Some Comments In General:

Mobile Data Terminals
The RD-LAP Protocol
Wireless Data Security

With Some Comments In Particular:

The Public Safety Wireless Data Networks of the Cities of Huntsville & Madison Alabama

Abstract

The cities of Huntsville and Madison Alabama operate a public safety MDT (Mobile Data Terminal) network using Motorola's RD-LAP (Radio Data-Link Access Protocol) wireless data transmission protocol. We review publicly available RD-LAP related documentation and demonstrate that this can be used to create a complete SDR (Software Defined Radio) using an Ettus Research USRP (Universal Software Radio Peripheral). It has been determined that encryption is not used on this public safety system and that a variety of sensitive information is being broadcast in the clear. This includes NCIC (National Crime Information Center) data in apparent violation of the FBI's (Federal Bureau of Investigation) CJIS (Criminal Justice Information Services) standards. The system at present is actively exploitable at the scanner enthusiast/hobbyist level with readily available and modest equipment requirements. Some further miscellaneous comments address various other perceived security lapses in this system.

Table of Contents

Introduction – MDT System Traffic Description....3
MDT Wireless Traffic Security Considerations....5
Experimental Hardware Setup....8
RD-LAP Signal Sources, Greater Huntsville Alabama Area....10
The RD-LAP Protocol – Physical Layer....11
RD-LAP SDR Software Implementation Overview....14
The RD-LAP Rate ¾ Trellis Coded Modulation Error Correction....22
RDLAP PDU & SDU Data Block Structures....26
Putting It All Together With An Example Message....30
Appendix A - RDLAP CRC Specifications....33
Appendix B - RDLAP Over The Air Interface Illustration....34
COLOPHON....35

Introduction – MDT System Traffic Description

A Mobile Data Terminal (MDT) system is a mobile computing device that communicates with a centralized server for records entry and retrieval[1]. A modern public safety communications system typically deploys in-car laptop computers with some type of wireless data link to the central dispatch office. The mobile computing device wirelessly sends and receives status reports, job assignments, emaisl, instant messages, etc. Other equipment may produce additional wireless data traffic such as GPS coordinates from an Automatic Vehicle Location (AVL) system.

Following are some common data types transmitted ***in the clear*** on the Huntsville & Madison Alabama MDT system.

Events requiring Public Safety Intervention (Police & Fire Departments):

- Event number
- Handling Agency & Assigned Units
- Times of creation, dispatch, completion, etc..
- Event Type
- Location – Address and nearest cross streets
- Complainant, Contact Information (Enhanced 911 Data)
- Status
- Past events reported for that particular address
- Comments

Automatic Vehicle Location (AVL) data collection using the Trimble ASCII Interface Protocol (TAIP)[2]:

- Requests From Central Dispatch to remote GPS Receivers
- Reports from Remote units including time, bearing, velocity, and geographical coordinates.

Periodic AVL data summaries broadcast to all users, giving geographical location (Alabama State Plane Coordinates), status, unit ids, unit affiliations, etc of sometimes two hundred fifty units county wide from the following agencies:

- Huntsville City Police & Fire Departments
- Madison City Police & Fire Departments
- Miscellaneous Madison County Cities: Owens Cross Roads, Gurley, New Hope
- Madison Fire & Sheriff Departments
- HEMSI (Huntsville Emergency Medical Services Inc), Private Ambulance Company

A license plate number check produces the following:

- Registered Owner's Name & Address
- Vehicle Type (e.g. Passenger), Make, Model Year, Color
- Plate Number & Expiration Dates
- VIN Number

---

1 http://en.wikipedia.org/wiki/Mobile_data_terminal

2 Useful for studies such as determining police officer probability density distribution functions and calculating overlap integrals against donut shop related eigenfunctions. Experimental evidence suggests arbitrarily many such units can and do occupy the same spatial location and therefore must be bosons. The triggering mechanism of these Bose-Einstein condensates is a matter of some conjecture.

Public safety officer related information:

- Full names
- Status
- Email & Instant messages
- Computer terminal logins

NCIC queries which are most commonly done on vehicle plate numbers and driver's license checks. Such NCIC queries result in the following data transmitted (did we already mention ***in the clear***):

- Driver's license number
- Name
- Date of birth
- Home Address
- Social Security Number
- Identifying characteristics (Race, Sex, Height, Weight, Hair, Eyes)
- Outstanding warrants
- Car registration information (make, model, year, color, VIN number)

Some modern MDT systems can also handle pictures, thumb-prints, et cetera. These capabilities do not seem to have been deployed on the Huntsville & Madison Alabama systems at this time.

MDT Wireless Traffic Security Considerations

Interception of MDT traffic not specifically forbidden by U.S. Law.

The federal governing law appears to be the Electronic Communications Privacy Act , U.S. Code Title 18, Part I, Chapter 119. It specifically allows monitoring of communications "by any governmental, law enforcement, civil defense, private land mobile, or public safety communications system, including police and fire, readily accessible to the general public"; readily accessible to the general public in this situation means not "scrambled or encrypted" and not "transmitted using modulation techniques whose essential parameters have been withheld from the public with the intention of preserving the privacy of such communication".

We shall see that all RD-LAP modulation techniques are publicly available and that no attempt has been made to scramble or encrypt the data. Therefore the only legally actionable solution to protect this type of data therefore appears to be the use of encryption techniques intended to preserve communications privacy. Encryption is a trivial task in this day and age and confers legal protection.

Since this is not a legal brief we will leave it as an exercise for the reader to determine the legality of intercepting this type of MDT traffic. Don't forget about state and local laws in your area.

Mandated FBI CJIS Security Standards Ignored

The FBI's CJIS[3] sets minimum data security standards for law enforcement organizations accessing the NCIC system. Wireless data links are specifically addressed. The Huntsville / Madison County MDT system provides access to NCIC records and therefore should adhere to these minimums.

Systems accessing NCIC "hot" file data after 9/30/2002 must use a minimum 128 bit encryption. The FCC license data (Table 1) suggests the system in question was implemented past the 9/30/2002 cutoff date for mandatory 128 bit encryption. But the Huntsville / Madison Alabama system does not use any form of encryption.

For systems put in operation prior to 9/30/2002 a minimum 56 bit encryption or a proprietary data transmission protocol that prevents recognizable clear text transmissions must be used. The RD-LAP protocol is publicly available and the higher level protocols do not use encryption of any kind allowing reception of recognizable clear text transmissions. So the system in question does not meet the old standard either.

User access controls are also found to be lacking. User log in at the start of a shift and remain logged in without any further action until the end of the shift. There is no evident message authentication thus allowing user session hijacking and arbitrary access to the NCIC system. NCIC incorporates extensive logging to prevent abuse of the system, but none of that does any good whatsoever when a legitimate public safety officer's data session is hijacked. In such a scenario audit records would reveal a pattern of NCIC misuse leading back to innocent public safety officer(s) and in the worst case result in severe sanctions of innocent personnel.

The FBI's apparent lack of oversight is somewhat puzzling[4].

---

3 http://www.fbi.gov/hq/cjisd/ncic.htm

4 Or maybe not. This is the same FBI after all which shall forever be remembered in information technology circles for the notorious $100,000,000 + Virtual Case File disaster. http://en.wikipedia.org/wiki/Virtual_Case_File

Unencrypted MDT Data ~ Massive Misuse Potential In The Wrong Hands

Simple passive monitoring of unencrypted MDT traffic represents a real security concern.

Foremost it should be apparent that a steady stream of personal identifying information would be including name, date of birth, Social Security Number et cetera represents a massive identity theft risk. Over time a data base could be built up to allow matching an illegal immigrant or terrorist with a cover identity that matches in all particulars including characteristics such as weight, height, hair, and eye color. And of course this database can exclude identities with outstanding warrants or prior criminal histories. A more prosaic application would be straight out identity theft[5].

Unit dispositions, real time geographic locations, emails, instant messages, and officer names publicly exposed to civilian oversight tends to irritate public safety officers, their departments, and generally the overall city or county government. Some civilian monitoring enthusiasts might use this type of information to help avoid undesired public safety enforcement activity (speed traps or otherwise).

It should be pointed out that passive monitoring is just about impossible to detect; such system exploitations may only become apparent long after the information leaks out to the wrong quarters. An unencrypted MDT system combined with widely available sub $100 PCMCIA cards shown in the next section presents a very real, very practical, and very easy exploitation approach.

Encryption or the lack thereof is only one component of a comprehensive security design. Some other aspects that must be addressed are access control (what type of access privileges are given to user X) and message authentication (is an incoming request really from user X or just someone pretending to be user X)[6]. The MDT system in question seems to rely on unique 32 bit hardware addresses to determine which terminal a message was sent from; these addresses are trivial to collect and then spoof. Because no message authentication exists the system will process any spoofed message at the attendant privilege levels of the authorized user (session hijacking). If one wants to take a whirl exploring the NCIC databases for fun & profit one may simply hijack the session of an authorized user. NCIC access logs will not provide meaningful information in this scenario.

Even the use of strong encryption and message authentication does not guarantee a functional and secure system. For example, replay attacks must also be protected against... after all, of what use is a critical infrastructure wireless communications system if someone were to constantly transmit a continuous stream of previously recorded user logout requests?

Going further, the RD-LAP protocol used in this system is not designed to resist jamming attacks. The technology exists to make wireless data communications much more secure against active denial of service attacks but no one is pushing its deployment[7]. It is ironic but not surprising that

---

5 Identity theft should be condemned in the strongest possible terms because not only is it theft, it also places a massive and undue burden on the victim with law enforcement generally much less than helpful. It is ironic therefore that in this situation we have law enforcement potentially aiding and abetting identity theft.

6 Lack of authentication is an endemic wireless security failure mode. Other examples of abusable and societally consequential systems might be wireless SCADA systems or the Emergency Alert System (EAS) where a properly placed and transmitted message automatically propagates statewide (minimal equipment) or nationwide (moderate equipment requirements).

7 One would think a society that has developed the means to push billions of dollars through the Department of Homeland Security faster than a Taco Bell burrito moves through a set of human intestines might want to set aside some money for this. On the contrary much of this money is being invested in equally vulnerable APCO-25 applications.

private sector wireless telecommunications infrastructure such as GSM or CDMA cellular telephone systems provide better security and jamming resistance than systems purchased for government use.

The Historical Context

A Google search suggests that scanner enthusiasts have been successfully monitoring MDT systems for a period of at least ten years. There is simply no excuse for not knowing encryption must be used as part of a comprehensive security strategy.

Final Comments:

One would think this callous and total disregard for data protection exposes the City of Huntsville, City Of Madison, and allied county departments to severe legal jeopardy. Perhaps a class action lawsuit on behalf of all victims whose identities have been so carelessly exposed and possibly stolen would be in order as the fastest way to encourage some corrective remedies.

In any case some type of immediate corrective action must be undertaken to minimize system down time after the FBI terminates NCIC access for every law enforcement agency in Madison County, Alabama.

The shattered public trust and faith in competent law enforcement both at the local and federal (FBI) level will of course take much longer to restore.

Experimental Hardware Setup

Hardware Description Using An Ettus Research USRP:

Signals are sampled off the air using an Ettus Research (http://www.ettus.com/) USRP (Universal Software Radio Peripheral) hardware with a DBSRX (800-2400 MHz) or TVRX (50-870 Mhz) receiver boards. All further processing takes place on a Linux PC system (or Windows PC running Cygwin) using GNU Radio software (http://gnuradio.org/trac). The USRP (an actively developed commercial hardware system) works hand in hand with the GNU Radio PC software component (an actively developed open source project).

The Ettus Research USRP + GNU Radio system is the preferred solution for its inherent flexibility. Since it is a software defined radio system various algorithms, filters, time constants, etc are easily tested and evaluated. This is ideal for development work in a fixed location.

The only

GNU Radio RD-LAP processing code may be released publicly if there is sufficient interest and time permits writing the required code modules.

Some Alternate Hardware Configurations:

There is no reason alternate setups could not be used, especially if the cost of a USRP (roughly ~ U.S. $1000) proves too high.

For example, a Consumer Microcircuits Limited MX929 4-Level FSK transceiver chip could be used to add RD-LAP receive capabilities to a suitable discriminator tapped radio scanner. Hardware development boards based on the MX929 and similar chips are readily available[8]. These publicly available chips handle the entire RD-LAP protocol thus making it possible to build an entire transceiver without having to know a thing about the RD-LAP protocol.

Analog Devices also produces nice transceivers such as the ADF7021 which may be suitable for selected public safety frequency bands but does not implement the RD-LAP protocol. The RD-LAP protocol functions implemented on a embedded micro-controller result in a highly integrated and highly portable system.

RD-LAP PCMCIA transceiver cards are also readily available and usable for this type of project. As an example:

8 The U.S. distributor of the MX929 chips lists single unit pricing of about ~$35 a piece. http://www.cdiweb.com/PortalParametricSearch.aspx?item=588408&type=82&id=2

Illustration 1: Surplus 90's vintage 19,200 bps RD-LAP PCMCIA wireless modem card with antenna adapter. Frequency coverage of 806-825, 851-870 MHz perfectly matches public safety systems.

Available from EBAY. Cheap. Highly portable. Some device driver skills required.

Various usable RD-LAP equipment appears from time to on EBAY with acquisition costs sometimes on the order of ~$10. This can include vehicular mounted RD-LAP transceivers boasting tens of Watts of transmit power perfect for those times when a lowly PCMCIA card just doesn't have sufficient reach.

RD-LAP Signal Sources, Greater Huntsville Alabama Area

We gratefully acknowledge the taxpayers who have provided the following Huntsville & Madison Alabama public safety RD-LAP systems for public experimentation:

Base Station Frequency	Call Sign	License Date	Licensee	Transmitter Site
860.7375 MHz	WPPD898	9/23/2004	Huntsville City	Madkin Mtn & Burritt Museum (Two Sites)
859.7375 MHz	WPPD898	9/23/2004	Huntsville City	Madkin Mtn & Burritt Museum (Two Sites)
858.7125 MHz	WPPF587	9/02/2004	Madison City	Rainbow Mtn

Table 1: Active and licensed public safety RD-LAP systems in Madison County, Alabama. The mobile frequencies are 45 MHz below the base station frequency in the 800 MHz band. The 860.7375MHz frequency is licensed but not active as of 22 Sep 2007.

There is one additional RD-LAP system in the Madison county area at 464.650MHz licensed to Sharp Communications. It is not public safety related but instead intended for private land mobile data applications.

The RD-LAP Protocol – Physical Layer

Public References

Consumer Microcircuits Limited MX929B Datasheet

Any standard communications book that covers multi-level FSK modulation schemes

The signal has the following characteristics:

- 4L-FSK (Four Level FSK) Modulation Format
- Nominal symbol levels at -3600, -1200, +1200, +3600Hz.
- Specified Deviation: ±5.6KHz
- RRC Symbol Filtering, BT = 0.2
- 9600 Symbols / second (19200 kilobits / second raw throughput)
- Intended for 25KHz Channel Spacing
- Constant Envelope Modulation

Note: Not C4FM (APCO-25 CAI) compatible. The phase is nowhere nearly as nicely controlled plus frequency deviation and baud rate are off by a factor of two.

Representative over the air characteristics are shown in the following three illustrations:

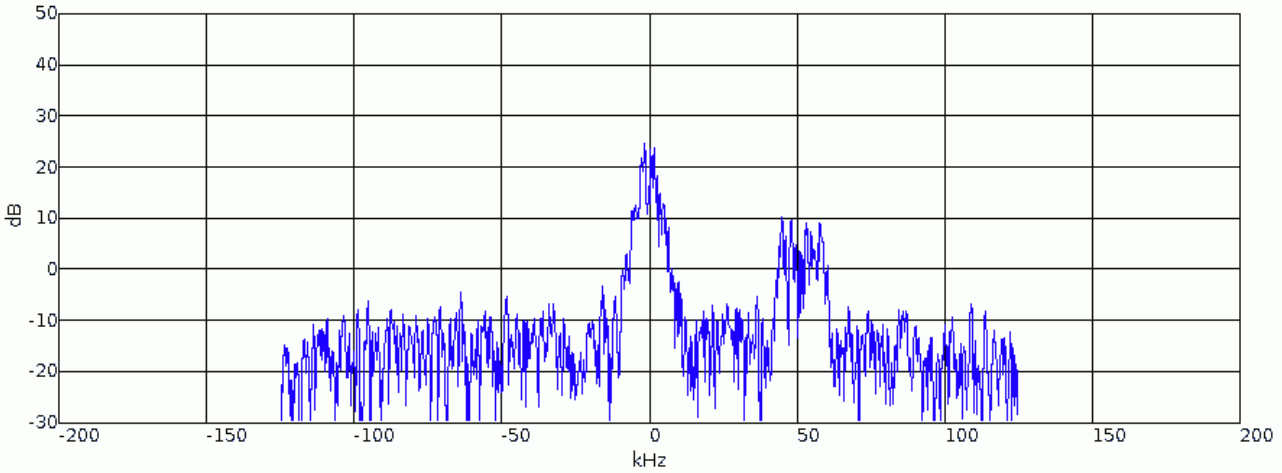


Illustration 2: FFT centered on 859.7375MHz 9600 symbol/second 4-Level FSK RDLAP transmission. A weaker unrelated signal at 859.7875MHz can also be seen.

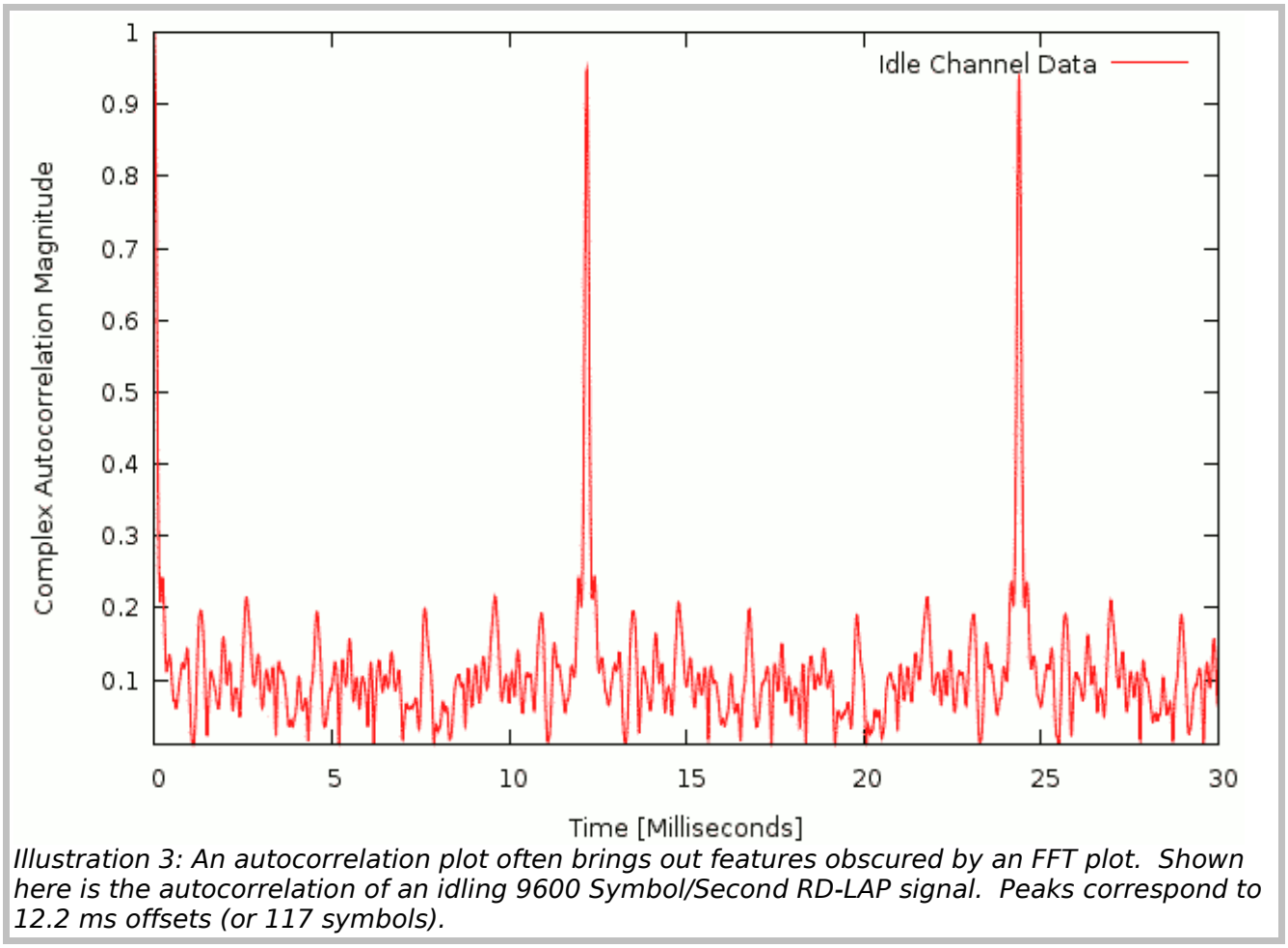


Illustration 3: An autocorrelation plot often brings out features obscured by an FFT plot. Shown here is the autocorrelation of an idling 9600 Symbol/Second RD-LAP signal. Peaks correspond to 12.2 ms offsets (or 117 symbols).

The system helps avoids keyup splatter by a ramped power up profile as evident in the following plot:

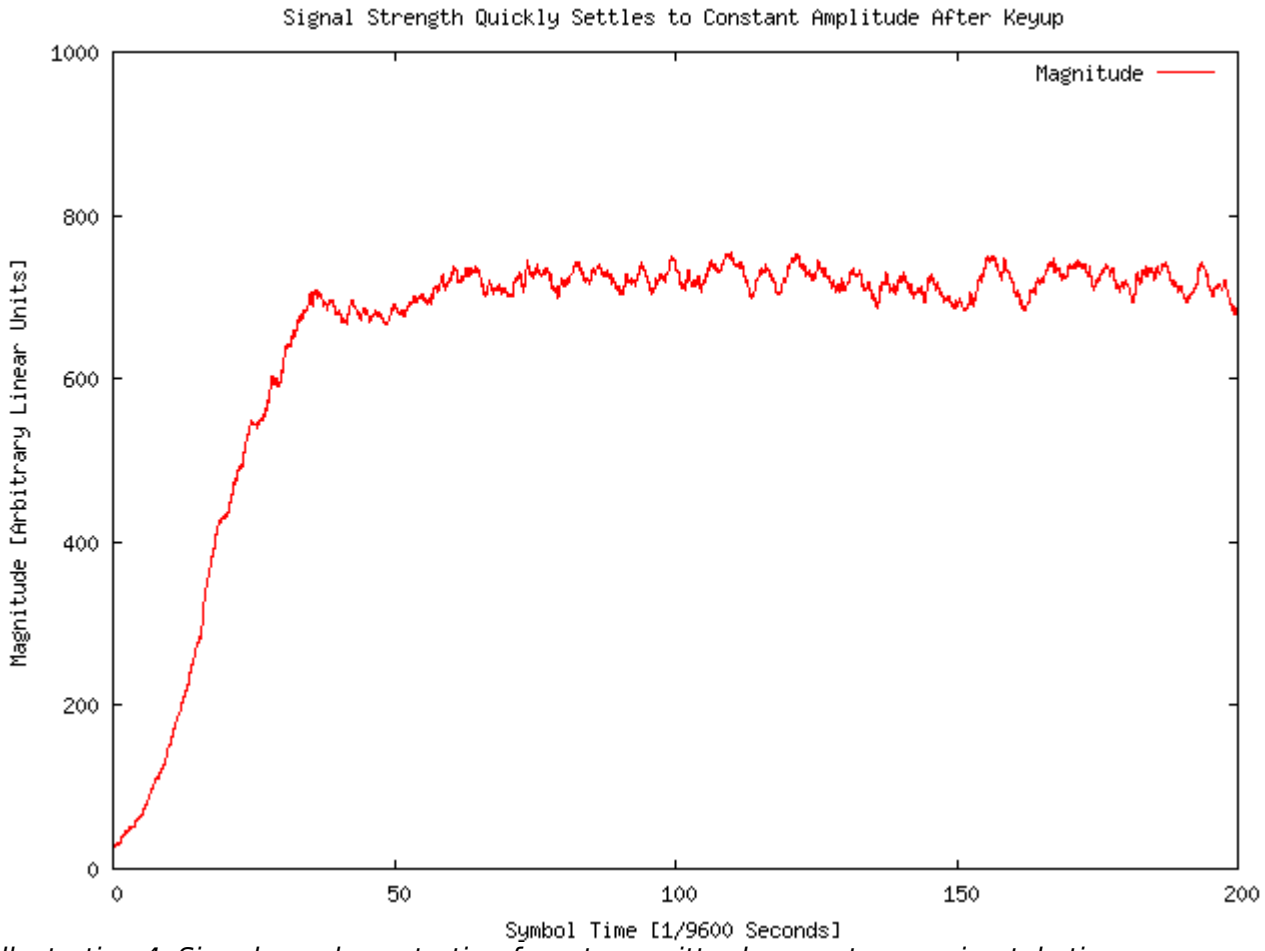


Illustration 4: Signal envelope starting from transmitter keyup at approximately time zero. Power ramps up within three milliseconds. Filtered with approximately 500 µS time constant.

RD-LAP SDR Software Implementation Overview

Public References

The described four level FSK demodulation algorithm roughly follows that described in U.S. Patent 5,553,101. Modifications had to be made for different sampling rates, modifying the A/D converter sampling clock to instead be a symbol sampling clock, addition of channel and RRC filters, et cetera. We have not changed the core algorithms that correct for the symbol timing, symbol spread, and carrier offset.

This section describes the first processing step, namely locking onto the received data stream and recovering a sequence of transmitted symbols. This method also works quite well on APCO-25 12.5 Khz C4FM channels.

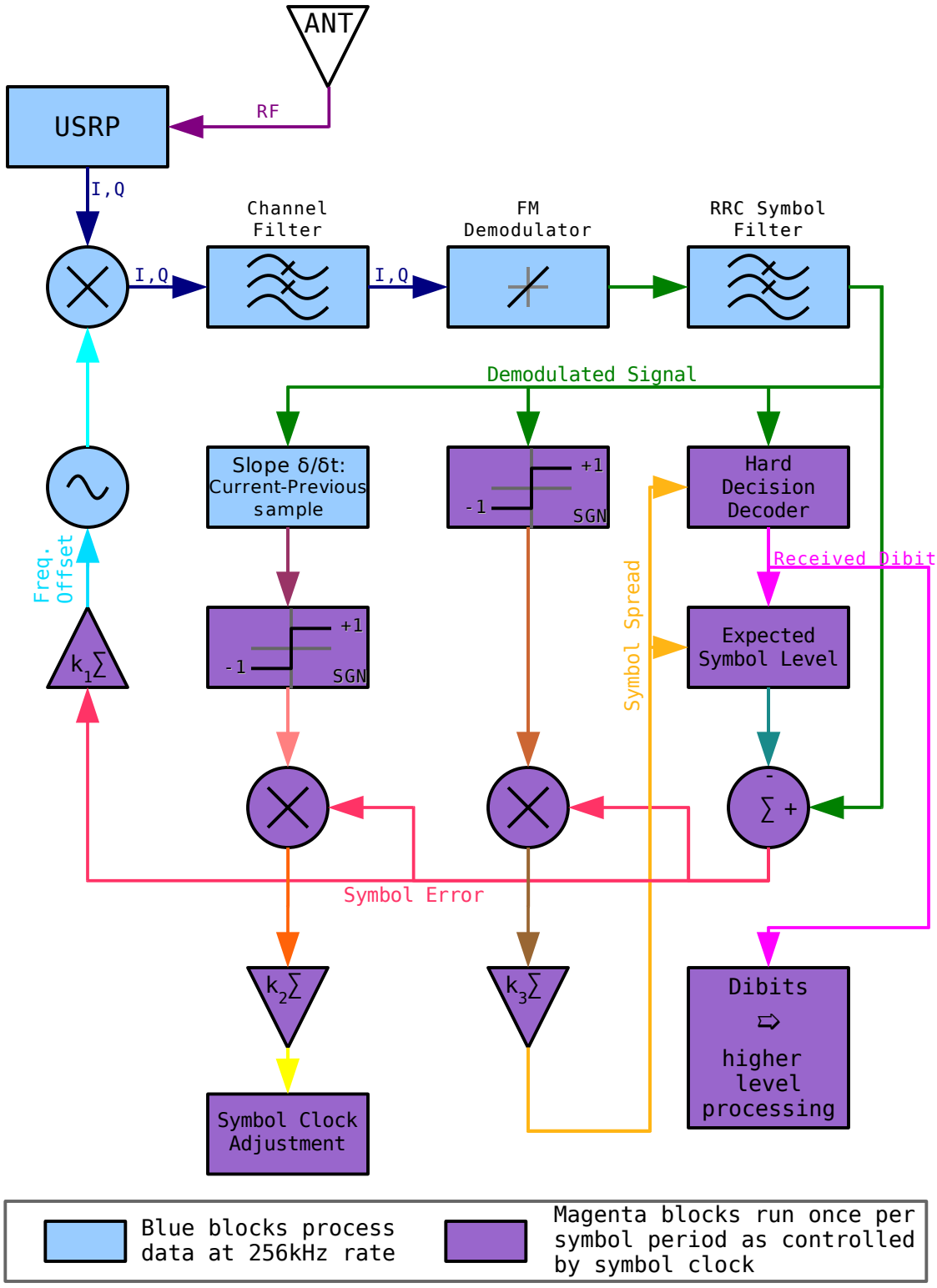
We start with a decimated complex (i.e. I + Q channel) 256 kilo-sample/second data stream received from the USRP. We apply a frequency shift to correct for any USRP receiver board offsets and any received carrier frequency offsets. A 25KHz channel selection filter (reasonably close to a brick wall frequency response) is used to select the desired desired data channel. The complex data stream at this point is run through a standard FM demodulator block and filtered using a standard RRC filter with a BT time constant of 0.2.

Addition of hard limits allowed for better response times. and allow for somewhat more filtering. The same concept as preventing integrator wrap-up on a PID controller.

At this point a tracker module recovers symbol timing, frequency deviation, and frequency offset information which generates a received 4-Level symbol stream. Subsequent sections of this document describe RD-LAP protocol specific processing used to synchronize and extract application data, and finally higher level application specific protocols which are used to decompress and assemble data into an html format for viewing in a normal web browser.

There is still some optimization possible. For example, the 256kHz sampling rate could be decimated to ease the processing burden on subsequent processing blocks. Furthermore, it is a waste of processing power to evaluate the RRC symbol filter at the 256kHz sampling rate as the RRC filter output is only evaluated near symbol sampling times (9600Hz in this case).

The 4LFSK Receiver block diagram show in drawing 1 on the next page. It exhibits very pleasing performance characteristics in that the operational time constants are rather non-critical.



Drawing 1: Four Level FSK receiver block diagram. k1, k2, and k3 constants set frequency offset, symbol clock, and symbol spread (carrier deviation) sensitivity respectively. Not shown: hard limits on symbol spread and frequency offset adjustment ranges.

At this point we need to verify proper receiver functionality. The behavior of the SDR symbol recovery algorithm can be seen when we plot the incoming signal, a few key parameters, and the sampled symbol points as a function of time:

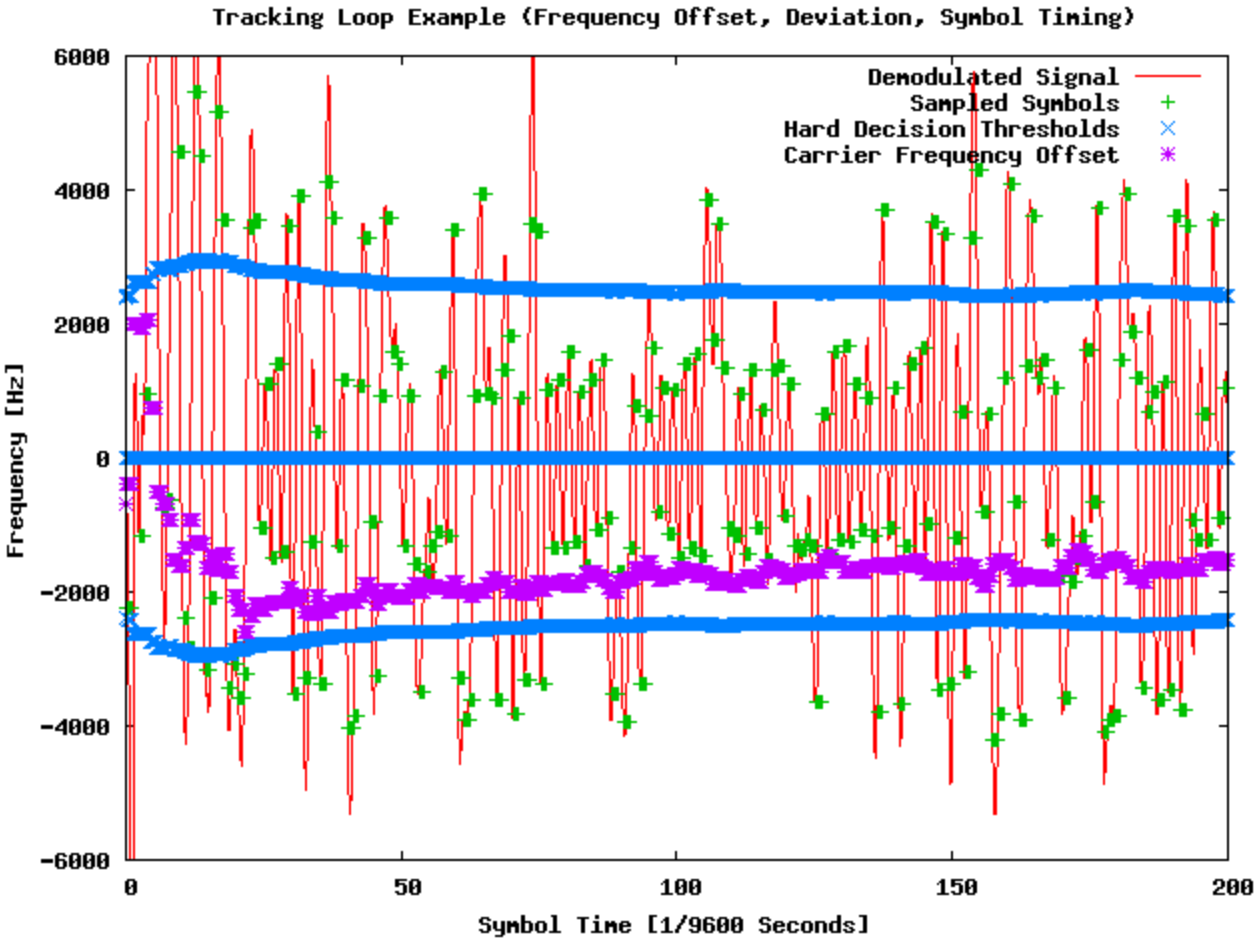


Illustration 5: Four level FSK symbol recovery showing successful carrier tracking of the 859.7375MHz Burritt Museum Transmitter Site Signal. Transmitter key up has occurred around time zero. Acquisition has succeeded by about the 25th symbol time point. Note the atrocious carrier frequency drift which can exceed 1 kHz over the medium term (order of seconds)!

Having successfully recovered symbol timing we can move on and generate an eye diagram:

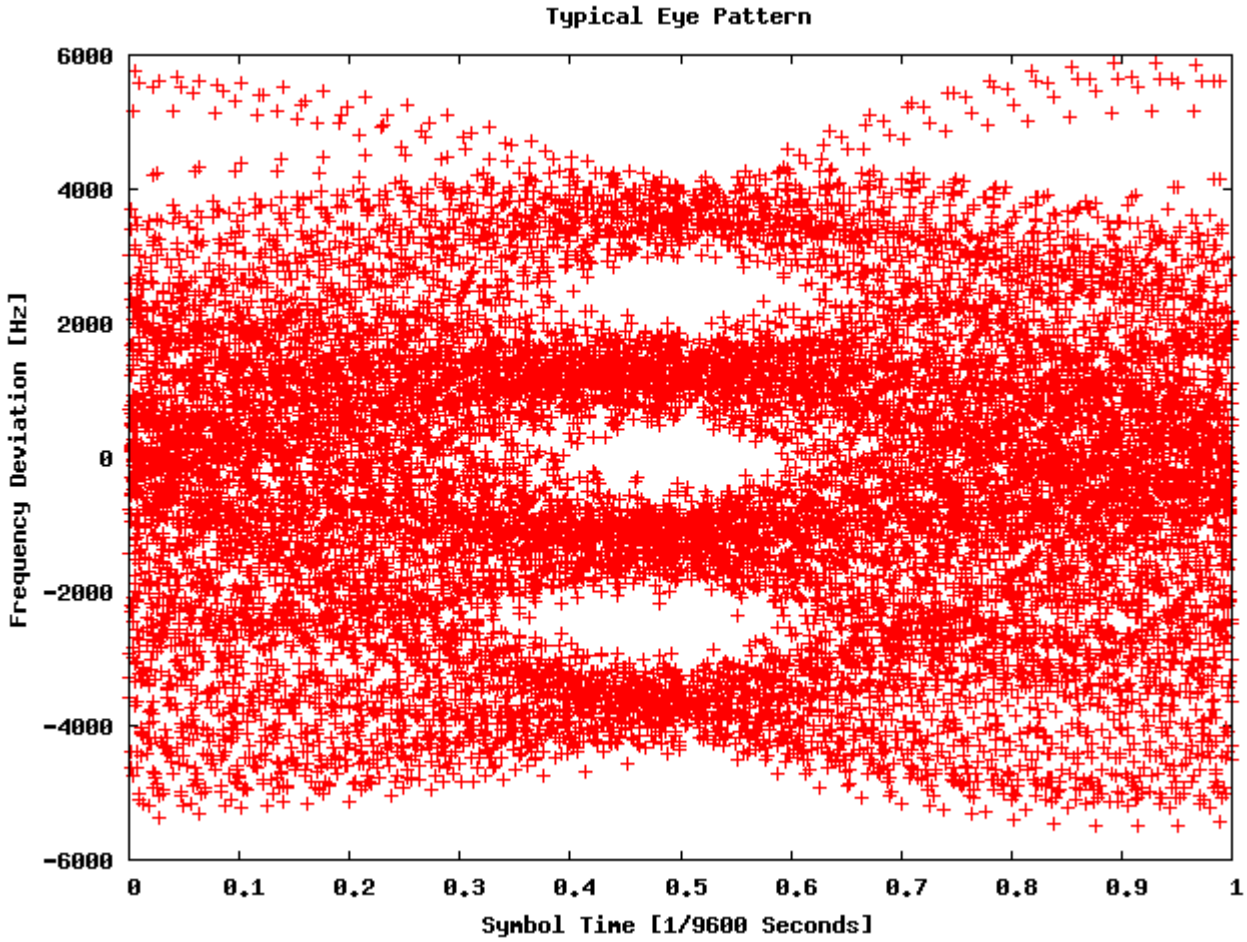


Illustration 6: Example received data eye pattern. Sampling occurs at T = 0.5 Symbol Times. The eye has opened adequately at the expected sample time and we expect a symbol stream with few errors. Note: Final RRC filter not applied, moderate S/N.

Nominal frequency deviation for data transmission can be see to be: -3600, -1200, +1200, +3600Hz. We are close to the specified RDLAP deviation of +/- 5.6 Khz even without the final symbol filtering.

As a final note: It will be seen that the forward error correction and interleaving uses adjacent symbols in pairs to form a 16 point constellation. It is sufficient to assess the reception quality using this type of eye diagram since we are receiving a stream of individual 4 Level FSK symbols; a constellation diagram made from 4L-FSK pairs is certainly possible but would not provide any additional useful information[9].

9 Other than maybe proving a susceptibility to mathematical self-gratification.

RD-LAP Over-The-Air Data Transmission Example

Public References

http://www.cmlmicro.com/Products/WData/FX929B.htm Please refer to Appendix B for a figure showing the RD-LAP over the air protocol taken from this FX929 datasheet.

We will present an explanation of RD-LAP data interleave and forward error correction in a step by step example going from an over-the-air data stream to recovered data. Therefore we start with a hypothetical data stream in the order the symbols would be received. This stream represents the output of the symbol timing/carrier tracking/carrier deviation tracking four-level FSK receive block with hard decision decoding into one of four possible symbol states labeled -3, -1, +1, and +3 (see previous section).

The hypothetical data stream broken up by function is shown in the following table:

Function		# of Symbols	Example Data
Symbol Sync		24	+3 +3 -3 -3 +3 +3 -3 -3 +3 +3 -3 -3 +3 +3 -3 -3 +3 +3 -3 -3 -3 -3 +3 +3
Frame Sync		24	-1 +1 -1 +1 -1 +3 -3 +3 -3 -1 +1 -3 +3 +3 -1 +1 -3 -3 +1 +3 -1 -3 +1 +3
Status Symbol		1	+1
Station ID Slot		22	+1 -1 +1 -1 -3 -1 -1 -1 +1 -1 +3 -3 -3 -3 -3 +3 +3 +1 +1 +1 -1 -3
Status Symbol		1	-3
Header Block	Micro-Slot 1	22	-3 +3 -3 -3 -1 +1 -1 +1 +1 +3 -1 -1 +3 -3 -3 -3 +3 -3 +3 +1 +1 +3
	Status Symbol	1	+1
	Micro-Slot 2	22	-1 +1 -1 +1 -1 -1 +1 -1 -3 -3 +3 -3 +3 +3 +3 +3 -1 +1 -1 +1 +3 -1
	Status Symbol	1	+1
	Micro-Slot 3	22	+1 -1 +1 -1 -3 +1 +1 -3 -1 +1 -1 +1 -1 +1 +3 -1 +1 -1 +3 -3 -1 -1
	Status Symbol	1	+1
Info Blocks	End of example data... Additional Blocks Holding User Data (Number Specified In Header Block) When all blocks have been transmitted we start transmitting a Frame Sync		

Note: Status Symbol -3 Indicates Idle; +3 = Busy; Otherwise Unknown

First an explanation of the various symbol functions:

At the start of every transmission comes the *symbol sync* sequence. Its sole purpose is to provide some well behaved data for the receiver to lock onto. No special efforts need to be made to detect or decode the symbol sync. Usually combined with a controlled transmitter power ramp up.

The 24 symbol *frame sync* sequence is used as a position reference from which all subsequent symbols can be parsed and processed correctly. It can be detected by a sliding correlation of the incoming data stream against the known, fixed frame sync sequence. It is rebroadcast periodically allowing late entry into the RDLAP data stream.

The *status symbols* allow remote stations to determine when the channel is idle and may attempt to send data to the base station. The remote unit symbol sync transmission sequence needs to commence sometime in window halfway through Micro-Slot 2 and the end of Micro-Slot 3.

The *station ID* broadcasts three octets of system/domain/station ID data and 6 bits of CRC information. Please refer to the "RDLAP Error Correction" section for an example.

The *header block* and any subsequent *information blocks* each consist of three micro-slots and three status symbols. The focus of this section is how these blocks are used to transmit data. A Micro-Slot consists of 22 Data Symbols Plus One Status Symbol; a block consisting of three micro-slots consists of 69[10] symbols which transmit 12 data octets using Forward Error Correction (FEC) and interleaving. The function of the actual data in each block is treated at a higher level in the "RDLAP Data Block Structures" section.

Once all necessary information blocks have been transmitted we move back to the frame sync state. If there is no data to transmit then one continuously transmits the 24 symbol frame sync + 1 symbol status + 22 symbol station ID + 1 status symbol + a single 69 symbol idle header block: 117 repeating symbols which form prominent autocorrelation peaks useful for automatically identifying RDLAP signals[11].

- - - -

At this point we turn our attention to extracting data from the header block. We start with the three micro-slots from the above example while being cognizant of the following points:

- The data interleave is spread out over all three micro-slots.
- The status symbols do not participate in the interleave or error correction and are stripped out
- From this point onwards all symbols are are processed in pairs. Each symbol pair can be considered a single constellation point containing three bits of information and one bit of redundant error correction information.
-

So we take the above example data, drop the status symbols, and come up with 33 symbol pairs. For greater clarity we highlight the first micro-slot symbol pairs in different colors:

```
-3+3 -3-3 -1+1 -1+1 +1+3 -1-1 +3-3 -3-3 +3-3
+3+1 +1+3 -1+1 -1+1 -1-1 +1-1 -3-3 +3-3
+3+3 +3+3 -1+1 -1+1 +3-1 +1-1 +1-1 -3+1
+1-3 -1+1 -1+1 -1+1 +3-1 +1-1 +3-3 -1-1
```

10 We like to think there is a deeper meaning to this... Cf. http://en.wikipedia.org/wiki/69_%28sex_position%29

11 e.g. the peak spacing suggests a symbol rate against which the frame sync sequence can correlated for confirmation. This processing be done on the raw, undemodulated signal.

We have arranged the data in 4 rows of 9, 8, 8, and 8 symbol pairs respectively. Incoming interleaved data enters horizontally going from left to right; de-interleaved data is read out vertically from the top down. We read out the following de-interleaved data stream:

-3+3 +3+1 +3+3 +1-3 -3-3 +1+3 +3+3 -1+1 -1+1 -1+1 -1+1 -1+1 -1+1 -1+1 -1+1
+1+3 -1-1 +3-1 +3-1 -1-1 +1-1 +1-1 +1-1 +3-3 -3-3 +1-1 +3-3 -3-3 +3-3 -3+1 -1-1 +3-3

It is readily apparent how to reverse this process to create interleaved data for data transmission: arrange transmit data top down and read interleaved data out left to right.

We are almost ready to extract the information bits! But first a few comments on the error correction...

The starting state should be set to all zeros; and there will also be a final set of all zero flush tribits. The error correction method is a standard constraint length six Trellis Coded Modulation (TCM) scheme. The TCM is rate ¾ where three information are transmitted with one redundant bit; this is why receive processing uses symbol pairs.

Going with the flow we present a table of allowed state transitions (or trellis as it were). To be useful in this example the table is set as a function of the previously data tribit and the currently received symbol pair; the current data tribit can then be directly read out out from the table. This table is useful only for example purposes or extremely good signal to noise ratios. The problem is that the data is chained and very much depends on the (correctly received) previous state. So for any practical purposes a Viterbi decoder should be implemented to handle the inevitable receive problems (refer to discussion in next section).

		Current Data Tribit As Function of previous data and current received symbol															
		Current Symbol															
		-3-3	-3-1	-3+1	-3+3	-1-3	-1-1	-1+1	-1+3	+1-3	+1-1	+1+1	+1+3	+3-3	+3-1	+3+1	+3+3
Previous Data	000	---	010	---	001	110	---	101	---	---	000	---	011	100	---	111	---
	001	---	000	---	111	100	---	011	---	---	110	---	001	010	---	101	---
	010	100	---	111	---	---	000	---	011	110	---	101	---	---	010	---	001
	011	010	---	101	---	---	110	---	001	100	---	011	---	---	000	---	111
	100	000	---	011	---	---	100	---	111	010	---	001	---	---	110	---	101
	101	110	---	001	---	---	010	---	101	000	---	111	---	---	100	---	011
	110	---	110	---	101	010	---	001	---	---	100	---	111	000	---	011	---
	111	---	100	---	011	000	---	111	---	---	010	---	101	110	---	001	---

Table 2: Trellis transition table, function of previous data tribit and current received symbol pair. Based off of U.S. Patent 6,700,938.

Forbidden transitions are shown as “---”. If a forbidden transition is encountered then a transmission error has occurred and the error correction procedure outlined in next section must be carried out.

Please note the least significant (rightmost) bit does not participate in the coding; if the second symbol of the symbol pair is -3 or -1 it will always be zero; if +1 or +3 then it will always be 1.

And now we are finally ready to extract the data. Starting with state “000” for the initial tribit and using Table 2 we get:

```
-3+3 +3+1 +3+3 +1-3 -3-3 +1+3 +3+3 -1+1 -1+1 -1+1 -1+1 -1+1 -1+1 -1+1 -1+1 -1+1
001  101  011  100  000  011  111  111  111  111  111  111  111  111  111  111

+1+3 -1-1 +3-1 +3-1 -1-1 +1-1 +1-1 +1-1 +3-3 -3-3 +1-1 +3-3 -3-3 +3-3 -3+1 -1-1 +3-3
101  010  010  010  000  000  000  000  100  000  000  100  000  100  011  110  000
```

Now strip out the final flush tribit (must be zero otherwise an error occurred!) and reformat the data into twelve octets:

```
00110101 11000000 11111111 11111111 11111111 11111111
10101001 00100000 00000000 10000000 01000001 00011110
```

Or after converting to Hex:

```
35 C0 FF FF FF FF A9 20 00 80 41 1E
```

All subsequent data blocks are received the same way.

The RD-LAP Rate ¾ Trellis Coded Modulation Error Correction

(Using Station ID slot as an example)

Public References

U.S. Patent 6,700,938 describes the rate ¾ error correction used in RD-LAP.

It should be noted that the same error correction appears used in the APCO-25 Packet Data Transmission Unit; in fact that part of the APCO-25 standard bears a distinct RD-LAP heritage with not just the error correction coding but also the block structure and fields.

For more a more general introduction to Trellis Coded Modulation refer to:

http://www.complextoreal.com/tutorial.htm

http://www.complextoreal.com/chapters/tcm.pdf

The RD-LAP error correction can be summarized in a nutshell: Use the Viterbi algorithm based on the allowable trellis transitions (Table 5). Those with a low tolerance of inane banter should directly refer to U.S. Patent 6,700,938 which is the primary reference source for the material presented in this section.

RD-LAP uses rate ¾ Trellis Coded Modulation (TCM) for error correction. As seen in the previous section three data bits are transmitted over the air with an additional redundant information bit as one of 16 constellation points. A constellation point is actually specified by a pair of sequentially transmitted symbols.

We state here (but do not prove) that the error correction can only handle errors that perturb the received symbol pair (i.e. one constellation point) to the nearest neighbors in the symbol constellation. Up to two consecutive errors of this type are correctable. More severe errors are not correctable. The following table shows an example of which received constellation points are correctable for a transmitted symbol of +1-1:

Signal Constellation

		2nd Symbol			
		-3	-1	+1	+3
1st Symbol	+3	X	↓	X	X
	+1	→	OK	←	X
	-1	X	↑	X	X
	-3	X	X	X	X

Table 3: Example error correction ability for transmitted symbol +1-1.

Green = Correct Symbol

Yellow = Correctable Error (Nearest neighbors)

Red = Uncorrectable error

Why does this occur? It is a consequence of the TCM. The redundant error correction information is used to select one of two otherwise independent constellation partitions (see Table 4). Errors caused by incorrectly decoded symbols that go to the nearest neighbors (shown above in yellow) are recognized as bad – the expected partition set is wrong! Worse errors tend to make it more difficult to determine what went wrong since among other things the correct partition may show up again which may make the erroneously received symbol sequence at this particular point look correct; in any case the erroneously received symbol sequence will more closely match an incorrect symbol stream than the correct one. This represents one of the primary goals of TCM - concentrate error correction ability on the most likely errors at the expense of less likely errors.

The first TCM set partition used to encode redundant error correction information is shown below. Note that all nearest neighbors to a given constellation point come from the opposite partition set.

Signal Constellation
First Partition

		2nd Symbol			
		-3	-1	+1	+3
1st Symbol	+3				
	+1				
	-1				
	-3				

Table 4: First partition of constellation using redundant error correction bit.
The two partitions are shown colored in salmon and violet.
(Apologies to the salmon/violet colorblind).

Now the only problem is determining which sequence of symbols was most likely to have been transmitted. This is where the Viterbi algorithm comes to our rescue. This algorithm returns the most likely allowed data stream that matches the incoming data stream.

Some TCM error correction abilities will be illustrated below with an example. Relatively small disturbances can be handled gracefully while larger disturbances exceed the error correcting abilities and must be handled by other mechanisms such as message acknowledgment and retransmission protocols. The ability to correct small disturbances is a perfectly valid design judgment for protection against benign processes such as Additive Gaussian White Noise (AWGN) but not against deliberate interference or jamming[12].

12 An intentional jammer would take advantage of the transmission format and the assumptions behind the error correction design. In an RD-LAP system an optimally crafted pulsed waveform with less power than an actual user can ensure a near unity probability of at least one uncorrectable error per data block resulting in severe system disruption.

An Example:

We start with the Station ID slot from the previous section and format into symbol pairs. Interleaving is not used. Assuming all symbols are received correctly we have the following data stream:

+1-1 +1-1 -3-1 -1-1 +1-1 +3-3 -3-3 -3+3 +3+1 +1+1 -1-3

Note that the constellation points (symbol pairs) are colored either salmon or violet to indicate which partition set the constellation point belongs to. The transmitted data tribits can then be extracted:

000 000 010 000 000 100 000 001 101 111 000

Taking a brief excursion we note the received data gives us the following system identification nformation:

System ID	Domain ID	BASE ID	CRC0 (6 Bits)	Flush
00000001	00000001	00000001	101111	000

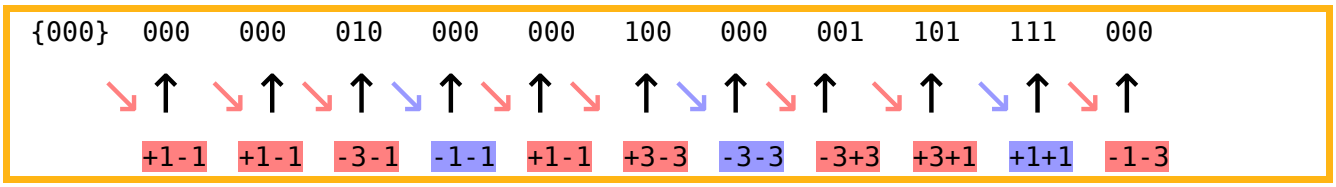
So how can we verify the above sequence was correctly received? We rewrite the trellis transition table from the previous section (table 1) as a function of previous and current data tribits. We then color code the emitted symbol pairs according to which partition they lie in to produce the following table:

Emitted Symbol Pair As Function of Previous, Current Data Tribits

Previous Data Tribit \ *Current Data Tribit*	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
000	+1-1	-3+3	-3-1	+1+3	+3-3	-1+1	-1-3	+3+1
001	-3-1	+1+3	+3-3	-1+1	-1-3	+3+1	+1-1	-3+3
010	-1-1	+3+3	+3-1	-1+3	-3-3	+1+1	+1-3	-3+1
011	+3-1	-1+3	-3-3	+1+1	+1-3	-3+1	-1-1	+3+3
100	-3-3	+1+1	+1-3	-3+1	-1-1	+3+3	+3-1	-1+3
101	+1-3	-3+1	-1-1	+3+3	+3-1	-1+3	-3-3	+1+1
110	+3-3	-1+1	-1-3	+3+1	+1-1	-3+3	-3-1	+1+3
111	-1-3	+3+1	+1-1	-3+3	-3-1	+1+3	+3-3	-1+1

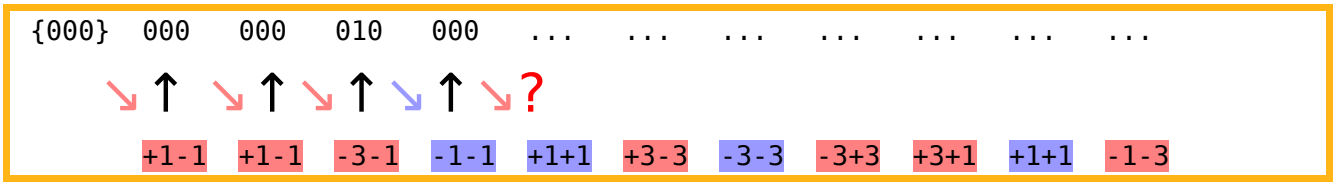
Table 5: Trellis transition table for RD-LAP data transmission: Emitted Symbol Pair As Function of Previous, Current Data Tribit; emitted symbol pair color coded according to constellation partition. Please note that only the first two bits of the previous data tribit determine which partition the emitted symbol belongs to.

We observe that if the previous data tribit starts with bits 00 or 11 the emitted symbol ends up in the same partition; and if the previous data tribit starts with 01 or 10 we end up in the other partition. We can use this observation to check a received data tribit against the next symbol pair's partition set. Remembering the initial state decoder state is 000 we get something like:



The colored arrows represent the expected partition set of the next constellation point. The black arrows represent the decoded data. The initial state is represented by {000}. Nothing fishy here – everything matches up as expected!

Now let us suppose that in the above datastream the +1-1 point was incorrectly received as +1+1, one of the nearest neighbors to the correct symbol. We start decoding data until we run into a problem – the expected partition membership does not match up:



At this point we know that the received symbol sequence is incorrect. The challenge now is to find that allowed sequence which most closely matches the received sequence which can be easily done with a Viterbi Decoder implementation[13]. In our example above the correct symbol stream will be easily recovered with a minimal error metric when the incorrect +1+1 constellation point is evaluated with +1-1 instead.

We start running into problems though if a more serious error with an incorrectly decoded constellation point not a nearest neighbor to the correct one occurs. For example, if the +1-1 were incorrectly received as -1+1 then at that point the expected and actual symbol partitions match; the Viterbi algorithm will actually go and try to correct the wrong following constellation point making an unrecoverable mess.

13 http://en.wikipedia.org/wiki/Viterbi_decoder

RDLAP PDU & SDU Data Block Structures

Public References

This entire section is based on information and diagrams from:

http://www.rapa.or.kr/data/book_issue_view.asp?idx=49

At this point we are still one layer below application specific data. We only concern ourselves with how application data is being transmitted, not what the application data is. For this we are interested in the PDU (Protocol Data Unit) structure. This represents the mechanism through which higher level application data is transferred via the RD-LAP protocol.

This section also briefly covers some SDU (Service Data Units) packets that handle administrative tasks like channel assignments.

First we discuss the main PDU structure used for transmitting higher level (i.e. application) data. There are two main types of packets: confirmed service where the recipient must acknowledge successful PDU reception and an unconfirmed service where one hopes all recipient(s) receive the data. Since the confirmed service only can send data to a single recipient one should expect to sometimes see the same piece of data being transmitted again and again to multiple recipients if the message is important enough to warrant receipt acknowledgment. This can be counterbalanced by the use of unconfirmed messages that are periodically transmitted to many users at once if the information is somewhat less critical and it doesn't matter if a message or two is missed.

Various CRCs are used to verify correct data reception. The CRCs are summarized in Appendix A.

The following pages describe the common PDU data types encountered while transmitting higher level application specific data:

PDU DATA FORMAT For Application Data

		Bit Field 7	6	5	4	3	2	1	0
HEADER BLOCK OCTET	0	0	A/N = 1	I/O	FORMAT = 10101 CONFIRMED; 00011 UNCONFIRMED				
	1	1	1	SAP IDENTIFIER					
	2	1	LOGICAL LINK IDENTIFIER						
	3								
	4								
	5								
	6	1	BLOCKS TO FOLLOW						
	7	NOT USED (00)		PAD OCTECT COUNT				SSN	
	8	0	N(S): PDU SEQ. NUMBER (CONFIRMED SERVICE ONLY)			FSNF – SDU FRAGMENT SEQ. NUMBER FIELD (CONFIRMED SERVICE ONLY)			
	9	SYN	0	DATA HEADER OFFSET					
	10	HEADER CRC1 (16 Bits)							
	11								
Intermediate Blocks	0	SDU DATA (APPLICATION SPECIFIC)							
	1								
	2	DATA HEADER IF PRESENT (Application Specific)							
	3	- -							
	4	SDU DATA, After DATA HEADER BYTES							
	5								
	6								
	7								
	8								
	9								
	10								
	11								
Final Block	0								
	1								
	2								
	3								
	4								
	5	- -							
	6	Between Zero And Eleven PAD Octets (0xAA)							
	7	Note: Up to 3 Pad octets may extend into last intermediate block							
	8	CRC2 (32 Bits)							
	9								
	10								
	11								

A/N Bit: 1 = PDU reception should be acknowledged

0 = No acknowledge (e.g. group broadcast)

I/O Bit: 1 = Outbound message (Base to remote unit)

0 = Inbound (Remote Unit to base)

Format: 10101 = Confirmed PDU

00011 = Unconfirmed PDU

11101 = Idle PDU (Format shown further down)

SAP ID: Denotes what sort of data is being transmitted in PDU

Data PDU SAP Identifiers

	SAP Value	Function
Client	000000	Client Data
Control	100000	Registration & Permission
	100001	Go To Channel
	100011	RPM Loop-back
	100100	RPM Statistics
	100101	RPM Out Of Service
	100110	RPM Paging
	100111	RPM Configuration
	101100	Channel Marker – Lists all channels used in system (12.5KHz resolution)
	111111	Vendor Specific

Logical Link Identifier: Unique Radio Physical Address

Blocks To Follow: Number of additional blocks in PDU (not counting header)

Pad Octet Count: Number of appended pad (0xAA) Octets

SSN: SDU Serial Number

N(S) PDU Sequence Number. Increments by one for each new PDU; allows checking that all transmitted PDUs in sequence have been received & detect missing PDUs

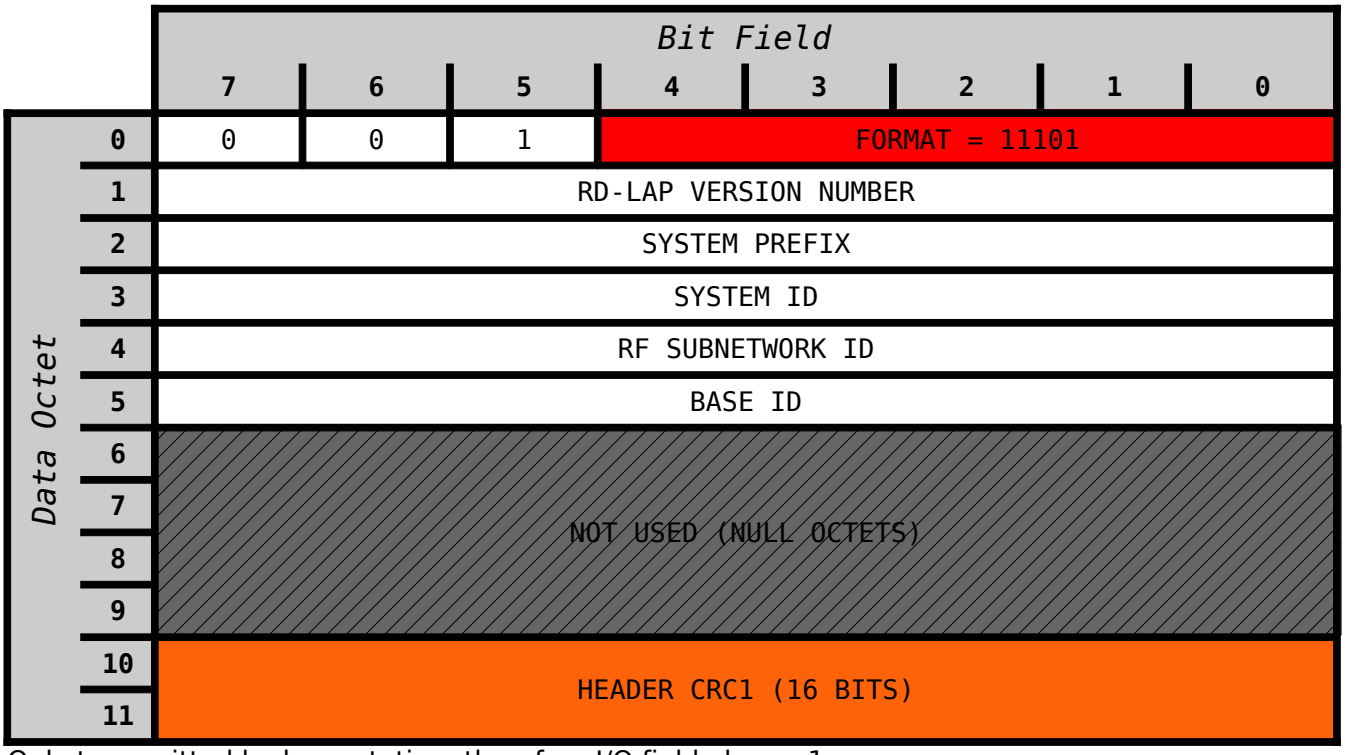
SYN 1 = Start of PDU

0 = Non-first PDU

NOTE: CONFIRMED PDU may continue sending. i.e. after first PDU subsequent data continues at the first byte of BLOCK 1.

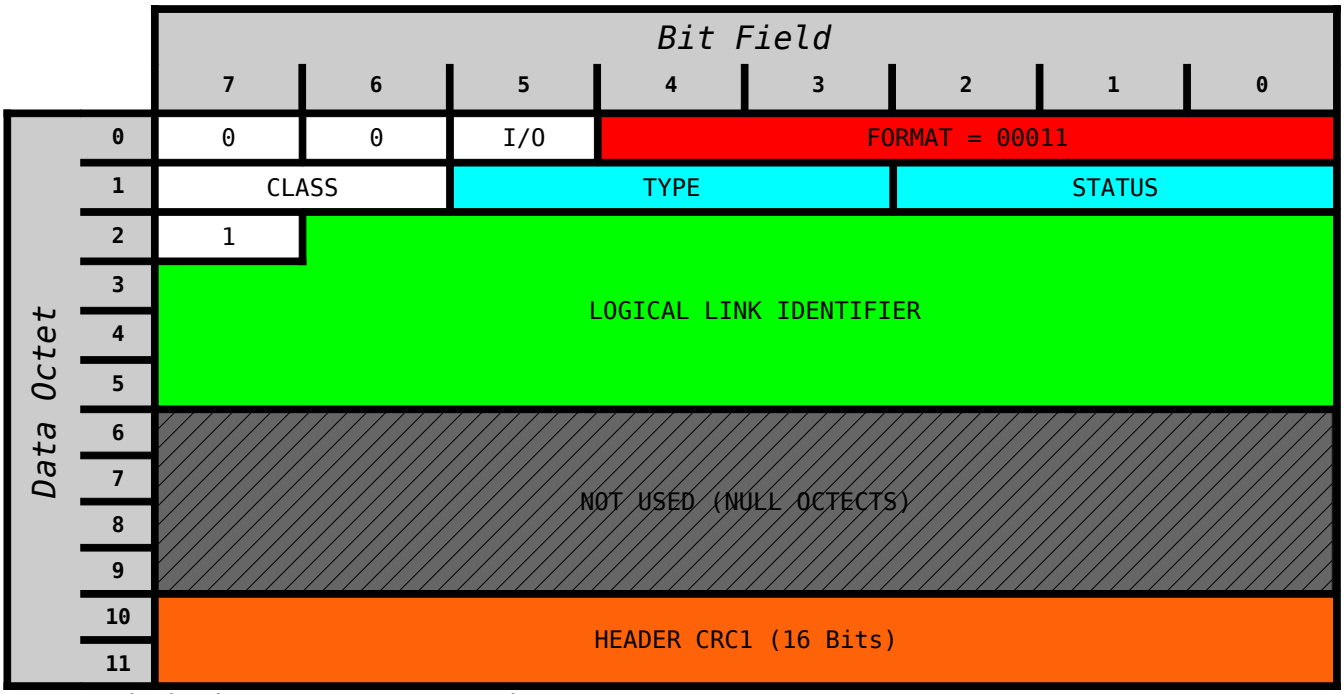
Unconfirmed may restart the header (FF addresses).

IDLE PDU DATA FORMAT



Only transmitted by base station; therefore I/O field always 1

RESPONSE DATA PDU FORMAT



Remote device has I/O = 0; Base Station = 1

Putting It All Together With An Example Message

Public References

Internet Standard RFC 1951 (DEFLATE Algorithm). Numerous online descriptions (see e.g. http://www.ietf.org/rfc/rfc1951.txt) and implementations (http://www.zlib.net/).

At this point we have finished RD-LAP specific processing and have received raw user level application data. This section demonstrates the application specific data format used in the Huntsville/Madison County area MDT system.

As a representative example of something that might be seen on this system let us suppose we have receive the follow raw RD-LAP traffic:

Example RD-LAP PDU Message

55 C0 80 12 34 56 8A 2B 78 00 22 BF	RD-LAP Header Block
00 66 00 00 80 12 34 56 82 02 00 00	1st Data Block
00 01 01 23 78 9C 4D 8A 3B 0A 80 30	2nd Data Block
10 05 4F 63 A3 04 DE 7E B2 BB 21 6A	
25 48 40 2D BC FF 61 4C 3A A7 7A C3	
9B 76 07 06 D3 B9 BF D7 13 6E 59 BB	
2D A2 CE 26 AE 92 10 96 23 84 84 96	
7E B8 F1 08 B4 24 35 10 89 1B 32 08	
0C 81 F6 65 70 04 0A 7E 08 D7 76 6C	
80 95 3A 6B 5E 3F 25 D9 16 FB AA AA	
AA AA AA AA AA AA AA AA 7C 14 DB F9	Final Data Block + CRC2

We reformat the received data to better reflect the application level formatting:

PDU Data Field Format – User Application Level		
Format	**Data**	**Use**
RD-LAP	55 C0 80 12 34 56 8A 2B 78 00 22 BF	RD-LAP Header. Described in previous sections
PDU Data Field	00 66	Application Data Octets count Data may be broken up over multiple PDUs as needed.
	00 00	Not included in octet count!
0x66 Data Octects	80 12 34 56	Logical Link ID (Identical to RD-LAP LLID). Redundant – always matches RD-LAP header link ID
	82	Additional Flags Giving Message Direction, Format
	02	Compression Flag 00 = No Compression; 02 = DEFLATE Compression a la RFC 1951
	00 00	
	00 01	Total number of PDUs needed to transmit data packet. (Non-Acked; Acked may have this always set to 0x0001)
	01 23	Application Sequence Number
	78 9C 4D 8A 3B 0A 80 30...	Application Data
RD-LAP	AA AA AA AA...	RD-LAP Pad Octets
	7C 14 DB F9	RD-LAP CRC2 Over PDU Data Field (Including Pad Octets)

The RD-LAP header block indicates a total of ten data blocks the header. There are also ten filler bytes at the end (the 0xAAs). This message is going from remote unit address 0x80123456 to the base and this is an ACKed message.

All checksums verify correctly so we proceed with the processing of the data blocks.

Note that the first data block repeats the remote unit ID address (application level Ids are identical to RD-LAP physical layer Ids).

Please note that 789C signature – this always starts off a message compressed with the RFC 1951 DEFLATE algorithm set for the maximum 32K dictionary size. This signature appears all over the place in diverse systems because when a developer has been told “Add compression” chances are he/she will settle on the DEFLATE algorithm due to good documentation, no IP or patent encumbrances, and freely usable open source implementations. And in this day and age one generally selects the maximum compression level producing the 789C signature[14].

It should also be noted that the tenth byte of the first data block is set to 0x02 which indicates that compression is active. So although the 789C strongly suggests RFC 1951 compression is used it is better practice to use the actual compression flag – sometimes different compression settings will be used.

The ninth byte of the first data block has flags set indicating this message originates from the remote unit.

Additional flags act as sequence numbers.

14 As a general comment, characterizing these types of signatures is a somewhat neglected art. The Malware / Virus scanning industry is probably the best publicly known application. But this concept can obviously be extended to on the fly detection and characterization of compression algorithms, cryptographic algorithms, and the like.

Starting at 78 9C and we pass the data through a RFC 1951 Deflate compliant decompression code. No need to write it yourself, c.f. the zlib library page. We end up with:

```
IM800000%G>RLN87654000+347263743-0865883131+000762540049-
460113760501020304050607080900000000000032;ID=0069;*45<
```

Well, we seem to have an AVL (Automatic Vehicle Location) report using the TAIP (Trimble ASCII Interface Protocol) which is publicly documented in many of the Trimble GPS manuals[15]. Parsing things out:

IM800000%G	NOT PART OF TAIP Report. Seems to be some sort of ID code
>	Start of new message TAIP delimiter
R	Message Qualifier R : Query or scheduled report response
LN	Message Identifier LN: Long Navigation Message
87654000	GPS Time of day
+347263743	Latitude : 34.7263743 N
-0865883131	Longitude: 86.5883131 W
+00076254	Altitude above mean sea level: 762.54 Ft
0049	Horizontal Speed: 4.9 MPH
-4601	Vertical Speed: -460.1 MPH
1376	Heading: 137.6 Deg
05	Number of SVs used
0102	1st SV ID + IODE
0304	2nd SV ID + IODE
0506	3rd SV ID + IODE
0708	4th SV ID + IODE
0900	5th SV ID + IODE
0000000000	Reserved
3	Source; 3 Indicates 3D DGPS fix
2	Data Freshness Indicator: 2 indicates < 10 sec old
;ID=0069	Optional Unit ID
;*45	Optional TAIC checksum
<	End Delimiter

So what have we learned? Unit 69 is screwed. It appears to be in the terminal phase of a trajectory approaching ground level at a rate of 460.1 MPH with just a slight southwest drift of 4.9 MPH. It was never this bad on the Dukes of Hazzard! Oh unit 69... we hardly knew ye.

Disclaimer: This is a hypothetical example as a real unit is somewhat unlikely to end up in this type of situation. The message format itself is correct.

All other traffic on this MDT system is similarly extracted.

15 Example at http://gpsd.berlios.de/vendor-docs/trimble/trimble-ace2.pdf

Appendix A - RDLAP CRC Specifications

CRC0 Specification (For Station ID Slot):

- Length: 6 bits
- CRC Polynomial: $x^6 + x^4 + x^3 + 1$ (Normal Representation 0x19)
- Initial State: 0x00 (Different From CRC-CCITT)
- Final Result: Inverted
- Test Vector: The 9 byte ASCII string "123456789" produces 0x3B

CRC1 Specification (For Header Block):

- Length: 16 bits
- CRC Polynomial: $x^{16} + x^{12} + x^5 + 1$ (Normal Representation 0x1021)
- Initial State: 0x0000 (Different From CRC-CCITT)
- Final Result: Inverted
- Test Vector: The 9 byte ASCII string "123456789" produces 0xCE3C

Note: This CRC is similar but not identical to CRC-CCITT.

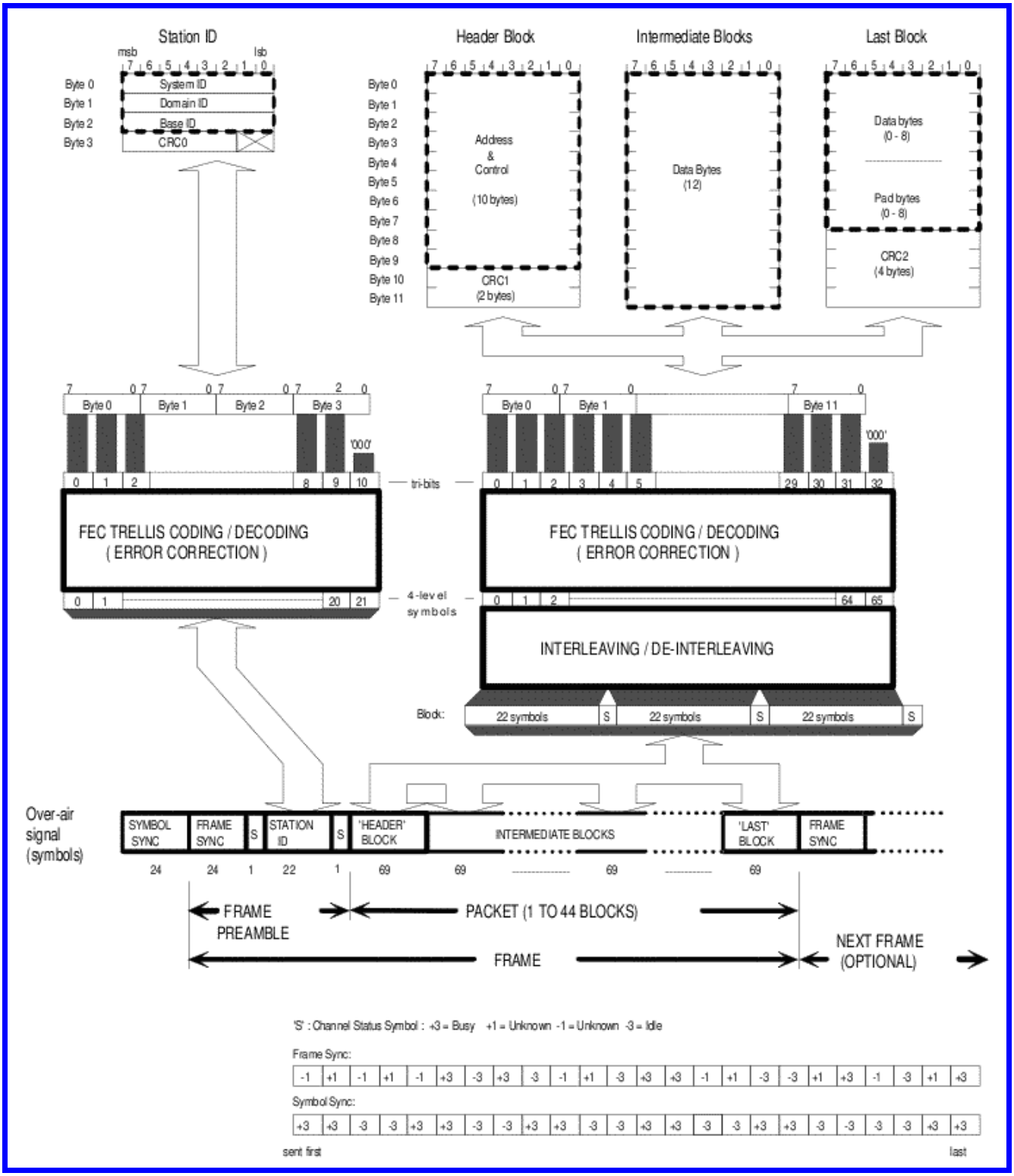
CRC2 Specification (For Intermediate/Final PDU Blocks, Excludes Header Block):

- Length: 32 bits
- CRC Polynomial: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
  (Normal Representation 0x04C11DB7)
- Initial State: 0x00000000
- Final Result: Inverted
- Test Vector: The 9 byte ASCII string "123456789" produces 0x765E7680

Note: CRC-32/POSIX

Appendix B - RDLAP Over The Air Interface Illustration

From Consumer Microcircuits Limited FX929B Datasheet



COLOPHON