Clarifying the Amateur Bell 202 Modem

Kenneth W. Finnegan, W6KWF
Masters Student, Electrical Engineering Department
California Polytechnic State University, San Luis Obispo
http://blog.thelifeofkenneth.com/
kennethfinnegan2007@gmail.com

Bridget Benson, Ph.D.
Assistant Professor, Electrical Engineering Department
California Polytechnic State University, San Luis Obispo
http://www.calpoly.edu/~bbenson/
bbenson@calpoly.edu

July 15, 2014

Abstract

Since its inception more than three decades ago, packet radio has seen several significant changes in technology and applications. Despite its age, the 1200 baud Bell 202 modulation used in some of the earliest packet systems still enjoys a wide user base in several of the major amateur packet networks. Oddly, despite being one of the longest-lived amateur packet modulations in use, the amateur radio community seems to have neglected to write a detailed specification document for this integral part of so many packet systems. This paper presents information gathered from network protocol specifications and amateur packet radio articles to assist other amateur radio operators implementing modems for this prototypical modulation.

Keywords: AFSK, AX.25, Bell 202, Packet Radio, Specification

1 Introduction

The primary goal of this paper is to give the reader a sufficient number of breadcrumbs to successfully build their own Amateur Bell 202 modem. The need for this document became apparent to the author while working on his master's thesis concerning the Automatic Packet Reporting System (APRS). Despite Bell 202 being the underlying modem used by nearly every VHF amateur radio packet network, there doesn't appear to be sufficient documentation for a radio amateur to implement their own modem compatible with the existing networks.

The rest of this document touches on the most significant aspects of the Amateur Bell 202 protocol, including:

- Pointing out that what amateur radio operators call “Bell 202” implicitly extends well beyond the original Bell 202. Therefore, the new term “Amateur Bell 202” is proposed to differentiate between the entire modem and the underlying modulation
- Drawing a new line between AX.25 and Amateur Bell 202 to make it clear that error detection is a concern for the modem and not the data link protocol
- Presenting a reference implementation of the checksum used to detect transmission errors in Amateur Bell 202 frames
- Discussing how the baseband modem signal should be modulated using VHF voice radios and what challenges this presents to modem performance

Despite Amateur Bell 202 often being decried for its age and vastly outliving its usefulness, it can’t be denied that it is still an integral part of the amateur radio digital communications landscape. While its deficiencies dictate that Amateur Bell 202 will rarely be the best choice for new packet radio networks, its relative simplicity causes Amateur Bell 202 to be an appealing gateway into the digital communications hobby.

2 Bell 202 in Amateur Radio

The Bell 202 protocol is an audio frequency shift keyed (AFSK) modulation that encodes data by shifting between 1200Hz and 2200Hz audio tones. These tones represent a binary one and zero respectively and transitions occur at a rate of 1200 symbols per second. Originally developed by AT&T for use on the telephone network [3], Bell 202 was popular among amateur radio operators due to the abundance of modems available on the second-hand market in 1981 when the FCC authorized amateur packet operations in the United States [7].

There isn’t a particularly clean mapping of packet radio protocols to the seven layers of the OSI

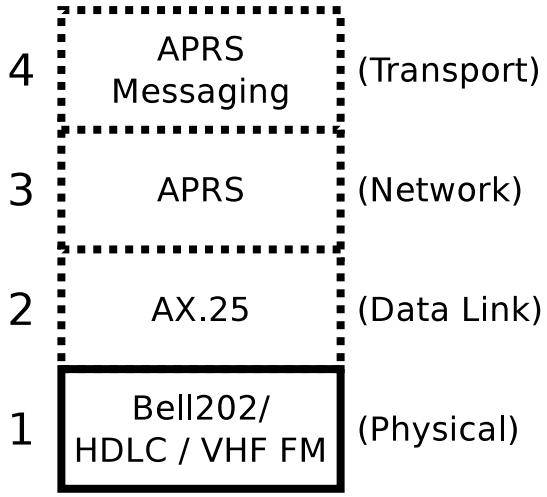

Figure 1: OSI network layers for a typical packet radio application. This paper refers to the entirety of layer 1 as “Amateur Bell 202”

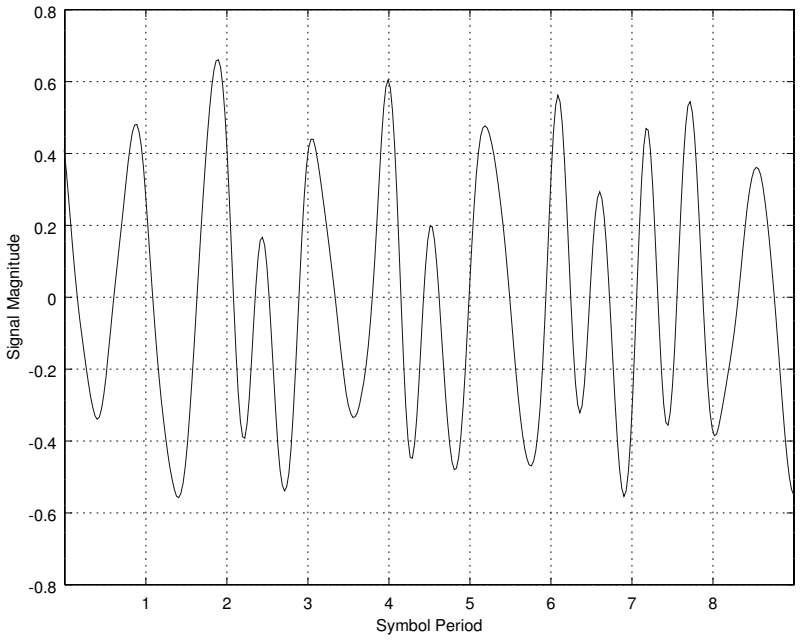

Figure 2: Amateur Bell 202 signal representing the ASCII letter A (0x41)

network model, but one can be formed to help clarify references to the different layers in this article. Figure 1 shows the example of Bell 202 as used in the Automatic Packet Reporting System (APRS) network stack. Due to Amateur radio operators using Bell 202 as the modem below AX.25, which is a derivative of the X.25 network protocol [4, §1.1], the physical layer implicitly includes the High-Level Data Link Control (HDLC) protocol for framing and bit stuffing [9]. Since using HDLC with Bell 202 modems is so implicit in amateur radio systems, the layer 1 packet protocol should be called "Amateur Bell 202," to distinguish it from the original Bell 202 developed by AT&T.

One implication of using HDLC is that frames are not encoded using the 1200Hz mark and 2200Hz space symbols of traditional Bell 202, but instead use an inverted non-return to zero (NRZI) encoding. NRZI calls for zeros in the original bit stream to be encoded as a continuous-phase frequency transition between consecutive symbols, while ones are encoded as the lack of a frequency change between two symbols [13]. Figure 2 shows a typical Amateur Bell 202 signal representing the ASCII letter "A," starting with the least significant bit.

3 Amateur Bell 202 Transmission Format

Figure 3 shows the format of a typical single-frame Amateur Bell 202 transmission. There are a number of important facets to note:

- The leading 0x00 octets are mentioned in very few documents discussing Amateur Bell 202, yet they reportedly improve modem throughput [10][12]. 0x00 encoded in NRZI causes a symbol transition every clock cycle and thus provides a more effective clock synchronization

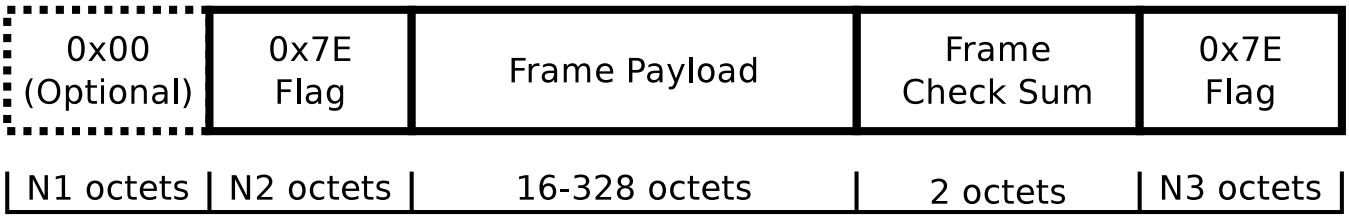

Figure 3: Amateur Bell 202/HDLC frame format

target than the originally specified 0x7E octets. 0x7E is actually the longest allowable string of 1s in the frame and therefore has the lowest amount of energy at the clocking frequency, which causes it to be the *worst* octet for asynchronous clock recovery.

- The octet 0x7E is used to indicate the beginning and end of HDLC frames. There is little guidance on the number of flag octets needed before or after frames in Amateur Bell 202 (represented by N2 and N3 in Figure 3) beyond stating that there must be at least one of each.
- The sum of N1 and N2 is variable in most modems via the "TXDelay" parameter, which specifies how long the preamble should be in 10ms increments.
- It is permissible to encode multiple frames per transmission, yet there is no guidance as to how many octets of 0x7E be included between them. Most modems insert several 0x7E octets between frames.[1] Tests indicate that many modems are sensitive to the number of flags before, between, and after frames as mentioned in the last three points. Finding definitive minimums would require testing a representative sample of the popular Amateur Bell 202 modems, which exceeds the author's resources.
- The frame payload and frame checksum must be modified such that no string of more than five 1's appear in a row. This is done by "bit-stuffing" the transmitted bitstream by appending a zero after any string of five ones at the transmitter, and subsequently dropping this zero following five ones at the receiver.[2]
- Every octet is encoded and transmitted least-significant bit first, except for the CRC-16-CCITT frame checksum, which is transmitted big-endian and most-significant bit first. [4, §3.8], [6, §8.1.1-2], [9]. See Section 3.2 for further discussion.
- The minimum and maximum payload sizes indicated in Figure 3 aren't enforced by any properties of Bell 202 or HDLC, but from the Maximum Transmission Unit (MTU) specified in the layer 2 AX.25 network protocol. Larger frames are possible and were often used in specialized AX.25 and IPv4 packet networks [11]. A practical upper limits on frame size is enforced by the need for successful frames to be completely error-free.

[1] For a specific example, the Argent Data OT3m TNC with firmware r56474 inserts 3 flags before a frame, 7 flags between two frames, and 5 flags after the final frame.

[2] Six ones in a row represent a 0x7E flag indicating the end of a frame or an idle carrier. Seven or more ones in a row indicate an invalid channel state that shouldn't happen, but regularly does, so modems must be able to handle arbitrary strings of ones gracefully.

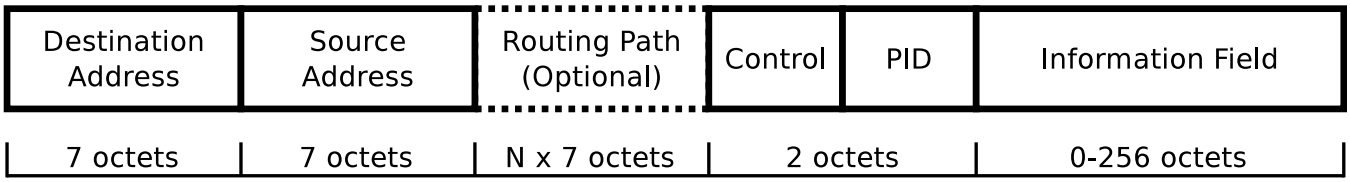

Figure 4: Modified AX.25 packet format excluding HDLC fields.

3.1 Excluding HDLC from Layer 2 AX.25

It is important to note that the presentation of the HDLC framing and checksum in Figure 3 as part of the layer 1 modulation instead of as part of the layer 2 AX.25 frame is novel to this work and hasn't been seen in any of the existing literature. This change was made because including the frame checksum and flags in the layer 2 documentation confuses the separation between Amateur Bell 202 and AX.25.

This is particularly important when AX.25 packets are transported across other layer 1 links which use their own framing protocols. The Keep It Simple Stupid (KISS) serial link between a host system and modem is the most notable transport where the HDLC fields are not included in outgoing frames, and are instead post-facto generated by the modem during transmission[3] [5]. M. Chepponis and P. Karn made the technically correct decision of excluding HDLC framing from the AX.25 payload when developing KISS, but this causes an unfortunate situation where KISS is transporting an entirely undocumented fragment of the AX.25 packet. Figure 4 presents the AX.25 layer 2 protocol as it should be documented without the layer 1 HDLC framing.

3.2 Calculating the Frame Check Sum

The Frame Check Sum (FCS) used for error detection is the well-known CRC-16-CCITT, which enjoys a wide deployment in network protocols and systems.[4] Unfortunately, the language used in §4.2.5 of the ISO specification for HDLC [13] is particularly awkward, and does not lend itself well to implementation. For the sake of clarity, Figure 5 presents one possible algorithm to calculate the FCS of a complete frame, which is implemented in ANSI C in Appendix A. The constant 0x8408 comes from a bit reversal of the 0x1021 generator polynomial since the presented algorithm calculates the CRC in bit reversed order.

The order of the two octets and bit order of the checksum as transmitted over the air is particularly muddled in the existing amateur radio literature. Many sources call for sending the checksum little-endian, while ITU V.42 §8.1.2.3 specifies big-endian, as is the convention for most network protocols. The original specifications also call for transmitting the checksum most significant bit (MSb) first, which is the opposite of the payload octets. This exception is noted in §3.8 of the AX.25 specification as well.

[3]The opposite is also true; incoming frames with correct checksums have them stripped, and incorrect checksums cause the frame to be dropped and never transported over the KISS link.

[4]Example systems using CRC-16-CCITT include HDLC, Bluetooth, the XMODEM file transfer protocol, and SD cards.

```
1: function CALCULATE_CRC(frame[ ], frame_length)
2:     crc ← 0xFFFF
3:     for all byte ← frame_0, frame_{frame_length−1} do
4:         for all bit ← byte_LSb, byte_MSb do
5:             if crc_LSb ≠ bit then
6:                 crc ← (crc ≫ 1) XOR 0x8408
7:             else
8:                 crc ← crc ≫ 1
9:             end if
10:        end for
11:    end for
12:    crc ← crc XOR 0xFFFF
13:    return crc
14: end function
```

Figure 5: Algorithm to calculate CRC-16-CCITT in reverse-bit order.

This creates plenty of confusion on the part of implementers, which is likely caused by the fact that available reference implementations of the CCITT checksum don't make it clear that they already integrate the needed bit reversal after the Cyclic Redundancy Check (CRC) division and one's complement. The algorithm presented in Figure 5 *does not* calculate the true CCITT CRC checksum, but follows the convention of calculating it in reverse-bit order, such that the final bit-reversal step may be omitted during modulation of the bit stream. Therefore, the checksum as presented should be sent lower octet first, using the same modulator as for payload octets that modulates the least significant bit (LSb) first. This ensures that the resulting checksum as transmitted will have the correct sequence, starting with the MSb and finishing with the LSb, and is why many implementations appear to completely ignore the need to send the FCS MSb. Calculating the FCS in reverse bit order and then modulating it in LSb instead of MSb order cancel each other out, and alleviates the need for the modulator to handle the FCS differently than the rest of the frame.

4 FM Deviation and Emphasis

Once the Amateur Bell 202 frame is generated, encoded using NRZI, and converted into a baseband AFSK signal, it still needs to be converted into a VHF FM signal and transmitted to other stations. Since Bell 202 was originally designed for telephone data service, the existing specifications give no guidance on the unique aspects of the amateur VHF FM physical layer. One such issue is what value of FM deviation to target when setting modem audio levels.

While quantitatively justifying this figure is beyond the ability of the author, a proposed specification for FM deviation is 3.5kHz for both 1200Hz and 2200Hz tones, or for the wider of the two tones if equal deviation is not possible [10].

This proposal is complicated by two major issues:

- The lack of availability of the necessary test equipment to measure FM deviation.

- The inconsistency in pre-emphasis and de-emphasis filters used by individual network nodes.

The VHF deviation meter needed to properly set modem deviation is prohibitively expensive for the typical packet radio operator, so presenting a figure such as 3.5kHz deviation to most users does little good. Qualitative and home-brew solutions have been developed for setting deviation levels [1], and these techniques should be better promoted until deviation meters become a more standard part of a packet operator's toolkit.

Pre-emphasis and de-emphasis is a concept in FM voice communications where higher baseband frequencies are modulated with larger deviation than lower frequencies to provide a consistent signal-to-noise ratio across the channel. Unfortunately, the advantages of these audio filters to packet operation are debatable, and they are not applied consistently. A single packet station is likely to have any permutation of pre-emphasized or flat transmit audio and de-emphasized or flat receive audio. This means that even when one station deliberately uses flat audio, there is no guarantee that it won't suffer from receiving another station's pre-emphasized signal or be received by another station using de-emphasis.

Different types of Bell 202 modems vary in how sensitive they are to this high/low pass filtering effect, but more importantly there is no benchmark established for what level of pre-/de-emphasis a modem should tolerate. One suitable source for such a benchmark would be to go back to the telephone networks where Bell 202 was originally used. Figure 6 shows the allowable audio distortion of a basic type 3002 channel as used in the telephone network; any level of distortion that falls inside the shaded region relative to a test tone at 1004 Hz is considered acceptable [2].

For application to Amateur Bell 202, normalizing the distortion thresholds to 1004Hz is less important than simply noting that the allowable distortion between the two tones of interest (1200Hz and 2200Hz) is 10dB in either direction. An insightful measurement for Bell 202 modem designers would be testing how quickly their modem's performance falls off as a packet signal approaches the ±10dB limits. Creating a test suite which could yield quantitative measures of a modem's sensitivity to the pre-emphasis/de-emphasis issue would be a valuable contribution beyond the scope of this paper.

5 Carrier Sense Multiple Access

Since Amateur Bell 202 is a half duplex modulation using re-purposed FM voice transceivers, one of the challenges to packet radio is avoiding multiple stations transmitting on the same channel at once. This is done using Carrier Sense Multiple Access (CSMA), where each station listens before transmitting to see if the channel is clear. Unlike other CSMA implementations, such as IEEE 802.3 Ethernet, Amateur Bell 202 doesn't enjoy the advantage that transmitters can at least sense when a collision has taken place and use that information to abort the transmission of the rest of the frame early. When a collision takes place on an Amateur Bell 202 channel, both colliding frames are transmitted in their entirety, but both are lost and the channel time wasted.

Besides the degenerate case of ignoring the current channel status completely when deciding to transmit a pending frame,[5] there are two popular algorithms used for channel access in North

[5]This is a surprisingly common channel access method, used primarily by what are called "dumb" or "deaf" APRS trackers, which are transmit-only and lack an FM receiver altogether.

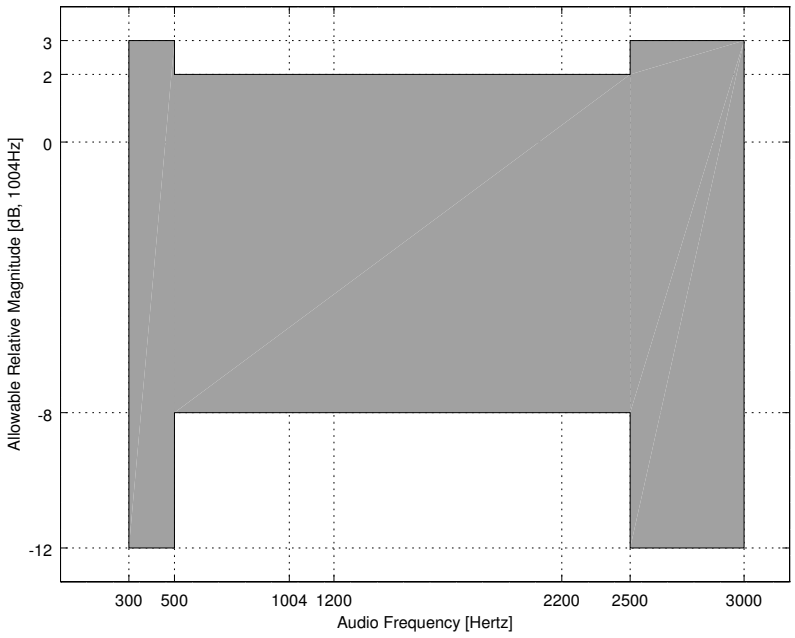

Figure 6: Allowable distortion in a basic 3002 telephone channel

America; DWait and P-persistent.

DWait is a deterministic algorithm where each station is assigned a fixed "quiet time" after the end of any other transmission before they will begin a locally pending transmission. This lends itself well to designed networks where the relative priority of each station is known and a corresponding DWait time is set for each station where a shorter DWait will always gain the channel over a longer one. Conversely, this doesn't lend itself well to ad-hoc networks, since two stations that happen to both operate near one another with similar DWait parameters will tend to collide and reduce throughput.

P-persistent is a stochastic algorithm that attempts to randomly spread stations apart when the channel becomes clear. This is configured with two variables: the slot time (SLottime), and the probability that a station should choose to transmit during a given slot (PErsist). The SLottime should be set to as short of a time interval as possible during which a station can reliably identify another station as transmitting before beginning its own transmission. The PErsist value should be tuned based on how likely another station is to transmit during the same time slot considering the number of other stations with pending traffic attempting to gain the channel.

A typical modem supporting the p-persistent algorithm will need three values adjusted based on the specific network hardware in use; PErsist, SLottime, and the transmitter preamble time TXDelay mentioned earlier in this paper.

- PErsist: Measured in units of 1/256, the suggested default is 63, which translates into a 0.25 chance of selecting a specific available slot [5]. A typical implementation selects a random number in the range [0,255] and tests if it is equal or less than the PErsist value. Therefore, a setting of 0 would result in a 0.004 chance of selecting a slot and a setting of 255 would result

in always selecting an open slot. The optimal value for a specific network is highly dependent on the local channel occupancy, so the suggested default shouldn't be considered definitive.

- SLottime: Measured in units of 10ms increments, the traditional default from sources such as the KISS specification and Kantronics hardware is a value of 10 (100ms) [5][8], but performance measurements of contemporary VHF radios indicate a need for a longer slot time. The new suggested value is 30 (300ms), which is discussed further in Appendix B.
- TXDelay: Measured in units of 10ms increments, the suggested default is a value of 50, which translates into a 500ms synchronization preamble from when a transmitter is keyed up until when a payload frame is transmitted [5]. This value is very conservative and can usually be reduced when receiving stations are properly configured with well aligned clock recovery mechanisms.

There are additional channel access methods beyond the two mentioned that are applied in amateur radio packet networks. Examples include Demand Assigned Multiple Access (DAMA), which is primarily used in European packet networks, and Time Slotting, which is used in carefully designed high-throughput networks. Since these alternatives see less application in American packet networks, they are excluded from this discussion and the reader need only appreciate that this is not a comprehensive survey of channel access methods.

6 Conclusion

By most measures, Amateur Bell 202 is a very poor performing modulation to be used by amateurs for packet operations. One-bit symbols cause Bell 202 to suffer from poor spectral efficiency, HDLC lacks any error correcting codes so single-bit errors cause entire frames to be dropped, and 1200 bits per second is a remarkably low data rate when even consumer radio systems are operating at hundreds of millions of bits per second throughput.

One aspect of Amateur Bell 202 that is appealing, other than the huge legacy systems still using it, is it's relative simplicity. The fact that amateurs are able to implement Amateur Bell 202 modems on systems as minimalistic as 8 bit microcontrollers, and that modems can interface with unmodified voice radios, make Amateur Bell 202 much more accessible than more sophisticated protocols.

Faster data rates and more sophisticated modems should never be discouraged, but the value of being able to learn about amateur digital communications via the simplicity of Bell 202 shouldn't be discounted. Unfortunately, as the technological context where protocols such as Amateur Bell 202 were developed moves further into the past, the need to write additional documentation for the next generation of packet radio operators is going to become increasingly critical to the digital communications aspect of the amateur radio hobby.

A Reference CRC-16-CCITT Implementation

```
// CRC-16-CCITT Reference Implementation in C
// Kenneth Finnegan, 2014
//
// This is a skeleton program that takes a static AX.25 frame,
// calculates the Frame Check Sum, and prints every octet as hex.

#include <stdio.h>
#include <stdint.h>

uint16_t calc_crc(uint8_t frame[], size_t frame_len);
void send_octet(uint8_t byte);

int main(void) {
        uint16_t crc;
        int i;

        // Sample APRS frame payload - FCS = {0x76, 0x4A}
        uint8_t testvector[] =
                { 0x82, 0xA0, 0xB4, 0x60, 0x60, 0x60, 0xE0, // "APZ___"
                  0x9C, 0x60, 0x86, 0x82, 0x98, 0x98, 0xE3, // "N0CALL-1"
                  0x03, 0xF0, 0x2C, 0x41}; // Control PID ",A"
        size_t testlength = sizeof(testvector);

        // Calculate the FCS
        crc = calc_crc(testvector, testlength);

        // "Transmit" the complete frame
        printf("             ........7E\n");
        for (i=0; i<testlength; i++) {
                send_octet(testvector[i]);
        }
        send_octet(crc & 0xFF); // Send FCS bits 8-15
        send_octet((crc >> 8) & 0xFF); // Send FCS bits 0-7

        printf("\n7E........\n");
        return 0;
}

// Calculate the CRC-16-CCITT of a given array of a given length
// NOTE: Operates completely in reverse-bit order
uint16_t calc_crc(uint8_t frame[], size_t frame_len) {
        int i, j;
```

```
        // Preload the CRC register with ones
        uint16_t crc = 0xffff;

        // Iterate over every octet in the frame
        for (i=0; i<frame_len; i++) {
                // Iterate over every bit, LSb first
                for (j=0; j<8; j++) {
                        int bit = (frame[i] >> j) & 0x01;
                        // Divide by a bit-reversed 0x1021
                        if ( (crc & 0x0001) != (bit) ) {
                                crc = (crc >> 1) ^ 0x8408;
                        } else {
                                crc = crc >> 1;
                        }
                }
        }

        // Take the one's compliment of the calculated CRC
        crc = crc ^ 0xffff;

        return crc;
}

// A dummy modulator that only prints each octet to the screen
// NOTE: The actual modulator should send the argument byte
// least significant bit first, and handle bit-stuffing
// the entire frame
void send_octet(uint8_t byte) {
        static int octetsperline = 0;

        printf("%02X ", byte);

        if ((++octetsperline) > 7) {
                octetsperline = 0;
                printf("\n");
        }
        return;
}

// END CRC-16-CCITT Reference Implementation
```

B SLottime Justification

The SLottime parameter of modems is dependent upon the total time it takes for one station to begin transmitting and for receiving stations to subsequently identify the channel as occupied. This sequence can be broken into the following stages:

- Transmitter key-up from standby (RX-TX turnaround time)
- RF propagation from transmitter to receiver (negligible)
- Receiver opening squelch and delivering AFSK signal to modem (While not usually measured, a representative measurement is the TX-RX turnaround time [11])
- Modem Data Carrier Detect (DCD) of the received signal

Radio Model	RX-TX Time	TX-RX Time	QST Issue
Yaesu FT-2600M	55ms	60ms	Dec 1999
Icom IC-910H	32ms	70ms	May 2001
Kenwood TM-271A	72ms	88ms	Mar 2004
Yaesu FT-7800R	190ms	98ms	Apr 2004
Yaesu FT-8800R	120ms	190ms	May 2005
Icom ID-800H	55ms	173ms	Nov 2005
Kenwood TM-V708A	56ms	86ms	Apr 2006
Yaesu FT-1802M	77ms	130ms	Jun 2006
Kenwood TM-V71A	75ms	102ms	Nov 2007
Icom IC-2820H	43ms	110ms	Nov 2007
Kenwood TM-D710A	75ms	106ms	Feb 2008
Yaesu FT-1900R	74ms	150ms	May 2010
Yaesu FTM-350R	134ms	70ms	Jan 2011

Since most Amateur Bell 202 packet activity is done using amateur VHF mobile radios, a survey was performed of ARRL QST magazine hardware reviews published on VHF-capable mobile radios since 1999. These reviews included RX-TX and TX-RX turnaround times which indicate that a significantly longer SLottime is needed than the traditionally suggested 100ms. This is due to *none* of the evaluated radios being fast enough to key up, a second radio to open squelch, and then allow any time for modem DCD before the end of a slot. It is the author's opinion that a new default of 300ms SLottime for modems be considered.

References

[1] J. Ackermann, "Setting Your TNC's Audio Drive Level." `http://www.febo.com/packet/layer-one/transmit.html`

[2] V. F. Alisouskas, W. Tomasi, *Digital and Data Communications*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc, 1985.

[3] American Telephone and Telegraph Company, *Data Sets 202S and 202T Interface Specification*, AT&T Publication 41212, July, 1976.

[4] W. Beech, et al., *AX.25 Link Access Protocol for Amateur Packet Radio Version 2.2*. Tucson, Arizona: Tucson Amateur Packet Radio Corp, 1998. `http://www.tapr.org/pdf/AX25.2.2.pdf`

[5] M. Chepponis, P. Karn, "The KISS TNC: A simple Host-to-TNC communications protocol," in *ARRL 6th Computer Networking Conference*, 1987, pp/ 38-43. Translated to HTML Jan. 1997 by P. Karn, `http://ax25.net/kiss.aspx`

[6] *Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion*, ITU-T standard V.42, 2002.

[7] S. Horzepa, *Your Gateway to Packet Radio*, Newington, Connecticut: American Radio Relay League, 1989.

[8] Kantronics, *KAM Plus Reference Manual*, Lawrence, Kansas: Kantronics, 1994.

[9] S. Miller, "1200 Baud Packet Radio Details." `http://n1vg.net/packet/index.php`

[10] S. Miller. Personal interview. 25 Mar. 2014.

[11] R. Patterson. Personal interview. 31 Mar. 2014.

[12] B. Simmons, "APRS Unveiled," in *QEX Magazine*, pp. 19-23, Nov./Dec. 2012.

[13] *Telecommunications and information exchange between systems — High-level data link control (HDLC) procedures*, ISO Standard 13239, 2002.