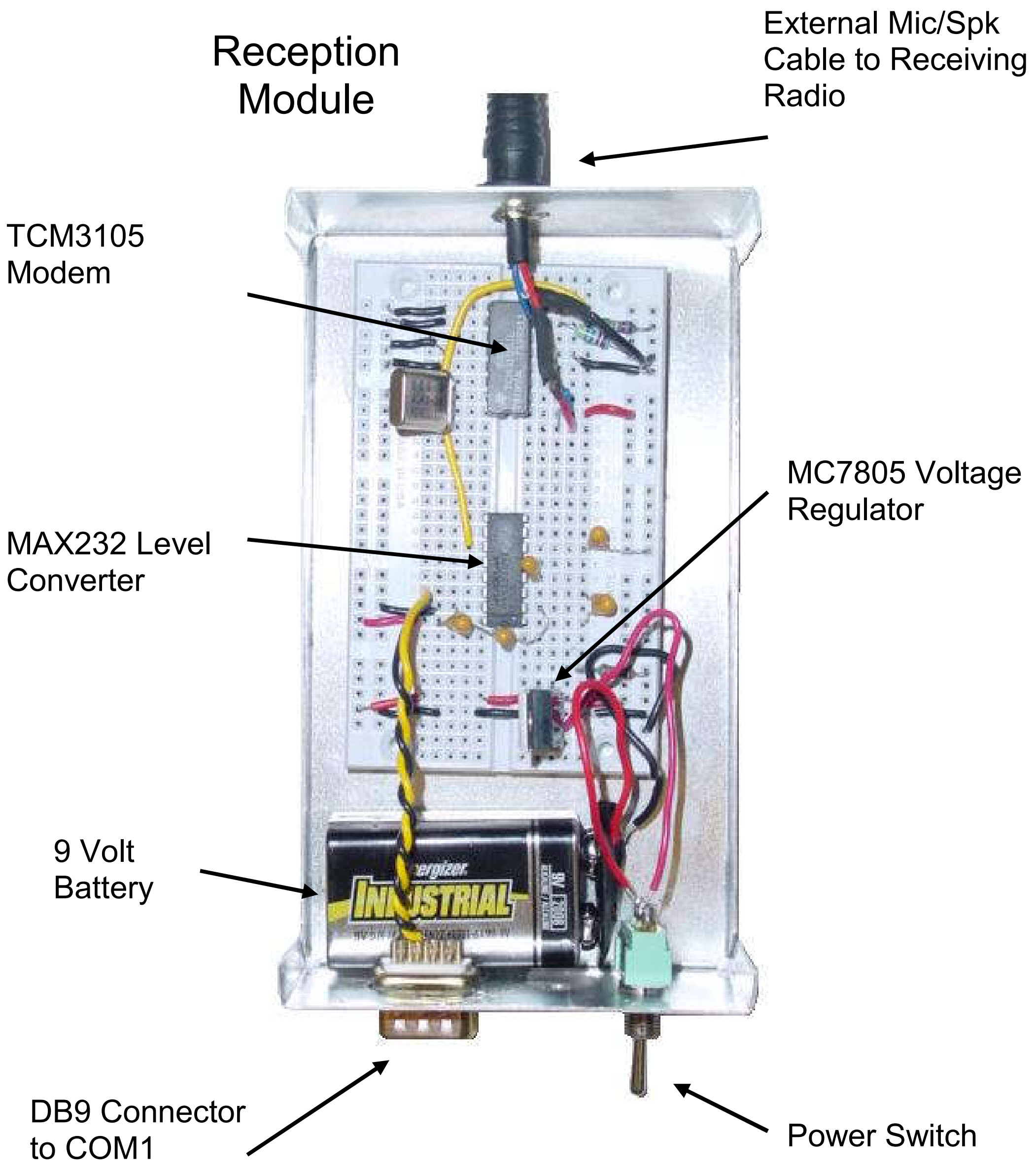


Appendix J – Photos



Transmission Module




Reception Module
External Mic/Spk Cable to Receiving Radio
TCM3105 Modem
MC7805 Voltage Regulator
MAX232 Level Converter
9 Volt Battery
DB9 Connector to COM1
Power Switch