Cellular Digital Packet Data (CDPD) Equipment: Some Practical Design Issues

Gregory D. Bump
Grayson Electronics Company
Blacksburg Research and Development Center
Corporate Research Center
Blacksburg, VA 24061

Theodore S. Rappaport
Mobile and Portable Radio Research Group
Bradley Department of Electrical Engineering
Virginia Tech
Blacksburg, VA 24061-0350

Abstract—Cellular Digital Packet Data (CDPD) is a new wireless packet data communication system which was developed by a consortium of U.S. cellular service providers to augment cellular voice communications. CDPD provides wireless packet data connectivity to mobile data communications customers. This paper presents an introduction to CDPD networks, various hardware interface challenges, and a software approach to implementing the (63,47) Reed-Solomon error correction code.

I. INTRODUCTION

THE largest cellular service providers in the United States banded together in 1993 to develop a new wireless packet data communications system to augment their voice communications systems. The Cellular Digital Packet Data (CDPD) network, the result of this effort, is currently being deployed in many of the largest U.S. cellular markets. CDPD is a packet-switched network that allows a mobile customer to access existing data networks using a cellular radio communications link. CDPD overlays an existing Advanced Mobile Phone Service (AMPS) system, but it uses AMPS capacity purely on a secondary, non-interfering basis [1]. A “sniffing” receiver in the CDPD base station is able to detect activity on any voice channel, allowing the base station to determine when a channel is available for use by CDPD. In this manner, a data link between CDPD and other systems is not required [2].

When there is contention for a voice channel by the AMPS and CDPD systems, AMPS has priority. Studies have shown, however, that at least 30% of the air time on a given voice channel is unused, even during peak hours [3]. CDPD is intended to make use of this “lost” time on a voice channel for data transfer, effectively increasing the capacity of the service provider’s system. CDPD uses 19.2 kbps Gaussian filtered Minimum Shift Keying (GMSK) modulation, Reed-Solomon error control, and provides Connectionless Network Services (CLNS) through the Connectionless Network Protocol (CLNP) [5]. Initially, CDPD will support the “IP” portion of Transmission Control Protocol/Internet Protocol (TCP/IP).

Several manufacturers, including AT&T, Motorola, PCSI, and Hughes, have released CDPD base station and subscriber equipment. Service providers around the country are deploying this equipment on various scales. Large potential customers, such as Federal Express and United Parcel Service (UPS), have committed to working with the service providers during some of these trials [5]. Commercial service should become available in 1996.

II. PHYSICAL LAYER OF THE CDPD AIRLINK

The RF link between a base station and a group of mobile stations is called the CDPD AirLink. The AirLink transmits and receives a sequence of bits at 19.2 kbps as a binary Gaussian Filtered Minimum Shift Keying (GMSK) waveform. This waveform is transmitted across the same 30 kHz voice channels used by AMPS cellular telephone systems and over the same frequency-duplexed pair of RF channels, with the base station transmitting 45 MHz above the mobile unit transmit physical channel layer. Additional services provided at the CDPD Airlink include the ability to tune to a specific RF channel pair, set the transmit power level, measure the received power level, and suspend and resume the monitoring of RF channel pairs in the mobile station to support power conservation [1].

CDPD uses the channel numbering scheme employed in the current AMPS/IS-54 U.S. cellular system. Those channels which are designated as voice channels can be used for CDPD transmissions only when they are not being used for voice communications. The 42 primary and 42 secondary control channels used in IS-54 are unavailable for use by CDPD. Channels which are allocated for CDPD use are classified as either dedicated or non-dedicated channels. A dedicated channel is always in use or available for use by CDPD; the voice system is not permitted to use that channel (in that particular cell or sector). A non-dedicated channel may only be used for CDPD when it is not in use by the voice system [1].

A. Hardware Implementation

Several hardware issues regarding the physical layer of the AirLink needed to be addressed in the development of the Grayson Electronics CDPD transceiver. These include the digital hardware interface, transmitter power rampup and rampdown, and GMSK waveform accuracy.

A.1 Digital Hardware Interface

A commercially available GMSK modem chip was used in the design of the Grayson Electronics CDPD transceiver that was the subject of this research. The MX-COM

IEEE ICUPC '96, Cambridge, MA, September 29 - October 2, 1996
MX589 contains nearly all of the modulator and demodulator circuitry required to implement GMSK for $B_s T = 0.3$ and 0.5, which correspond to the specified values for GSM and CDPD, respectively. ($B_s$ is defined as the 3dB filter bandwidth of the Gaussian-shaped filter and $T$ is the data symbol time duration).

A block diagram of the hardware design of the Grayson Electronics mobile module is shown in Fig. 1. The 8051 micro-controller is the source of transmitted data which passes through a 16-bit shift register in a XILINX XC3090 Field Programmable Gate Array (FPGA). The FPGA generates an interrupt for the 8051 at a rate of 1200 Hz (1/16th of the baud rate 19.2 kbps) which causes the 8051 to output 16 bits of transmit data to a register in the FPGA. These 16 bits are shifted out serially along with a 19.2 kHz clock signal to the MX589 transmitter. A TMS320-C50 Digital Signal Processor (DSP) receives data from and provides hardware control for the MX589 GMSK receiver via the same XILINX FPGA. Received data bits are serially shifted into a 16-bit shift register in the FPGA and then transferred to the DSP via another 1200 Hz interrupt. With this hardware configuration, a full-duplex CDPD GMSK modem has been implemented.

A.2 Transmitter Power Rampup and Rampdown

When a CDPD mobile unit performs a channel hop, it must gradually decrease its transmit power on the current channel, hop to the new channel, and gradually increase its transmit power from the powered-down state so that spectral “splattering” is minimized. The CDPD System Specification states that the mobile unit may take no more than 2 msec for both the ramp-up and ramp-down transitions.

The rampup and rampdown processes are triggered by a single logic line controlled by the Intel 8051 processor in the Grayson Electronics CDPD transceiver. When a complete data packet has been encoded and is ready to be transmitted, the 8051 pulls the line HI, which in turns triggers a rampup process. When the packet has been completed the 8051 pulls the line LO and continues to transmit a dotting sequence while the rampdown process occurs.

A.3 GMSK Waveform Accuracy

The two main areas of concern with the GMSK waveform are the modulation index accuracy and the data rate accuracy. The CDPD System Specification states that the modulation index $h$ must be 0.3 ± 5% and that the data rate must 19.2 kbps with an accuracy of 50 parts per million. One of the main reasons that we selected the MX-COM MX589 GMSK modem IC is that its modulation index can be adjusted externally and that its baud rate accuracy is determined by the accuracy of the external clock to the chip. Engineers at Grayson Electronics have investigated the modulation index accuracy of the modem and have selected appropriate external components which yield the best results. The baud rate accuracy was achieved by digitally deriving a 19.2 kHz clock signal from the highly accurate 11.0592 MHz crystal clock used for the Intel 8051.

III. REED-SOLOMON ERROR CORRECTION CODES

This section describes the use of the (63,47) Reed-Solomon (RS) error correcting codes in the CDPD network. We implemented a RS encoder in the Intel 8051 micro-controller, and a RS decoder in TMS320-C5x assembly language for the Grayson Electronics CDPD transceiver system. The (63,47) RS code used in CDPD accepts $k = 47$ message symbols and appends $n - k = 16$ parity symbols to create a block of $n = 63$ symbols. The code symbols contain $m = 6$ bits per symbol and can correct up to $t = 8$ symbol errors.

RS codes make use of non-binary fields $GF(2^m)$, where $GF$ is a Galois Field. These fields have more than 2 elements (all of which are unique) and are extensions of the binary field $GF(2) = \{0,1\}$. Since the $GF(2^m)$ field is non-biary, a new symbol $\alpha$ is introduced, and each nonzero element in the field can be represented as a power of $\alpha$.

Finite field arithmetic can be implemented efficiently in software with two lookup tables. One table, called the alpha table, answers the question “what is the value of $\alpha$ raised to the power $x$?” The second table, called the power table, answers the question “what is the power to which $\alpha$ must be raised to arrive at a given field element?” The tables can be interpreted as $\alpha^x = \alpha^y$ and $\text{power}[x] = \log_{\alpha}(x)$, respectively.

A. Reed-Solomon Encoding

The encoded RS polynomial can be expressed by [10]

$$c(x) = d(x) + p(x) = \sum_{i=0}^{n-1} c_i x^i ,$$

where

$$d(x) = c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \cdots + c_{2t+1} x^{2t+1} + c_{2t} x^{2t}$$

is the information polynomial containing $k$ input message symbols, with each $c_i$ an element of $GF(2^m)$, and

$$p(x) = c_{2t-1} x^{2t-1} + \cdots + c_1 x + c_0$$

is the error polynomial which contains $t$ error symbols.
is the parity polynomial which contains the 2t parity symbols.

A vector of n field elements \( (c_0, c_1, \ldots, c_{n-1}) \) is a code word if and only if it is a multiple of the generator polynomial \( g(x) \). The generator polynomial for a \( t \)-error correcting RS code has the following form [10]:

\[
g(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{2t}).
\] (4)

A common method for encoding a cyclic code is to derive \( p(x) \) by dividing \( d(x) \) by \( g(x) \). This yields a quotient polynomial \( q(x) \) and a remainder polynomial \( r(x) \) [11]; thus

\[
d(x) = g(x)q(x) + r(x).
\] (5)

The codeword polynomial can be expressed as

\[
c(x) = p(x) + g(x)q(x) + r(x),
\] (6)

which, if we set \( p(x) = -r(x) \), gives

\[
c(x) = g(x)q(x).
\] (7)

Having ensured that the codeword polynomial is a multiple of \( g(x) \), the division of \( d(x) \) by \( g(x) \) to obtain \( p(x) \) can be accomplished with a shift register as shown in [11]. We chose instead to implement the encoder in software. First, the generator polynomial \( g(x) \) is built through a simple software loop that implements (4). For the (63,47) RS code, the generator polynomial is

\[
g_{63,47}(x) = x^{16} + \alpha^{28} x^{15} + \alpha^{41} x^{14} + \alpha^1 x^{13} + \alpha^{16} x^{12} + \alpha^{24} x^{11} + \alpha^{54} x^{10} + \alpha^{32} x^9 + \alpha^{38} x^8 + \alpha^{50} x^7 + \alpha^{25} x^6 + \alpha^{12} x^5 + \alpha^{21} x^4 + \alpha^{23} x^3 + \alpha^{17} x^2 + \alpha^{21} x + \alpha^{10}.
\] (8)

The codeword is then constructed by simulating the action of the shift register in software. Symbols 62 through 16 are the input data symbols and symbols 15 through 0 are the generated parity symbols.

**B. Reed-Solomon Decoding**

Suppose that a codeword \( c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \) is transmitted and that the received codeword is \( r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1} \). The error pattern is the difference between these polynomials; i.e.,

\[
e(x) = c(x) - r(x) = e_0 + e_1 x + \cdots + e_{n-1} x^{n-1}.
\] (9)

If \( e(x) \) is nonzero, then some of the symbols were received in error. Our goal is to determine the location of each error and, since this is a non-binary code, the error magnitudes. To do this in a computationally efficient manner, we first calculate a set of partial syndromes.

Let the \( 2t \) partial syndromes \( S_i \), where \( 1 \leq i \leq 2t \), be defined as \( S_i = r(\alpha^i) \). Since each codeword is a multiple of the generator polynomial, we know that \( \alpha^1, \alpha^2, \ldots, \alpha^{2t} \) are roots of each transmitted codeword \( c(x) \). It follows that \( c(\alpha^i) = 0 \) and thus \( S_i = c(\alpha^i) + e(\alpha^i) = e(\alpha^i) \). It is clear that the \( 2t \) partial syndromes \( S_i \) depend only on the error pattern \( e(x) \) and not on the specific received codeword \( r(x) \) [10]. Furthermore, these syndromes can be found without performing time-consuming polynomial divisions.

We can also improve computational efficiency by noting that the partial syndromes

\[
S_i = r(\alpha^i) = r_0 + r_1 \alpha^i + r_2 \alpha^{2i} + \cdots + r_{n-1} \alpha^{(n-1)i}
\] (10)

can be rearranged [11] as

\[
S_i = (\cdots (r_{n-1} \alpha^{i} + r_{n-2}) \alpha^i + r_{n-3}) \cdots \alpha^i + r_0.
\] (11)

Since the (63,47) RS code used in CDPD systems has \( t = 8 \), a total of 16 partial syndromes must be calculated. Now an algorithm based only on the syndromes can be used to determine the error pattern \( e(x) \).

Suppose the error pattern contains \( v \) errors at locations \( x^{j_1}, x^{j_2}, \ldots, x^{j_v} \), where \( 0 \leq j_1 < j_2 < \cdots < j_v \leq n - 1 \). Furthermore, let the error magnitude at each location \( x^{j_i} \) be denoted as \( e_{j_i} \). Then \( e(x) \) has the form

\[
e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + \cdots + e_{j_v} x^{j_v}.
\] (12)

Now define the set of error locator numbers \( \beta_i = \alpha^{j_i}, i = 1, 2, \ldots, v \). Then the set of \( 2t \) partial syndromes becomes the following system of equations:

\[
\begin{align*}
S_1 & = e_{j_1} \beta_1 + e_{j_2} \beta_2 + \cdots + e_{j_v} \beta_v \\
S_2 & = e_{j_1} \beta_1^2 + e_{j_2} \beta_2^2 + \cdots + e_{j_v} \beta_v^2 \\
& \vdots \\
S_{2t} & = e_{j_1} \beta_1^{2t} + e_{j_2} \beta_2^{2t} + \cdots + e_{j_v} \beta_v^{2t}.
\end{align*}
\] (13)

Any algorithm which solves (13) can be used to determine the error pattern \( e(x) \) and is thus a RS decoding algorithm. The error magnitudes \( e_{j_i} \) are found directly, and the error locations \( x^{j_i} \) can be derived from the error locator numbers \( \beta_i \) [10].

**B.1 Calculation of Error Locations**

The \( 2t \) equations in the system are symmetric functions in \( \beta_1, \beta_2, \ldots, \beta_n \), known as power-sum symmetric functions [10]. Now define the error locator polynomial \( \sigma(x) \) as

\[
\sigma(x) = (1 + \beta_1 x) (1 + \beta_2 x) \cdots (1 + \beta_v x) = \sigma_0 + \sigma_1 x + \cdots + \sigma_v x^v.
\] (14)

The roots of \( \sigma(x) \) are \( \beta_1^{-1}, \beta_2^{-1}, \ldots, \beta_v^{-1} \), which are the multiplicative inverses of the error location numbers \( \beta_i \). Thus, \( \sigma(x) \) indirectly contains the locations of each of the errors in \( r(x) \). Note that \( \sigma(x) \) is an unknown polynomial whose coefficients must also be determined during the RS decoding process.

The coefficients of \( \sigma(x) \) and the error-location numbers \( \beta_i \) are related by the following equations [10]:

\[
\begin{align*}
\sigma_0 & = 1 \\
\sigma_1 & = \beta_1 + \beta_2 + \cdots + \beta_v \\
\sigma_2 & = \beta_1 \beta_2 + \beta_2 \beta_3 + \cdots + \beta_{v-1} \beta_v \\
& \vdots \\
\sigma_v & = \beta_1 \beta_2 \cdots \beta_v.
\end{align*}
\] (15)
The unknown quantities $\sigma_j$ can be related to the known partial syndromes $S_i$ by the following set of equations, known as Newton’s Identities:

\[
\begin{align*}
S_1 + \sigma_1 &= 0 \\
S_2 + \sigma_1 S_1 + 2\sigma_2 &= 0 \\
S_3 + \sigma_1 S_2 + \sigma_2 S_1 + 3\sigma_3 &= 0 \\
&\vdots \\
S_v + \sigma_1 S_{v-1} + \cdots + \sigma_{v-1} S_1 + v\sigma_v &= 0.
\end{align*}
\]

The most common method used to determine $\sigma(x)$ is an iterative algorithm that was developed by Berlekamp [13] and modified by Massey [14]. The first step in this algorithm is to find a minimum degree polynomial $\sigma^{(1)}(x)$ whose coefficients satisfy the first Newton Identity in (16) by setting $\sigma^{(1)}(x) = S_1$. Next, the algorithm tests whether $\sigma^{(1)}(x)$ satisfies the second Newton Identity; if so, then a new polynomial is defined such that $\sigma^{(2)}(x) = \sigma^{(1)}(x)$. If not, then a correction term is added to $\sigma^{(1)}(x)$ to form $\sigma^{(2)}(x)$ such that it has minimum degree and the first two identities in (16) are satisfied [10]. The procedure continues until $\sigma^{(2t)}(x) = \sigma(x)$ is found.

The inverse error location numbers $\beta_i^{-1}$, which are the roots of $\sigma(x)$, can now be found by evaluating the error locator polynomial at every element in $GF(2^m)$. If $\sigma(a^i) = 0$, then $a^i$ is a root of $\sigma(x)$. Each $\beta_i$ indicates a coefficient of the received vector $r(x)$ which is in error. For example, if $\beta_i = a^{24}$, then symbol $r_{24}$ was received in error.

B.2 Calculation of Error Magnitudes

The error magnitudes for symbols in the RS code range from 1 to $2^m - 1$, and are calculated by using the error evaluator polynomial $Z(x)$, defined as [11]

\[
Z(x) = 1 + (S_1 + \sigma_1) x + (S_2 + \sigma_1 S_1 + \sigma_2) x^2 + \cdots + (S_v + \sigma_1 S_{v-1} + \cdots + \sigma_v) x^v.
\]

The error magnitude $e_{zi}$ at each location $z^i$ is [10]

\[
e_{zi} = \frac{Z(\beta_i^{-1})}{\prod_{k=1}^{v} (1 + \beta_k \beta_i^{-1})}
\]

Now the errors in the received codeword can be corrected by adding $e(x)$ to $r(x)$. If $r(x)$ contains $t$ or fewer errors, then the result is $c(x)$, the originally transmitted codeword.

C. Reed-Solomon Decoder Implementation

A typical RS decoder uses five distinct algorithms. The first algorithm calculates the $2t$ partial syndromes $S_i$, and the second performs the Berlekamp-Massey algorithm which calculates the error locator polynomial $\sigma(x)$. Next, a root calculation algorithm is performed to obtain the specific error locations in the received word from the error locator polynomial. The fourth step in the decoding process is the calculation of the magnitude of the error at each error location. Finally, the error pattern is added to the received word to correct up to $t$ errors successfully.

RS decoders can be implemented in hardware, software, or a mix of hardware and software. Hardware implementations are typically very fast, but they cannot be used for a wide variety of RS code sizes. For example, there are several single-chip RS decoders available which decode Intelsat 8ESS-308 RS codes. These codes are commonly used in satellite communications, digital video applications, and compact disk technology. They have specific generator polynomials and operate only in GF(256) using 8-bit code symbols [12]. The hardware decoders can operate at rates up to 10 Mbytes per second, but they are not flexible enough to work with the (63,47) RS code which functions in GF(64). At the time of this writing, there are no chips available which operate on 6-bit symbols from GF(64). Fortunately, CDPD operates at the much slower data rate of 19.2 kbps, so a real-time software implementation of a (63,47) RS decoder can be achieved. A software approach may be more attractive to a CDPD system developer because of shorter development time, lower development cost, and greater flexibility. Source code listings in C that we used to construct the (63,47) RS encoder and decoder algorithms can be found in [15].

Two special problems had to be solved during the development of the (63,47) RS decoder software: how to handle a received codeword with more than $t$ errors, and how to obtain sufficient execution speed.

Two specific error conditions occur when there are more than $t$ errors in $r(x)$. One occurs when the received vector $r(x)$ appears to be a valid codeword $c'(x) \neq c(x)$ with $t$ or fewer errors. The software corrects these “errors,” resulting in the wrong codeword $c'(x)$ being output from the decoder with no indication of an error. Since there is no way to detect or correct this problem in the RS decoder, CDPD systems must identify the error with a higher level protocol and request data retransmission.

The other condition occurs when one or more of the $2t$ partial syndromes are non-zero (indicating that there are errors), but the calculated error locator polynomial $\sigma(x)$ is not factorable in GF(64). Since $\sigma(x)$ has no real roots, it is not possible to determine any error locations so no correction is possible. This condition can be detected in the RS decoder and the data in the current received codeword can be discarded, followed by a retransmission request.

The second RS decoder implementation issue encountered was developing sufficient software execution speed. Solutions involved various hardware platform changes, each of which required several software rewrites. The most computationally intense algorithms which the TMS320C50 DSP has to perform included the word synchronization routine (a sliding window correlator), the RS decoding, and raw data I/O. The RS decoder was by far the most complex to implement, so the author and engineers at Grayson Electronics began searching for a processor which best accommodated the algorithms.

Much of the time required for the RS decoding algorithms is spent calculating addresses for the finite field
coding hardware and software are functioning properly. The complete system, shown in Fig. 2, has been tested in a Hewlett-Packard assembly language for fast execution speed. This led to the need for a processor which could handle the indirect addressing of multiple operands in a quick and efficient manner. The Texas Instruments TMS320C5x series of fixed-point DSP chips has a bank of eight Auxiliary Registers which all act as pointers into data memory. The real power of this bank of registers is that the software engineer can avoid unnecessary temporary storage of pointers and data values because new addresses can be calculated in other registers as the operands from previously calculated addresses are being accessed. The 20 million instructions per second (MIPS) rate of this DSP is also very desirable. In order to accommodate real-time RS decoding of the 19.2 kbps data rate, TMS320C5x assembly language programming was necessary. As a result, 20 msec of CDPD data (the time length of $i$ CDPD forward channel block) with $i = 8$ symbol errors could be RS decoded in less than 4 msec.

IV. Conclusions

The major topics required to develop quality CDPD AirLink equipment cover a wide variety of electrical engineering specialty areas. The implementation of a GMSK transceiver involves analog and digital communications theory. The development of network protocol implementations requires knowledge in the area of computer systems architecture, and the implementation of RS error correction encoders and decoders requires an understanding of coding theory.

We researched and implemented each of the topics described above using a combination of hardware and software. Gaussian minimum shift keying was implemented using a commercially available GMSK modem chip (MX-COM MX589). A $(63,47)$ RS decoder was written in TMS320-C5x assembly language for fast execution speed.

The prototype hardware and software are currently being tested in a Hewlett-Packard 89320 top box for the HP-8920. The complete system, shown in Fig. 2, has been tested in an actual CDPD cell-site on the Sprint Cellular system in Raleigh, North Carolina. The Grayson CDPD transceiver was able to successfully receive a Grayson-originated transmission. This confirms that the CDPD encoding and decoding hardware and software are functioning properly.

V. Acknowledgements

This research was sponsored by TSR Technologies, Inc., Grayson Electronics, and the MPRG Industrial Affiliates program.

REFERENCES