Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	Optional FEC Enhancement for WirelessMAN-SCa
Date Submitted	**2002-07-02**
Source(s)	Sean Sonander Akelia Wireless 2 Venture Road Southampton, UK Voice: +44(0)2380 763716 Fax: +44(0)2380 763714 sean@akelia.com Brian Eidson Conexant Systems, Inc. Anader Benyamin- Seeyar Advantech AMT Company
Re:	
Abstract	This document describes a revision to the Section 8.3.2.1.2.3, Document P802.16a/D4-2002
Purpose	The author intends to use this document in resolving an optional FEC comment
Notice	
Release	
Patent Policy and Procedures	

Optional FEC Enhancement for WirelessMAN-SCa

Sean Sonander
Akelia Wireless

Brian Eidson
Conexant Systems, Inc.

Anader Benyamin- Seeyar
Advantech AMT Company

1 Block Turbo Codes

Support of the Block Turbo Code (BTC) FEC is optional.

1.1 Encoding

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 1(a). k_x information bits in each row are encoded into n_x bits by using a constituent block (n_x, k_x) row code. k_y bits in each column are then encoded into n_y bits by using a constituent block (n_y, k_y) column code, where the check bits of the rows are also encoded. The resulting BTC will have block length $n_x \times n_y$ bits and information length $k_x \times k_y$ bits.

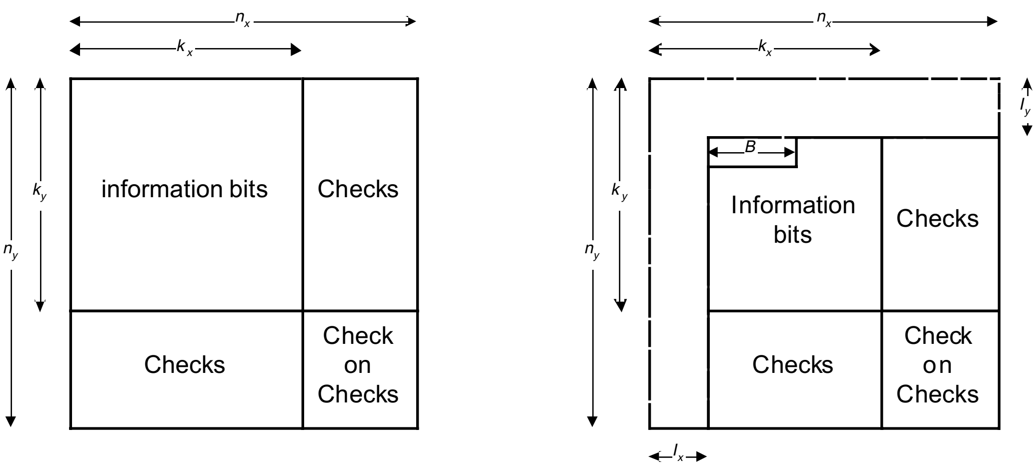

Figure 1 – BTC (a) and Shortened BTC Structure (b)

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 1.

All codes in Table 1 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 exclusive-ORing the information bits.

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 2, while the last check bit is a parity check bit, derived by exclusive-ORing the *n*-1 information and parity bits of the (*n*-1, *k*) Hamming code.

Table 1 - List of BTC Component Codes

Constituent Codes (n, k)	**Code Type**
(64, 63), (32, 31), (16, 15), (8,7)	Parity Check Codes
(64, 57), (32, 26), (16, 11), (8, 4)	Extended Hamming Codes

Table 2 – (n, k) Hamming Code Generator Polynomials

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

Shortening shall be used to reduce the size of a BTC FEC block, so that smaller source allocations may be transmitted over the channel. The parameters I_x, I_y and B specify the shortening of a BTC; they are applied as follows:

Remove I_x columns from the encoded matrix, starting with the leftmost column.

Remove I_y rows from the encoded matrix, starting with the top row.

Remove B individual data bits from the top row starting with the lsb.

Figure 1(b) illustrates an example of a shortened BTC. The resulting shortened BTC has block length $(n_x-I_x)(n_y-I_y)-B$ bits, source information length $(k_x-I_x)(k_y-I_y)-B$ bits and code rate

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

Equation 1

The parameters I_x, I_y and B are functions of row and column constituent codes and the length of the desired (shortened) BTC allocation. Clause 1.2 provides details on this functional relationship.

If a source allocation is longer than a baseline BTC FEC block, as many full BTC blocks as possible shall be transmitted, followed by a last block, which shall be shortened if it does not fill a full BTC block.

Data bit ordering for the composite BTC matrix is the left most bit in the first row is the lsb. the next bit in the first row is the next-to-lsb, and 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 sent to a symbol mapper which uses a Gray map depicted in Figure 171 for BPSK and QPSK, and the pragmatic maps depicted in Figure 172 and Figure 173 for 16-QAM, 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.

1.2 Constituent Code Selection and Shortening

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 ranges from 1 to 5, 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 3 - Table 5 specify 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 of from the specified row and column

codes. The code selection bank that most closely matches the desired code rate and performance should be chosen as the active code bank.

Table 3 - BTC Code Selection Bank C_{bank}=1, Rate approx. 0.94

Component Row Code (n_x, k_x)	**Component Column Code (n_y, k_y)**	**Base BTC (n, k)**
(64, 63)	(64, 63)	(4096, 3969)
(32, 31)	(32, 31)	(1024, 961)
(16, 15)	(16, 15)	(256, 225)
(8, 7)	(8, 7)	(64, 49)

Table 4 - BTC Code Selection Bank C_{bank}=2, Rate approx 0.69

Component Row Code (n_x, k_x)	**Component Column Code (n_y, k_y)**	**Base BTC (n, k)**
(64, 57)	(64, 57)	(4096, 3249)
(32, 26)	(32, 26)	(1024, 676)
(16, 11)	(16, 11)	(256, 121)
(8, 4)	(8, 4)	(64, 16)

Table 5 - BTC Code Selection Bank C_{bank}=5, Rate Approx 0.8

Component Row Code (n_x, k_x)	**Component Column Code (n_y, k_y)**	**Base BTC (n, k)**
(64, 57)	(64, 63)	(4096, 3591)
(32, 26)	(32, 31)	(1024, 806)
(16, 11)	(16, 15)	(256, 165)
(8, 4)	(8, 7)	(64, 28)

For the chosen code selection bank 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 must be shortened to the exact information block size. The procedure for doing so is as follows:

Step 1. Determine the row and column component codes.

In any code selection bank C_{bank}, 4 base BTCs are given. The base BTC with the smallest information block length k, providing that $K \leq k$ shall be chosen. Should K exceed the largest information block length available in the code selection bank, the base BTC with the largest information block is selected, with parameters (n, k). From the total K information bits, blocks of k bits are encoded according to Clause 1.1, until the number of remaining information bits is less than k. The parameter K is then set to equal the remaining number of information bits. With code selection bank unchanged, the process repeats from Step 1.

Should K be equal to $k_x \times k_y$, then K fits exactly into the base BTC and the information bits are encoded as described in Clause 1.1 with no shortening. If this is not the case, then K is less than $k_x \times k_y$, and the BTC made up from the component codes (n_x, k_x) and (n_y, k_y) must be shortened.

Step 2. Determine the shortening parameters.

Determine b_i, using

$$b_i = \left(k_x - floor\left(\frac{i+1}{2}\right)\right)\cdot\left(k_y - floor\left(\frac{i}{2}\right)\right) - K \quad 0 \le i < 2k_y - 1$$

and choosing the integer i that assigns b_i its lowest positive integer outcome. I_x, I_y and B are then calculated using Equation 2.

$$I_x = floor\left(\frac{i+1}{2}\right)$$

$$I_y = floor\left(\frac{i}{2}\right)$$

$$B = (k_x - I_x)(k_y - I_y) - K$$

Equation 2

With these shortening parameters, exactly K information bits may be used by the shortened BTC. The information bits are encoded according to Clause 1.1 with shortening.