Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**	
Title	**Addendum to Recirculation Ballot #13a Editorial Comment(s)**	
Date Submitted	**2003-12-29**	
Source(s)	Brian Eidson Conexant Systems, Inc. 9860 Scranton Rd, Suite 1000 San Diego, CA 92121 USA	Voice: +1 858 713-4720 Fax: +1 858 713-3555 mailto: brian.eidson@conexant.com
Re:	IEEE 802.16-03/51r3 and IEEE P802.16-REVd/D2-2003	
Abstract	Supports an multi-item editorial comment touching various areas within clause 8.2 that the author has submitted for Recirculation Ballot #13a. This reply comment is associated with Comment #248 on the WirelessMAN-SCa PHY.	
Purpose	To provide text and editing instructions for the comment referenced by author's Commentary submission.	
Notice	This document has been prepared to assist IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.	
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.16.	
Patent Policy and Procedures	The contributor is familiar with the IEEE 802.16 Patent Policy and Procedures <http://ieee802.org/16/ipr/patents/policy.html>, including the statement "IEEE standards may include the known use of patent(s), including patent applications, provided the IEEE receives assurance from the patent holder or applicant with respect to patents essential for compliance with both mandatory and optional portions of the standard." Early disclosure to the Working Group of patent information that might be relevant to the standard is essential to reduce the possibility for delays in the development process and increase the likelihood that the draft publication will be approved for publication. Please notify the Chair <mailto:chair@wirelessman.org> as early as possible, in written or electronic form, if patented technology (or technology under patent application) might be incorporated into a draft standard being developed within the IEEE 802.16 Working Group. The Chair will disclose this notification via the IEEE 802.16 web site <http://ieee802.org/16/ipr/patents/notices>.	

Addendum to Recirculation Ballot #13a Editorial Comment(s)

Brian Eidson
Conexant Systems, Inc.

Summary:

The resolution to Comment #248 n IEEE 802.16-03/51r3 was excellent; however, the terminology it generated was not consistently or fully applied throughout clause 8.2 (on the WirelessMAN-SCa PHY). In some subclauses, alternate terminologies were retained that could lead to reader confusion. This contribution proposes a number of editorial changes, applied throughout clause 8.2, that are intended to improve its readability, clarity and internal alignment.

==========

Make the following editorial changes (indicated in blue) to clause 8.2, beginning page 337, line 1.

8.2 WirelessMAN-SCa PHY

The WirelessMAN-SCa PHY is based on single-carrier technology and designed for NLOS operation in the 2–11 GHz frequency bands (per 1.3.4). For licensed bands, channel bandwidths allowed shall be limited to the regulatory provisioned bandwidth divided by any power of 2 no less than 1.25 MHz.

Elements within this PHY include:

- TDD and FDD definitions, one of which must be supported.
- TDMA uplink.
- TDM or TDMA downlink.
- Block adaptive modulation and FEC coding for both uplink and downlink.
- Framing ~~elements~~ structures that enable improved equalization and channel estimation performance over NLOS and extended delay spread environments.
- PS-unit granularity in burst sizes.
- Concatenated FEC using Reed–Solomon and pragmatic trellis coded modulation (TCM) with optional interleaving.
- Additional BTC and CTC FEC options ~~using BTC and CTC~~.
- No-FEC option using ARQ for error control.
- Space time coding (STC) transmit diversity option.

— Robust modes for low CINR operation.

— Parameter settings and MAC/PHY messages that facilitate optional AAS implementations.

Within the discussion of the WirelessMAN-SCa PHY, five terms (payload, burst, burst set, burst frame, and MAC frame) are used in discussion of the organization of transmissions.

Payload refers to individual units of transmission content that are of interest to some entity at the receiver.

A burst contains payload data and is formed according to the rules specified by the burst profile associated with the burst. The existence of the burst is made known to the receiver through the contents of either the uplink or downlink maps. For the uplink, a burst is a complete unit of transmission that~~which~~ includes a leading preamble, encoded payload, and trailing termination sequence.

A burst set is a self-contained transmission entity consisting of a preamble, one or more concatenated bursts, and a trailing termination sequence. For the uplink, burst set is synonymous with burst.

A burst frame contains all information included in a single transmission. It consists of one or more burst sets.

A MAC frame refers to the fixed bandwidth intervals reserved for data exchange. For TDD, a MAC frame consists of one downlink and one uplink subframe, delimited by the TTG. For FDD, the MAC frame corresponds to the maximum length of the downlink subframe. FDD uplink subframes operate concurrently with downlink subframes but on a separate (frequency)~~an adjacent~~ channel. ~~The downlink and uplink subframes each hold a burst frame.~~

The downlink and uplink subframes each hold a burst frame.

8.2.1 Transmit processing

Figure 153 illustrates the steps involved in transmit processing. Source data shall first be randomized, ~~and~~ then FEC encoded and mapped to QAM symbols. ~~The~~ QAM symbols shall ~~next~~ be framed within a burst set, which typically introduces additional framing symbols. Symbols within a~~The~~ burst set~~ symbols~~ shall ~~then~~ be multiplexed into a duplex frame, which may contain multiple bursts. ~~The~~ I and Q symbol components shall be injected into pulse shaping filters, quadrature modulated up to a carrier frequency, and amplified with power control so that the proper output power is transmitted.

Except where indicated otherwise, transmit processing is the same for both the uplink and downlink.

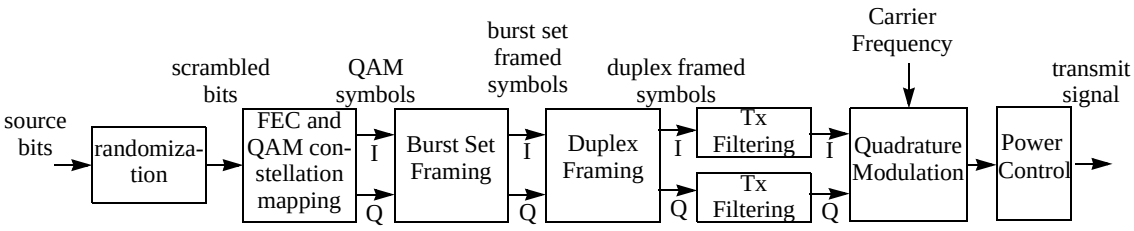

Figure 153—Transmit processing

8.2.1.1 Source Bit Randomization

Source bits, i.e., the original information bits prior to FEC encoding, shall be randomized during transmission.

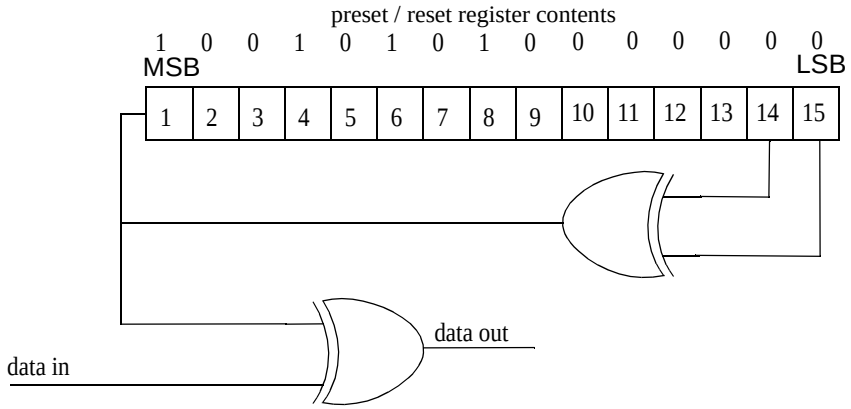

Figure 154—Randomizer for energy dispersal

As Figure 154 illustrates, source bit randomization shall be performed by modulo-2 addition (XORing) source (information) data with the output of a Linear-Feedback Shift Register (LFSR) possessing characteristic polynomial $1 + X^{14} + X^{15}$. The LFSR shall be preset at the beginning of each burst set (~~i.e.,~~ directly following the preamble) to the value 100101010000000, and shall be clocked once per processed bit. ~~This implies that t~~The LFSR is not preset between time division multiplexed allocations that may reside within a single burst set.

~~Note that o~~Only source bits are randomized. This includes source payloads, plus uncoded null (zero) bits that may be used to fill empty payload segments. ~~Only the source bits are randomized.~~ Elements that are not a part of the source data, such as framing elements and pilot symbols shall not be randomized. Null (zero) bits used to complete a QAM symbol (when an allocation does not fill an entire QAM symbol) shall not be ~~scrambled~~randomized.

8.2.1.2 FEC

FCH payloads shall be encoded in accordance with section 8.2.1.5.3. Adaptive modulation and the concatenated FEC of 8.2.1.2.1 shall be supported for all other payloads. The support of 8.2.1.2.3 as FEC as well as omitting the FEC and relying solely on ARQ for error control (see 8.2.1.2.2) is optional for payloads carried outside the FCH.

8.2.1.2.1 Concatenated FEC

The concatenated FEC is based on the serial concatenation of a Reed–Solomon outer code and a rate-compatible TCM inner code. Block interleaving between the outer and inner encoders is optional. Figure 155 illustrates the flow between blocks used by a concatenated FEC encoder.

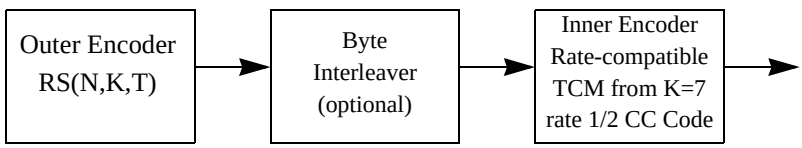

Figure 155—Concatenated FEC encoder blocks

8.2.1.2.1.1 Outer code

The outer code consists of a Reed–Solomon code.

This Reed–Solomon code shall be derived from a systematic RS (*N=255, K=239*) code using GF(2^8). The following polynomials are used for the systematic code:

Code Generator Polynomial: $g(x) = (x + \lambda^0)(x + \lambda^1)(x + \lambda^2)\ldots(x + \lambda^{2T-1}), \lambda = 02_{HEX}$

Field Generator Polynomial: $p(x) = x^8 + x^4 + x^3 + x^2 + 1$

The bit/byte conversion shall be MSB first.

This RS code may be shortened and punctured to enable variable block sizes and variable error-correction capability,

where

N	is the number of overall bytes after encoding
K	is the number of data bytes before encoding
R=N–K	is the number of parity bytes.

When a block is shortened to *K'* data bytes, the first *239-K'* data bytes of the block to be encoded shall be set to zero, but shall not be transmitted. When a codeword is punctured to *R'* parity bytes, only the first *R'* of the total *R*=16 parity bytes shall be transmitted.

Support of shortening *K* of the base code to values smaller than 239 bytes while maintaining *R*=16 is mandatory, and is governed by the burst profile specification for *K* (see 11.1.1.2 or 11.1.2.2). The capability to also puncture, such that $R \leq 16$, is mandatory, and is governed by the burst profile specification for *R*. However, payloads that cannot be modified by burst profile changes, such as the contents of the FCH, shall not be punctured.

When a source allocation does not divide into an integer number of *K* byte Reed–Solomon code words, the last (fractional) RS code word shall be shortened to a smaller value 1<=*K*'<*K* that accommodates the remainder bytes. All code words, including the shortened last codeword, shall use the *R* specified by the burst profile (see Table 269 and Table 274) for the RS code words within that allocation.

8.2.1.2.1.2 Block Interleaver

Support of interleaving between the inner and outer code with a depth of $N_R = 10$ is mandatory. Interleaving shall not be defined in the FCH burst profile. When interleaving is used, its usage and parameters shall be specified within a burst profile.

The interleaver changes the order of bytes from the Reed–Solomon (RS) encoder output. A de-interleaver in the receiver restores the order of the bytes prior to RS decoding. The interleaver is a block interleaver, where a table is 'written', i.e., filled, a byte at a time row-wise (one row per RS code word) and 'read' a byte at a time column-wise. The number of rows, N_R, used by the interleaver is a burst parameter. So that bursts are not generated that exceed an intended receiver's capabilities, the largest N_R supported by a terminal is communicated during SS basic capability negotiation.

Operating parameters for the interleaver are summarized in Table 151.

Table 151—Operating parameters for block interleaver

Parameter	Description
C	Interleaver Width (number of columns), in bytes. Equivalent to the nominal Reed–Solomon codeword length, N.
N_R	Maximum Interleaver Depth (number of rows), in bytes. Equals the maximum number of RS codewords that the block interleaver may store at any given time.
B	Nominal Interleaver Block Size, in bytes. $B = C\,N_R$.
P	RS-encoded Size of Packet, in bytes, to be interleaved.

When $P \le B$ and/or a RS codeword is shortened (so that not all of the columns within its row are filled), the interleaver shall be read column by column (taking a byte from each column), skipping empty elements within the table.

When $P > B$, data shall be parcelled into subblocks, and interleaving performed within each of the subblocks. The depth of these subblocks shall be chosen such that all subblocks have approximately the same depth (number of rows) using the following calculations:

Total RS codewords in packet: $T = \left\lceil \frac{P}{C} \right\rceil$

Number of subblocks: $S = \left\lceil \frac{P}{B} \right\rceil$

Interleaver depth of longest subblocks: $C_{max} = \left\lceil \frac{T}{S} \right\rceil$

Number of blocks with depth C_{max}: $Q_{C_{max}} = T - S(C_{max} - 1)$

Number of blocks with depth $C_{min} = C_{max} - 1$: $Q_{C_{min}} = S - Q_{C_{max}}$.

The first $Q_{C_{max}}$ subblocks within a packet shall use a (dynamic) interleaver depth C_{max}, and the remainder of the subblocks shall use an interleaver depth $C_{min} = C_{max} - 1$.

8.2.1.2.1.3 Inner code

The inner code is a rate-compatible pragmatic TCM code [B46], [B47] derived from a rate 1/2 constraint length K=7, binary convolutional code.

The encoder for the rate 1/2 binary code shall use the following polynomials to generate its two code bit outputs, denoted X and Y:

$$\begin{aligned} G_1 &= 171_{OCT} \qquad For\ \ X \\ G_2 &= 133_{OCT} \qquad For\ Y \end{aligned} \tag{10}$$

A binary encoder that implements this rate 1/2 code is depicted in Figure 156.

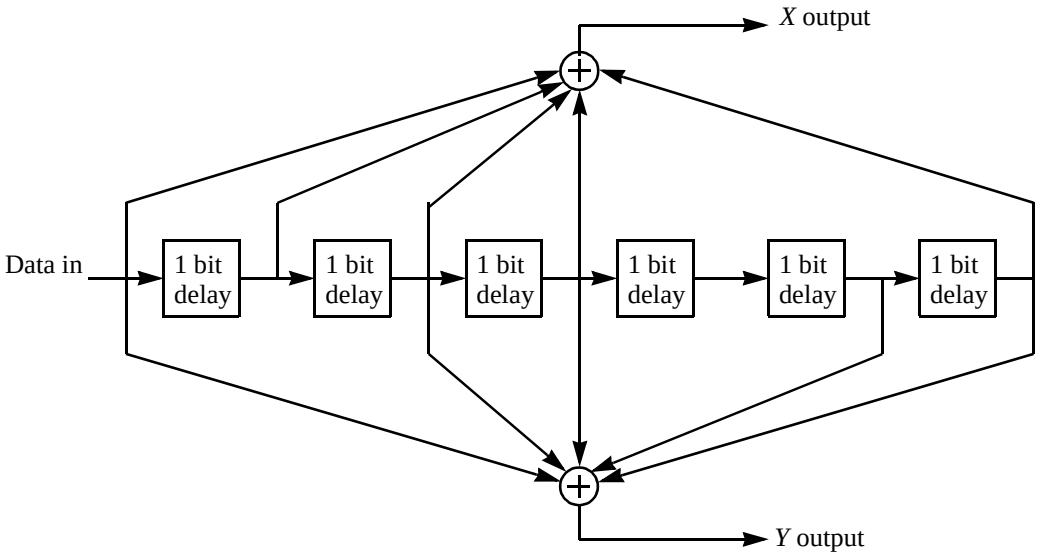

Figure 156—Binary rate 1/2 convolutional encoder

To generate binary code rates of 2/3, 3/4, 5/6, and 7/8, the rate 1/2 encoder outputs shall be punctured. The puncturing patterns and serialization order for the *X* and *Y* outputs are defined in Table 152. In the puncture patterns, a '1' denotes a transmitted output bit and a '0' denotes a nontransmitted (punctured) bit.

Table 152—Puncture patterns and serialization for convolution code

	Code Rates				
Rate	1/2	2/3	3/4	5/6	7/8
X Output Puncture pattern	1	10	101	10101	1000101
Y Output Puncture pattern	1	11	110	11010	1111010
Punctured *XY* serialization	X_1Y_1	$X_1Y_1Y_2$	$X_1Y_1Y_2X_3$	$X_1Y_1Y_2X_3Y_4X_5$	$X_1Y_1Y_2Y_3Y_4X_5Y_6X_7$

A~~T~~he pragmatic TCM code is constructed from both nonsystematic coded bits (that are taken from the outputs of the punctured rate 1/2 binary convolutional encoder) and systematic uncoded bits (that are taken directly from the encoder input). The resulting coded bits are then mapped to symbol constellations. Supported modulations and code rates for uplink and downlink transmissions are listed in Table 153. The choice of a particular code rate and modulation is made via burst profile parameters.

Since the RS outer code generates byte-denominated records but the inner code generates symbol-denominated outputs, some RS record sizes could require a fractional QAM symbol at the end of the data record. When this occurs,

sufficient (nonrandomized) zero-valued (null) bits shall be appended to the end of the inner encoder's input record to complete the final symbol. A receiver shall discard these null bits after inner decoding..

Table 153—Supported modulations and inner (TCM) code rates

Modulation	Support (M=Mandatory, O=Optional)		Inner code rates	Bits/symbol
	UL	DL		
Spread BPSK	M	M	(pre-spread) 1/2, 3/4	(post-spread) 1/(2*Fs), 3/(4*Fs)
BPSK	M	M	1/2, 3/4	1/2, 3/4
QPSK	M	M	1/2, 2/3, 3/4, 5/6, 7/8	1, 4/3, 3/2, 5/3, 7/4
16-QAM	M	M	1/2, 3/4	2, 3
64-QAM	M	M	2/3, 5/6	4, 5
256-QAM	O	O	3/4, 7/8	6, 7

Inner code blocks are to be zero-state terminated in transitions between adaptive modulation (and FEC) types, at the ends of bursts, or as instructed by the MAC and frame control.

When using zero state termination, the baseline rate 1/2 convolutional encoder shall be initialized with its registers in the all-zeros state. Inner encoding shall begin from this state, by accepting bit inputs. To terminate the inner code (and return the encoder to the all-zeros state) at the end of a code block, at least 6 zero inputs shall be fed into the baseline rate 1/2 binary convolutional encoder to ensure its register memory is flushed, i.e., its state memory is driven to zero. Once the first flushing zero bit is introduced into the convolutional encoder memory, all input bits, including the systematic input bits that are parallel to the binary convolutional encoder inputs, shall have zero value.

Table 154 specifies the exact number of systematic and nonsystematic bits that shall be used to flush a pragmatic TCM encoder for a given modulation and code rate.. It also tabulates the number of symbols consumed in the code termination process. Spread BPSK with a spreading factor of Fs consumes Fs-times more symbols than BPSK

Table 154—Flushing bit requirements for inner code termination

Modulation	Inner code rate	Number of flushing bits			Number of consumed symbols
		Nonsystematic	Systematic	Total	
spread BPSK	1/2	(pre-spread) 6	(pre-spread) 0	(pre-spread) 6	(post-spread) 12*Fs
	3/4	(pre-spread) 6	(pre-spread) 0	(pre-spread) 6	(post-spread) 8*Fs
BPSK	1/2	6	0	6	12
	3/4	6	0	6	8

Table 154—Flushing bit requirements for inner code termination (Continued)

Modulation	Inner code rate	Number of flushing bits			Number of consumed symbols
		Nonsystematic	Systematic	Total	
QPSK	1/2	6	0	6	6
	2/3	7	0	7	5
	3/4	6	0	6	4
	5/6	6	0	6	4
	7/8	7	0	7	4
16-QAM	1/2	6	0	6	3
	3/4	6	12	18	6
64-QAM	2/3	6	6	12	3
	5/6	6	4	10	2
256-QAM	3/4	6	12	18	3
	7/8	6	8	14	2

— **Encoding for spread BPSK, ~~A~~all ~~R~~rates**

See 8.2.1.3.2

— **Encoding for BPSK and QPSK ~~M~~modulations, ~~A~~all Rrates**

For BPSK, the binary outputs of the punctured binary encoder shall be directly sent to the BPSK symbol mapper, using the multiplexed output sequence shown in the 'XY'-headed row of Table 152. For QPSK, the multiplexed output sequence in Table 152 is alternately assigned to the I and Q coordinate QPSK mapper, with the I coordinate receiving the first assignment.

Clause 8.2.1.3.1 describes the constellation mapping procedure and Figure 165 and Figure 166 depict bits-to-symbol-constellation maps that shall be used for BPSK and QPSK, respectively.

— **Encoding for ~~R~~rate 1/2 16-QAM**

Figure 157 illustrates the rate 1/2 pragmatic TCM encoder for 16-QAM. The baseline rate 1/2 binary convolutional encoder first generates a 2-bit constellation index, b_3b_2, associated with the I symbol coordinate. Provided the next encoder input, it generates a two-bit constellation index, b_1b_0, for the Q symbol coordinate. The I index generation shall precede the Q index generation. Note that this encoder should be interpreted as a rate 2/4 encoder, because it generates one 4-bit code symbol per two input bits. For this reason, input records of lengths divisible by two shall be fed to this encoder.

Figure 166 depicts the bits-to-constellation map that shall be applied to the rate 1/2 16-QAM encoder output. This is a Gray code map.

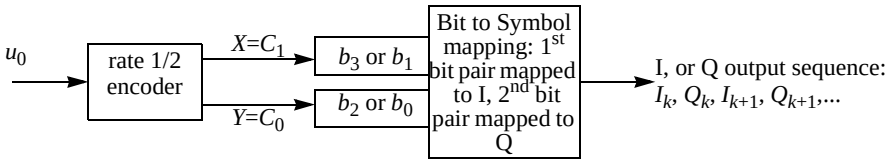

Figure 157—Pragmatic TCM encoder for rate 1/2 16-QAM

— **Encoding for ~~R~~rate 3/4 16-QAM**

Figure 158 illustrates the rate 3/4 pragmatic TCM encoder for 16-QAM. This encoder uses the baseline rate 1/2 binary convolutional encoder, along with two systematic bits that are passed directly from the encoder input to the encoder output. With this structure, the encoder is capable of simultaneously generating 4 output bits per three input bits. The sequence of arrival for the $u_2u_1u_0$ input into the encoder is u_2 arrives first, u_1 second, u_0 last. During the encoding process, the encoder generates a two-bit constellation index, b_3b_2, for the I symbol coordinate, and simultaneously generates another two-bit constellation index, designated b_1b_0, for the Q symbol coordinate. Note that whole symbols shall be transmitted, so input records of lengths divisible by three shall be fed to this encoder.

Figure 169 depicts the bits-to-symbol-constellation map that shall be applied to the rate 3/4 16-QAM encoder output. This is pragmatic TCM map.

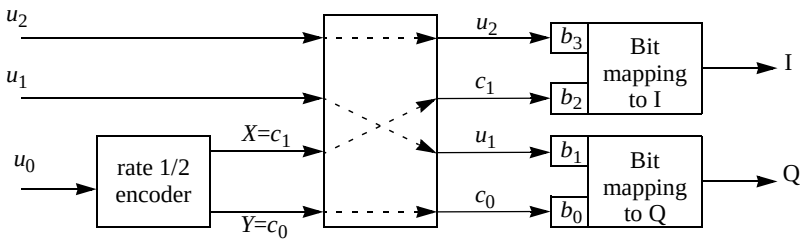

Figure 158—Pragmatic TCM encoder for rate 3/4 16-QAM

— **Encoding for ~~R~~rate 2/3 64-QAM**

Figure 159 illustrates the rate 2/3 pragmatic TCM encoder for 64-QAM. This encoder uses the baseline rate 1/2 binary convolutional encoder, along with one systematic bit that is passed directly from the encoder input to the encoder output. The sequence of arrival for the u_1u_0 input into the encoder is u_1 arrives first, u_0 last. The encoder (as a whole) then generates a 3-bit constellation index, $b_5b_4b_3$, which is associated with the I symbol coordinate. Provided another 2-bit encoder input, the encoder generates another 3-bit constellation index, $b_2b_1b_0$, which is associated with the Q symbol coordinate. The I index generation should precede the Q index generation. Note that this encoder should be interpreted as a rate 4/6 encoder, because it generates one 6-bit code symbol per four input bits. For this reason, input records of lengths divisible by four shall be fed to this encoder.

Figure 169 depicts the bits-to-symbol-constellation map that shall be applied to the rate 2/3 64-QAM encoder output. This is a pragmatic TCM map.

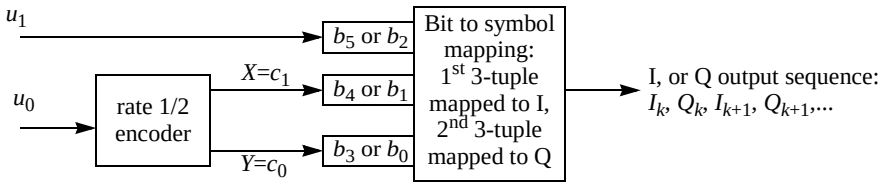

Figure 159—Pragmatic TCM encoder for rate 2/3 64-QAM

— **Encoding for ~~R~~rate 5/6 64-QAM**

Figure 160 illustrates the rate 5/6 pragmatic TCM encoder for 64-QAM. This encoder uses a rate 3/4 punctured version of the rate baseline rate 1/2 binary convolutional encoder, along with two systematic bits that are passed directly from the encoder input to the encoder output. The rate 3/4 punctured code is generated from the baseline rate 1/2 code using the rate 3/4 puncture mask definition in Table 152. Puncture samples are sequenced c_3 first, c_2 second, c_1 third, and c_0 last. The sequence of arrival for the $u_4u_3u_2u_1u_0$ input into the encoder is u_4 arrives first, u_3 arrives second, u_2 arrives third, u_1 arrives next to last, and u_0 arrives last. During the encoding process, the pragmatic encoder generates a 3-bit constellation index, $b_5b_4b_3$, for the I symbol coordinate, and simultaneously generates another 3-bit constellation index, $b_2b_1b_0$, for the Q symbol coordinate. Note that whole symbols shall be transmitted, so input records of lengths divisible by five shall be fed to this encoder.

Figure 169 depicts the bits-to-symbol-constellation map that shall be applied to the rate 5/6 64-QAM encoder output. This is a pragmatic TCM map.

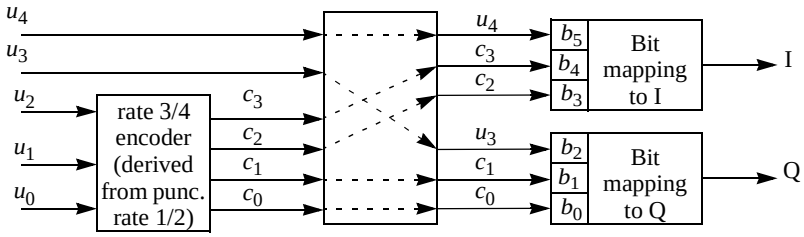

Figure 160—Pragmatic TCM encoder for rate 5/6 64-QAM

— **Encoding for ~~R~~rate 3/4 256-QAM**

Figure 161 illustrates the rate 3/4 pragmatic TCM encoder for 256-QAM. This encoder uses the baseline rate 1/2 binary convolutional encoder, along with two systematic bits that are passed directly from the encoder input to the encoder output. The sequence of arrival for the $u_2u_1u_0$ input into the encoder is u_2 arrives first, u_1 next, u_0 last. Note that the encoder (as a whole) first generates a 4-bit constellation index, $b_7b_6b_5b_4$, which is associated with the I symbol coordinate. Provided another 4-bit encoder input, it generates a 4-bit constellation index, $b_3b_2b_1b_0$, which is associated with the Q symbol coordinate. The I index generation should precede the Q index generation. Note that this encoder should be interpreted as a rate 6/8 encoder, because it generates one 8-bit code symbol per six input bits. For this reason, input records of lengths divisible by six shall be fed to this encoder.

Figure 161 depicts the bits-to-symbol-constellation map that shall be applied to the rate 3/4 256-QAM encoder output. This is a pragmatic TCM map.

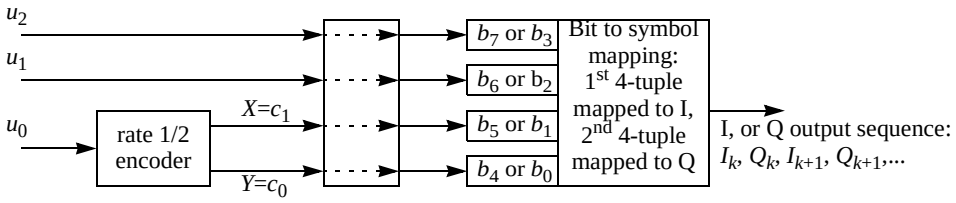

Figure 161—Optional pragmatic TCM encoder for rate 3/4 256-QAM

— **Encoding for ~~R~~rate 7/8 256-QAM**

Figure 162 illustrates the rate 7/8 pragmatic TCM encoder for 256-QAM. This encoder uses a rate 3/4 punctured version of the rate baseline rate 1/2 binary convolutional encoder, along with two systematic bits that

are passed directly from the encoder input to the encoder output. The rate 3/4 punctured code is generated from the baseline rate 1/2 code using the rate 3/4 puncture mask definition in Table 152. Puncture samples are sequenced c_3 first, c_2 second, c_1 third, and c_0 last. The sequence of arrival for the $u_6u_5u_4u_3u_2u_1u_0$ input into the encoder (as a whole) is u_6 arrives first, u_5 arrives second, u_4 arrives third, u_3 arrives fourth, u_2 arrives fifth, u_1 arrives next to last, and u_0 arrives last. During the encoding process, the encoder generates a 4-bit constellation index, $b_7b_6b_5b_4$, for the I symbol coordinate, and simultaneously generates another 4-bit constellation index, $b_3b_2b_1b_0$, for the Q symbol coordinate. Note that whole 256-QAM symbols should be transmitted, so input records of lengths divisible by seven shall be fed to this encoder.

Figure 162 depicts the bits-to-symbol-constellation map that shall be applied to the rate 7/8 256-QAM encoder output. This is a pragmatic TCM map.

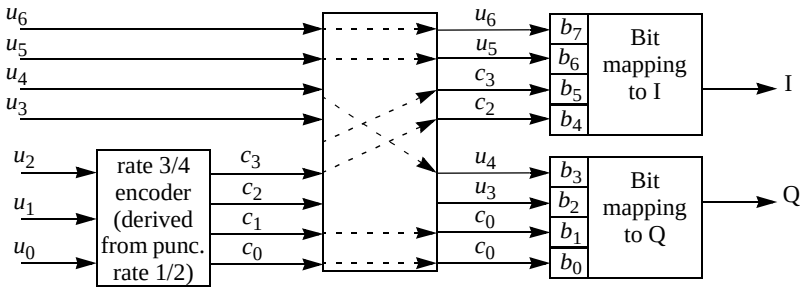

Figure 162—Optional pragmatic TCM encoder for rate 7/8 256-QAM

8.2.1.2.2 No FEC

In the No FEC option, scrambled source data shall be mapped directly to a QAM symbol constellation, using the appropriate Gray coding map. These maps are found in Figure 165 (for BPSK), Figure 166 (for QPSK, 16-QAM), Figure 167 (for 64-QAM), and Figure 168 (for 256-QAM). No-FEC operation is mandatory for QPSK but optional for other modulations ~~methods~~.

In the event that the source record size in bytes does not divide into an integral number of QAM symbols, sufficient unscrambled zero-valued (null) bits shall be appended to the end of the data record to complete the last symbol. These null bits shall be discarded at the receiver.

8.2.1.2.3 BTCs

Support of the BTC FEC is optional.

A BTC is formed from block row codes, each with rate parameters (n_x, k_x) and block column codes, each with rate parameters (n_y, k_y). The BTC is encoded by writing data bits row by row into a two-dimensional matrix as illustrated in Figure 163. The k_x information bits in each row are encoded into n_x bits by using a constituent block (n_x, k_x) row code. The k_y information bits in each column are encoded into n_y bits by using a constituent block (n_y, k_y) column code, where the checkbits of the rows are also encoded. The resulting BTC shall have block length $n = n_x \times n_y$ bits and information length $k = k_x \times k_y$.

The constituent row code and constituent column code used to form the rows and columns of a BTC shall be specified by BTC-specific burst profile parameters. The constituent codes available for specification are listed in Table 155. All codes in Table 155 shall be formed by appending a check bit or check bits to the end of the information bits. A parity check code shall use one check bit, derived by XORing the information bits.

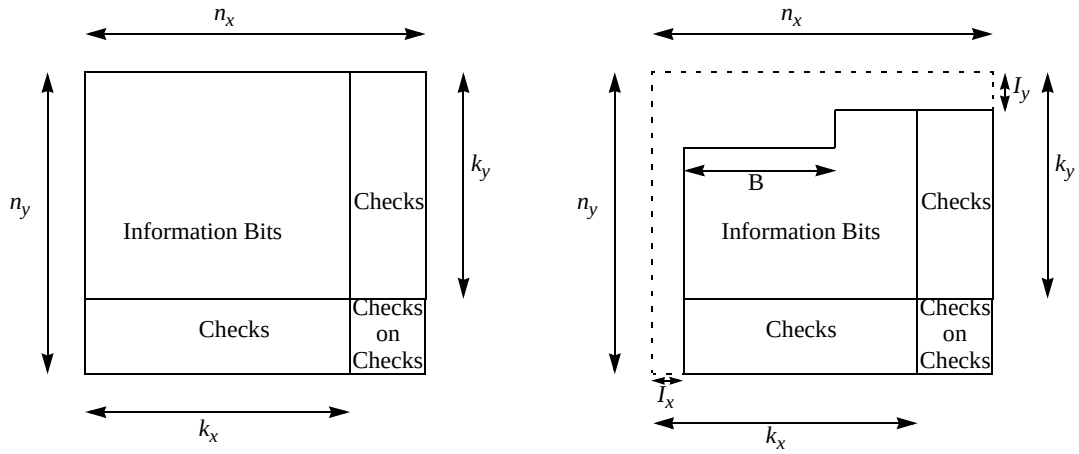

Figure 163—BTC and shortened BTC structures

An extended Hamming component code shall use n-k check bits. The first *n–k*–1 check bits are the check bits of a (*n*–1, *k*) Hamming code derived from one of the generator polynomials listed in Table 156, while the last check bit is a parity check bit, derived by XORing the *n*–1 information and parity bits of the (*n*–1, *k*) Hamming code.

Table 155—BTC component codes

Component codes (*n, k*)	Code type
(64,57), (32,26), (16,11), (8,4)	Extended Hamming Code
(64,63), (32,31), (16,15), (8,7)	Parity Check Code

Table 156—(n-1,k) Hamming code generator polynomials

n–1	k	Generator polynomial
7	4	X^3+X^1+1
15	11	X^4+X^1+1
31	26	X^5+X^2+1
63	57	X^6+X+1

To match an arbitrary required packet size, BTCs may be shortened by removing symbols from the BTC array. Rows, columns or parts thereof can be removed until the appropriate size is reached. Two steps, illustrated in Figure 163, are involved in the shortening of product codes:

Step 1) Remove I_x rows and I_y columns from the two-dimensional code. This is equivalent to shortening the constituent codes that make up the product code.

Step 2) Remove *B* individual bits from the first row of the two-dimensional code starting with the LSB.

The resulted block length of the code is $(n_x–I_x)(n_y–I_y)–B$. The corresponding information length is given as $(k_x–I_x)(k_y–I_y)–B$. Consequently, the code rate is given by:

$$R = \frac{(k_x - I_x)(k_y - I_y) - B}{(n_x - I_x)(n_y - I_y) - B} \tag{11}$$

Data bit ordering for the composite BTC matrix is defined such that the leftmost bit in the first row is the LSB. The next bit in the first row is the next-to-LSB. The last data bit in the last data row is MSB. An encoded BTC block shall be read out of the encoded matrix (for transmission) as a serial bit stream, starting with the LSB and ending with the MSB. This bit stream shall be sent to a symbol mapper which uses a Gray map depicted in Figure 165 for BPSK, Figure 166 for QPSK, 16-QAM, and the Gray maps depicted in Figure 167 and Figure 168 for 64-QAM, and 256-QAM. If not enough encoded bits are available to fill the last symbol of an allocation, sufficient zero-valued bits (unscrambled) shall be appended to the end of the serial stream to complete the symbol.

Two independent variables, C_{bank} and K, collectively specify the BTC encoding and shortening process. C_{bank} is a burst profile parameter specified by the MAC. Its values range from 1 to 3, where each integer refers to a unique set of constituent codes, chosen to set the code rate of the BTC. K is the desired information block length, in bits, to be transmitted within an allocation. Table 157 specifies three code selection banks, each containing 4 different base BTCs, covering a range of information and encoded data block lengths. Each base BTC is composed from the specified row and column codes. The code selection bank that most closely matches the desired performance should be chosen as the active code bank.

Table 157—BTC code banks

	Component row code (n_x,k_x) bits	**Component column code (n_y,k_y) bits**	**Base BTC block ($n_y n_x$,$k_y k_x$) bits**
C_{bank}=1 $R \cong 0.94$	(64,63)	(64,63)	(4096,3969)
	(32,31)	(32,31)	(1024,961)
	(16,15)	(16,15)	(256,225)
	(8,7)	(8,7)	(64,49)
C_{bank}=2 $R \cong 0.80$	(64,63)	(64,57)	(4096,3591)
	(32,31)	(32,26)	(1024,806)
	(16,15)	(16,11)	(256,165)
	(8,7)	(8,4)	(64,28)
C_{bank}=3 $R \cong 0.69$	(64,57)	(64,57)	(4096,3249)
	(32,26)	(32,26)	(1024,676)
	(16,11)	(16,11)	(256,121)
	(8,4)	(8,4)	(64,16)

From the selected C_{bank}, a corresponding row and column constituent code shall be chosen that best matches the total number of information bits, K, to be encoded. Then the result shall be shortened to the exact information block size. The procedure for doing so shall be as follows:

Step 1: Determine the row and column component codes.

Select the base BTC with the smallest $k_x \times k_y$, such that $K \le k_x \times k_y$. Should K exceed the largest information block length available in the code selection bank, the K information bits shall be split across N_{blocks} = ceil(K/(8 * maxInfoBlock)), where maxInfoBlock is the number of information bytes in the largest BTC in the code selection bank. The first $N_{blocks} - 1$ blocks shall encode ceil($K/8/N_{blocks}$) bytes. The final block shall encode the remaining ($K/8$)–($N_{blocks} - 1$)*ceil($K/8/N_{blocks}$) bytes. Each of the N_{blocks} blocks is encoded according to Step 2, substituting for K the number of information bits assigned to that block. Shortening of the BTC may be required for all N_{blocks} blocks. The code selection bank shall remain unchanged for the duration of the N_{blocks} blocks.

Step 2: Determine the row and column component codes for remnant bits

Select the base BTC with the smallest $k_x \times k_y$, such that $K \le k_x \times k_y$.

Should K be equal to $k_x \times k_y$, then K fits exactly into the base BTC and the information bits are encoded without shortening. If this is not the case, then the $(n_y \times n_x, k_y \times k_x)$ BTC shall be shortened according to Step 3 after which the K bits are encoded.

Step 3: Determine shortening parameters

Select i such that:

$$\arg\min_i \left[\left(\left(k_x - \left\lfloor \frac{i+1}{2} \right\rfloor \right) \cdot \left(k_y - \left\lfloor \frac{i}{2} \right\rfloor \right) - K \right) > 0 \right] \qquad 0 \le i < 2k_y - 1 . \tag{12}$$

The obtained i specifies the shortening parameters: $I_x = \lfloor (i+1)/2 \rfloor$, $I_y = \lfloor i/2 \rfloor$ and $B = (k_x - I_x) \times (k_y - I_y) - K$.

8.2.1.2.4 Convolutional Turbo Codes

Support of the Convolutional Turbo Code FEC is optional.

8.2.1.2.4.1 CTC encoder

The Convolutional Turbo Code encoder, including its constituent encoder, is depicted in Figure 164. It uses a double binary Circular Recursive Systematic Convolutional code. The bits of the data to be encoded are alternately fed to inputs A and B, starting with the MSB of the first byte being fed to A. The encoder is fed by blocks of k bits or N couples ($k = 2*N$ bits). For all ~~the~~ frame sizes k is a multiple of 8 and N is a multiple of 4. Furthermore, N shall be limited to: $8 \le N/4 \le 256$. Zero padding should be used for block sizes less than 32 bytes. For allocations longer than 256 bytes (i.e., 512 nibbles), the allocation (in units of nibbles), A_n determines the number of interleaver code blocks, n_B, that shall be transmitted. The first n_F of these blocks shall each use an interleaver of length N_F nibbles, whereas the remainder shall each use an interleaver of length $N_F + 1$ nibbles.

$$\begin{aligned} n_B &= \left\lceil \frac{A_n}{512} \right\rceil \\ N_F &= \left\lfloor \frac{A_n}{n_B} \right\rfloor \\ n_F &= n_B(N_F + 1) - A_n \end{aligned} \tag{13}$$

The polynomials defining the encoder connections are described in octal and binary symbol notations as follows:

— for the feedback branch: 0xB, equivalently $1+D+D^3$ (in binary symbolic notation)

— for the Y parity bit: 0xD, equivalently $1+D^2+D^3$

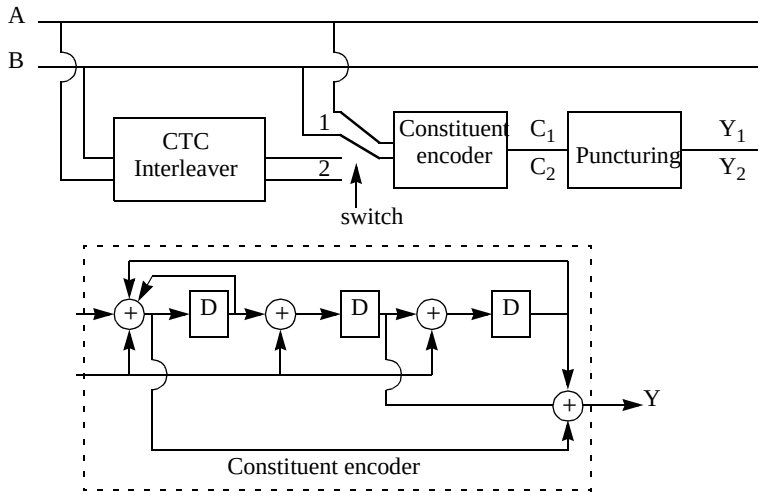

Figure 164—CTC encoder

First, the encoder (after initialization by the circulation state Sc_1, see subclause 8.2.1.2.4.2) is fed the sequence in the natural order (position 1) with the incremental address $i = 0,...,N–1$. This first encoding is called C_1 encoding. Then the encoder (after initialization by the circulation state Sc_2, see in subclause 8.2.1.2.4.2) is fed by the interleaved sequence (switch in position 2) with incremental address $j = 0,…,N–1$. This second encoding is called C_2 encoding.

8.2.1.2.4.2 CTC interleaver

The interleaver requires the parameters P_0, which shall be the nearest prime number greater than $\sqrt{N/2}$, with $P_0 \geq 3$, which is dependent on the block size N. The two-step interleaver shall be performed by:

Step 1: Switch alternate couples

or $j = 1…N$

if (j_{mod_2} ==0) let (B,A) = (A,B) (i.e. switch the couple)

Step 2: $P_i(j)$

The function $P_i(j)$ provides the interleaved address i of the consider couple j.

or $j = 1…N$

switch j_{mod_4} :
case 0 or 1: $i = (P_0 \cdot j + 1)_{\mathrm{mod}_N}$
case 2 or 3: $i = (P_0 \cdot j + 1 + N/4)_{\mathrm{mod}_N}$

8.2.1.2.4.3 Determination of CTC circulation states

The state of the encoder is denoted S (0 <= S == 7) with S the value read binary (left to right) out of the constituent encoder memory (see Figure 164). The circulation states Sc_1 and Sc_2 are determined by the following operations:

Step 1) Initialize the encoder with state 0. Encode the sequence in the natural order for the determination of Sc_1 or in the interleaved order for determination of Sc_2. In both cases the final state of the encoder is $S0_{N–1}$;

Step 2) According to the length N of the sequence, use Table 158 to find Sc_1 or Sc_2.

Table 158—Circulation state lookup table (*Sc*)

N_{mod_7}	$S0_{N-1}$							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0	6	4	2	7	1	3	5
2	0	3	7	4	5	6	2	1
3	0	5	3	6	2	7	1	4
4	0	4	1	5	6	2	7	3
5	0	2	5	7	1	3	4	6
6	0	7	6	1	3	4	5	2

8.2.1.2.4.4 CTC puncturing

The various code-rates are achieved through selectively deleting the parity bits (puncturing). The puncturing patterns are identical for both codes C_1 and C_2.

Table 159—Circulation state lookup table (*Sc*)

Rate $R_n/(R_n+1)$	**Y**															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1/2	1	1														
2/3	1	0	1	0												
3/4	1	0	0	1	0	0										
5/6	1	0	0	0	0	1	0	0	0	0						
7/8	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

8.2.1.2.4.5 CTC modulation mapping

The encoded bit is fed into the mapper for BPSK and QPSK in the following order:

$A_0, B_0 .. A_{N-1}, B_{N-1}, Y_{10}, Y_{1,1} .. Y_{1,M}, Y_{20}, Y_{2,1} .. Y_{2,M},$

where M is the number of parity bits and the I channel is fed first. The order in which the encoded bit are fed into the mapper for 16-QAM, 64-QAM and 265-QAM is:

$A_0, B_0 .. A_{Rn} B_{Rn}, Y_{1,0}, A_{Rn+1}, B_{Rn+2} .. A_{2Rn}, B_{2Rn}, Y_{2,0} ..$

Let S be half the number of bits in the modulation symbol. Then if (R_n+1) equals S, the parity bits are mapped to the two least protected locations in the constellation (16-QAM = $\{b_0, b_2\}$, 64-QAM = $\{b_0, b_3\}$, 256-QAM = $\{b_0, b_4\}$). If (R_n+1) equals $2S$, the parity bits are mapped to the least protected bit in the modulation scheme in the Q channel, using the mapping specified in the mandatory mode (b_0 for 16-QAM, 64-QAM and 256-QAM).

8.2.1.3 Modulations and constellation mapping

Table 160 lists supported modulations.

Table 160—Modulation support

Modulation	Support (M=Mandatory, O=Optional)	
	UL	DL
Spread BPSK	M	M
BPSK	M	M
QPSK	M	M
16-QAM	M	M
64-QAM	M	M
256-QAM	O	O

With the exception of spread-BPSK, FEC-encoded bits are mapped directly to a modulation constellation using one of the constellation maps in 8.2.1.3.1. Since multiple mappings are defined for several of the modulations, the appropriate FEC and code rate description shall be consulted to determine the specific mapping to be used. 8.2.1.4 specifies the modulation procedure to be used for spread-BPSK modulation.