**IEEE P802.15**

**Wireless Personal Area Networks**

|  |  |
| --- | --- |
| Project | IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)  |
| Title | Text proposal for High Bandwidth PHY |
| Date Submitted | July 12, 2018, revised January 13, 2019 |
| Source | Volker Jungnickel (Fraunhofer HHI) | Voice: [ ]Fax: [ ]E-mail: [ ] |
| Re: | [CfP for HB PHY in IEEE P802.15.13] |
| Abstract | [Proposal for High Bandwidth PHY in 802.15.13]  |
| Purpose | [Inform TG13 about HB-PHY.] |
| Notice | This document has been prepared to assist the IEEE P802.15. 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 acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15. |

## High Bandwidth PHY

The High Bandwidth (HB) PHY enables data rates between 10 Mbit/s and 10 Gbit/s. The main approach to achieve very high data rates is to combine high optical clock rate (OCR) with high spectral efficiency. This approach serves applications where high data rates are needed, e.g. for augmented and virtual reality (AR/VR) in public and private premises. Direct current (DC) biased orthogonal frequency multiplexing (OFDM) is used at several optical clock rates (OCR) together with low density parity check codes (LDPC) and quadrature amplitude modulation (QAM) with variable size M of the constellation alphabet on each sub-carrier. Controlled by higher layers, the HB PHY includes means to adapt the data rate to varying channel conditions by modifying i) the QAM alphabet size M per OFDM subcarrier / subcarrier group, ii) the code rate and iii) the best set of transmitters. The numerology is defined in Table 1.

### **General Information**

|  |
| --- |
| **High Bandwidth PHY Operating Modes[[1]](#footnote-1)** |
| **Modulation** | DC biased OFDM |
| **Subcarrier Spacing *FSC*** | 195.3125 KHz |
| **OFDM Symbol Duration** | 5.120 ns |
| **Cyclic Prefix length for header (payload)** | 1.280 (k\*160) [[2]](#footnote-2) ns |
| **Number of bits/subcarrier** | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 |
| **FEC, Information Block Size** | LDPC, 120 or 540 bytes |
| **Code Rates** | 1/2, 2/3, 5/6, 16/18, 20/21 |
| **OCR / MHz** | **Clock cycle / ns** | **Frequency up-shift *FUS* / MHz** | ***N*(*Nsupported*) /** **clock cycles[[3]](#footnote-3)** | **Gross data rate / Mbit/**s |
| **Min.** | **Max.** |
| 25 | 40 | 12.5 | 128 (117) | 11 | 253 |
| 50 | 20 | 25 | 256 (245) | 23 | 530 |
| 100 | 10 | 50 | 512 (501) | 47 | 1084 |
| 200 | 5 | 100 | 1024 (1013) | 96 | 2192 |
| 400 | 2 | t.b.d.[[4]](#footnote-4) | ca. 200 | ca. 4000 |
| 1000 | 1 | ca. 475 | ca. 10000 |

**Table 1 Numerology for High Bandwidth PHY**

OCR in Table 1 are obtained from a common reference clock of 200 MHz available from low-cost off-the-shelf crystal oscillators by dividing it as 200 MHz/2n where n=0, 1, …5. The reference clock can be obtained via Ethernet using the precision time protocol (PTP) defined in IEEE std. 1588v2. Jitter can be improved by combining PTP with synchronous Ethernet (SynchE) defined in ITU-T rec. G.8262.

The general structure of the HB PHY transmitter is shown in Figure 1.



**Figure 1 Structure of the HB PHY transmitter**

 The HB PHY supports LDPC-based forward error correction, adaptive bit-loading, based on feedback information provided over the reverse link, and DC-OFDM modulation.

* 1. **Forward Error Correction**

The structure of the Forward error correction (FEC) is shown in Figure 2. Header and payload bits of the incoming frame are first scrambled and then encoded using a low-density parity-check (LDPC) encoder. After the LDPC encoder, the header and payload are each segmented into an integer number of symbol frames that can be handed over to Adaptive Bitloading.



**Figure 2 Structure of forward error correction unit**

* + 1. **Scrambler**

All data starting from the first bit of the PHY-frame header and ending by the last bit of the payload shall be scrambled with a pseudorandom sequence generated by the linear feedback shift register (LFSR) with the polynomial *p*(*x*) *= x*23 + *x*18 + 1, as shown in Figure 3.



**Figure 3 Scrambler for the header and payload data**

For the scrambling of the header data, the LFSR generator is initialized at the first bit of the header with the initialization vector 2AAAAA16 (where the LSB corresponds to C1).

For the scrambling of the payload data, if the scrambler initialization (SI) field in the PHY-frame header is not equal to zero, a second initialization is performed. The method for generating SI values is out of Scope of this Standard. For a second initialization, the first four bits of the LFSR (C1 to C4) may be set to the value of SI=C4C3C2C1, while all other bits C5 to C23 are set to 1. The first bit to be scrambled is XOR'ed with the first bit generated by the LFSR after initialization (i.e., C18 ⊕ C23 of the initialization vector).



**Figure 4 LDPC encoder**

* + 1. **LDPC encoder**
			1. **Low-density parity-check code**

The FEC encoder is shown in Figure 4. It consists of a systematic Quasi-Cyclic Low-Density Parity-Check Block-Code (QC-LDPC-BC) encoder and a puncturing mechanism. The parameters of the FEC encoder are the number of incoming information bits, *K* (information block of bits), the number of coded bits, *NM* (coded block of bits), the number of parity-check bits, *NM* – *K,* the number of output bits, *NFEC* ≤ *NM*, (FEC codeword, whose size depends on the puncturing pattern), the mother code rate, *RM* = *K*/*NM*, defined as the code rate before puncturing and the code rate, *R*=*K*/*NFEC*, defined as the code rate after puncturing. The information block size shall be one of the values specified in Table 3.

The encoder supports mother codes with rates *RM* = 1/2, *RM* = 2/3 and *RM* = 5/6. From these mother codes, codes with higher code rates are obtained through puncturing. The puncturing block shall support patterns providing all code rates presented in Table 3. The codeword at the output of the puncturing block is of size *NFEC* ≤ *NM*. The bits shall be output in the ascending order of codeword indices determined by vector **v**' (see below), with this order the first information bit input to the encoder will be the first at the output of the puncturing.

The code rate of the mother code, *RM* = *K/NM*, is determined by a (*NM – K*) × *NM* size parity-check matrix composed by an array of *c* × *t* circulant *b* × *b* sub-matrices **A***i,j*

The parameters *c*,*t* (0 < *c* ≤ *t*) imply a rate *RM* = (*t* – *c*)/*t*. By selecting different sets of *c*,*t*, different rates can be obtained. The sub-matrices **A***i,j* are either a rotated identity or a zero matrix and have a size of *b* × *b*, where parameter *b* = *NM/t* is called the expansion factor of **H** and controls the code block size, *NM*. The parity-check matrix, **H**, is described in its compact form as

A zero sub-matrix in position (*i*,*j*) is labelled with *ai,j* = –1, and a rotated identity sub-matrix is labelled with a positive integer number *ai,j* defining the number of right column shifts of the identity matrix. One matrix for each mother code rate and block size is defined in the following.

The compact form Hc of parity-check matrix (1/2)H corresponding to mother code with rate *RM*= 1/2 (*t* = 24, *c* = 12) and number of coded bits *NM* = 336 is given as

-1 -1 -1 6 -1 -1 9 6 -1 -1 2 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

-1 0 -1 -1 -1 3 -1 12 1 -1 -1 3 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1

-1 9 11 -1 -1 13 -1 -1 2 12 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1

 1 -1 -1 11 -1 -1 7 -1 -1 -1 11 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1

-1 -1 -1 4 8 -1 -1 -1 -1 -1 2 5 4 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1

-1 3 0 -1 -1 8 -1 -1 1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1

-1 -1 -1 0 6 -1 -1 -1 -1 5 13 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1

-1 -1 -1 9 -1 -1 -1 3 -1 -1 3 1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1

 9 0 13 -1 -1 12 -1 -1 8 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1

-1 5 -1 -1 1 4 -1 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1

-1 -1 -1 8 -1 -1 8 -1 -1 9 0 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0

10 11 -1 -1 -1 3 -1 -1 0 -1 -1 -1 4 8 -1 -1 -1 -1 -1 -1 -1 -1 -1 0

The compact form Hc of parity-check matrix (1/2)S corresponding to mother code with rate *RM*=1/2 (*t* = 24, *c* = 12) and number of coded bits *NM* = 1920 is given as

27 -1 -1 -1 55 19 -1 30 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

-1 -1 0 -1 1 -1 70 -1 47 -1 62 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1

-1 -1 41 -1 -1 -1 44 -1 -1 59 60 25 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1

16 77 -1 -1 -1 5 -1 48 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1

-1 -1 -1 45 -1 27 -1 46 19 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1

-1 -1 63 -1 -1 -1 55 -1 -1 -1 48 26 10 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1

-1 -1 -1 42 -1 21 -1 58 -1 41 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1

-1 -1 -1 -1 78 0 -1 7 52 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1

-1 29 9 -1 -1 -1 37 -1 -1 -1 35 21 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1

-1 -1 22 72 -1 -1 47 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1

35 -1 -1 -1 -1 13 -1 35 -1 70 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0

-1 46 28 -1 -1 -1 38 -1 -1 -1 8 -1 10 58 -1 -1 -1 -1 -1 -1 -1 -1 -1 0

The compact form Hc of parity-check matrix (1/2)L corresponding to mother code with rate *RM*= 1/2 (*t*= 24, *c*= 12) and number of coded bits *NM* = 8640 is given as

-1 34 -1 95 -1 279 -1 -1 -1 -1 248 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

-1 -1 0 -1 0 -1 -1 -1 -1 134 356 275 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1

51 -1 27 -1 -1 -1 -1 -1 22 152 -1 57 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1

-1 124 -1 290 -1 281 15 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1

-1 340 -1 99 336 -1 -1 1 -1 -1 -1 -1 33 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1

163 -1 46 -1 -1 -1 -1 -1 -1 306 -1 86 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1

-1 185 -1 24 -1 -1 -1 94 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1

-1 223 -1 225 325 -1 -1 -1 -1 -1 297 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1

46 -1 314 -1 -1 -1 59 -1 -1 67 -1 120 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1

-1 -1 121 -1 -1 -1 -1 161 -1 303 -1 264 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1

-1 303 -1 8 -1 185 -1 -1 138 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0

-1 -1 312 -1 -1 -1 100 -1 -1 144 -1 307 33 166 -1 -1 -1 -1 -1 -1 -1 -1 -1 0

The compact form Hc of parity-check matrix (2/3)S corresponding to mother code with rate *RM* = 2/3 (*t*= 24, *c*= 8) and number of coded bits *NM* = 1440 is given as

49 -1 -1 21 31 -1 57 -1 -1 19 -1 29 2 -1 19 -1 -1 0 -1 -1 -1 -1 -1 -1

-1 7 22 -1 -1 37 -1 32 10 -1 26 -1 -1 59 -1 48 -1 0 0 -1 -1 -1 -1 -1

53 -1 -1 20 50 -1 -1 3 16 -1 49 -1 -1 28 14 -1 -1 -1 0 0 -1 -1 -1 -1

-1 58 23 -1 -1 15 54 -1 -1 5 -1 18 49 -1 -1 13 -1 -1 -1 0 0 -1 -1 -1

55 -1 -1 58 -1 9 -1 26 57 -1 41 -1 31 -1 21 -1 -1 -1 -1 -1 0 0 -1 -1

-1 10 49 -1 59 -1 7 -1 -1 30 -1 18 -1 48 -1 7 59 -1 -1 -1 -1 0 0 -1

48 -1 -1 50 18 -1 -1 11 52 -1 59 -1 -1 37 -1 10 0 -1 -1 -1 -1 -1 0 0

-1 24 16 -1 -1 0 53 -1 -1 41 -1 38 51 -1 58 -1 59 8 -1 -1 -1 -1 -1 0

The compact form Hc of parity-check matrix (2/3)L corresponding to mother code with rate *RM* = 2/3 (*t*= 24, *c*= 8) and number of coded bits *NM* = 6480 is given as

78 -1 -1 167 237 -1 3 -1 266 -1 -1 102 153 -1 -1 212 -1 0 -1 -1 -1 -1 -1 -1

-1 83 189 -1 -1 68 -1 178 -1 90 205 -1 -1 13 4 -1 -1 0 0 -1 -1 -1 -1 -1

-1 226 147 -1 46 -1 -1 76 -1 116 -1 211 -1 112 -1 118 -1 -1 0 0 -1 -1 -1 -1

92 -1 -1 214 -1 236 241 -1 157 -1 143 -1 214 -1 207 -1 -1 -1 -1 0 0 -1 -1 -1

144 -1 -1 258 264 -1 53 -1 114 -1 172 -1 -1 82 262 -1 62 -1 -1 -1 0 0 -1 -1

-1 153 120 -1 -1 199 -1 126 -1 61 -1 183 15 -1 -1 134 -1 -1 -1 -1 -1 0 0 -1

-1 100 -1 141 -1 36 -1 17 -1 156 -1 124 162 -1 -1 57 0 -1 -1 -1 -1 -1 0 0

196 -1 187 -1 73 -1 80 -1 139 -1 57 -1 -1 236 267 -1 62 256 -1 -1 -1 -1 -1 0

The compact form Hc of parity-check matrix (5/6)S corresponding to mother code with rate *RM* = 5/6 (*t*= 24, *c*= 4) and number of coded bits *NM* = 1152 is given as

-1 13 32 47 41 24 -1 25 22 40 1 31 8 15 20 15 42 30 13 3 -1 0 -1 -1

25 46 15 43 45 29 39 47 23 38 39 12 -1 21 -1 38 33 0 0 -1 39 0 0 -1

35 45 45 38 14 16 6 11 -1 18 7 41 35 17 32 45 41 -1 18 17 0 -1 0 0

 9 32 6 22 26 31 9 8 22 32 40 4 18 40 36 -1 -1 23 31 41 39 20 -1 0

The compact form Hc of parity-check matrix (5/6)L corresponding to mother code with rate *RM* = 5/6 (*t*= 24, *c*= 4) and number of coded bits *NM* = 5184 is given as

 -1 47 146 203 184 112 -1 116 103 181 3 140 38 68 91 70 191 138 62 14 -1 0 -1 -1

117 203 67 194 206 133 174 212 104 171 176 56 -1 96 -1 167 149 4 1 -1 177 0 0 -1

153 206 198 173 55 72 28 53 -1 82 34 186 161 80 144 204 187 -1 84 77 0 -1 0 0

 44 147 27 83 118 130 41 38 100 146 183 19 85 180 163 -1 -1 106 140 185 177 94 -1 0

* + - 1. **Encoding**

One method of encoding is to determine a systematic generator matrix **G** from **H** such that **GHT**= **0**. A *K*-bit information block **u**=[*u*0, *u*1,...,*u*K-1] can be encoded by the systematic generator matrix **G** via the operation **v**= **uG** to become a *NM*-bit coded block **v** = [**u** | **p**] = [*u*0, *u*1,...,*u* NM -1]**,** where **p**=[*p*0, *p*1,...,*p*NM-K-1]are the parity-check bits. Encoding an LDPC code from **G** can be quite complex. However, the QC-LDPC-BC codes specified here are such that very low complexity encoding directly from **H** is possible.

The encoder supports the coded block sizes and rates presented in Table 3. The parity-check matrix **H** used to encode a block of information bits is selected according to the mother code indicated in Table3. The encoding is processed as follows:

1) A group of incoming *K* information bits **u**=[*u*0, *u*1,...,*u*K-1] are collected and copied to the output of the encoder to form a block of systematic code bits.

2) *NM–K* parity-check bits, **p**=[*p*0, *p*1,...,*p*NM-K-1] are computed using the parity-check matrix **H** and the information block **u**. The resulting coded block **v**= [**u** | **p**] satisfies the parity check equations **vHT**=**0**. Here **0** is a zero row vector of dimension *NM*-*K*.

3) The *NM*-*K* parity check bits **p** are copied to the output of the encoder as a block of parity check bits **p**=[*p*0, *p*1,...,*p*NM-K-1] to form the output coded block **v** = [**u** | **p**] = [*v*0, *v*1,...,*v* NM -1].

4) The output of the encoder **v** is the input to the puncturing block (see Figure 4).

* + - 1. **Puncturing**

Puncturing discards some of the coded block bits to achieve a higher code rate (*R*). Puncturing is applied to both information and parity-check bits. The puncturing block uses the puncturing patterns specified in Table 2. The puncturing patterns are denoted as , where *T* is the length of the puncturing pattern and *i* is the number of zeros in the pattern. The first pattern does not result in any code rate changes and is introduced to be consistent with the puncturing notation.

|  |  |
| --- | --- |
|  | **Puncturing pattern** |
|  | [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]  |
|  |  |
|  |  |
|  |  |
|  |  |

**Table 2 Puncturing patterns**

The coded block **v** input to the puncturing block is processed using the puncturing pattern as follows. For the pattern , the puncturing block omits all incoming coded bits *vt*,t=0,…,NM-1 for which . The resulting output FEC code word is **v**’ = [*v*0, *v*1,..., *v* NFEC -1] with *NFEC* ≤ NM.

* + - 1. **FEC Encoding parameters**

The FEC encoding scheme supports the encoding parameters specified in Table 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Code rate, *R***  | **Information block size, *K***  | **Puncturing pattern, PP** | **Mother code matrix** | **FEC codeword size, *NFEC*** |
| For header | 1/2 | PHYH = 168 |  | (1/2)H | 336 |
| For payload | 1/2 | 960 |  | (1/2)S | 1920 |
| 1/2 | 4320 |  | (1/2)L | 8640 |
| 2/3 | 960 |  | (2/3)S | 1440 |
| 2/3 | 4320 |  | (2/3)L | 6480 |
| 5/6  | 960 |  | (5/6)S | 1152 |
| 5/6 | 4320 |  | (5/6)L | 5184 |
| 16/18 | 960 |  | (5/6)S | 1080 |
| 16/18 | 4320 |  | (5/6)L | 4860 |
| 20/21 | 960 |  | (5/6)S | 1008 |
| 20/21 | 4320 |  | (5/6)L | 4536 |

**Table 3 FEC Encoding parameters**

* 1. **Adaptive Bitloading**

The incoming symbol frames are first mapped onto OFDM subcarriers (tones). Tone mapping is also performed on unused subcarriers which are stuffed with random data. This may be important in interference-limited environments (i.e. where multiple OWC signals are received). The resulting list of symbols on each tone is mapped onto complex-valued IQ constellation points where the modulation alphabet depends on the number of bits mapped onto each tone. According to the PPDU format described in Section 1.2, preambles for synchronization and channel estimation are multiplexed in time with header information and optional fields which contain MIMO preambles as well as payload data before passing all signals into the OFDM modulator. In the following, the OFDM modulator characteristics is described.

* + 1. **Subcarrier spacing and indexing**

The subcarrier spacing *FSC* is the frequency spacing between any two adjacent subcarriers. The physical index *i* corresponds to the order of subcarriers in ascending frequency. The subcarrier with physical index *i* shall be centered at frequency *f* = *FUS* – (*N*/2 – *i*) × *FSC*. The index *i* goes from 0 to *N* – 1. The logical index indicates the order in which data is loaded on subcarriers. In the HB PHY the physical index and the logical index are the same, i.e., the subcarriers are loaded in order of ascending frequency.

Not all subcarriers may always be used for data transmission; some of them may be switched off. Others may be only used with reduced power. Particular subcarriers used for data transmission depend on channel characteristics, such as attenuation and noise. The related functions are performed by subcarrier masking and gain scaling. For the purpose of tone mapping, the following types of subcarriers are defined.

1. Supported subcarriers (SSCs) are those on which transmission is allowed.
	1. Active subcarriers (ASCs) have loaded bits (*b* ≥ 1) for data transmission, are subject to constellation point mapping, constellation scaling and constellation scrambling.
	2. Inactive subcarriers (ISCs) do not have data bits loaded (e.g., because SNR is low), can be used for measurement or other purposes and are subject to transmit power shaping.
2. Masked subcarriers (MSCs) are those on which transmission is not allowed, i.e., the gain on this subcarrier is set to zero. In this specification, only permanently masked subcarriers (PMSCs) are considered which are never allowed for transmission.

### **Tone mapping**

The tone mapper divides the incoming symbol frames into groups of bits and associates each group of bits with special subcarriers onto which these groups are loaded. This information along with subcarrier-specific gain scaling values are passed to the constellation encoder.

Tone mapping is defined by a bit allocation table (BAT). It associates subcarrier indices with the number of bits to be loaded on a subcarrier. The BAT is indicated to the receiving node in the BAT\_ID field in the PHY header. Up to 32 BAT\_IDs can be defined as shown in Table 4.

|  |  |  |
| --- | --- | --- |
| **BAT\_ID** | **Type** | **Content** |
| 0 | predefined | uniform 1-bit loading on all subcarriers, except MSC |
| 1 | predefined | uniform 2-bit loading on all subcarriers, except MSC |
| 2 to 7 | predefined | reserved |
| 8 to 31 | runtime | see below |

**Table 4 Bit allocation tables used in HB PHY**

A runtime BAT indicates the number of bits loaded on each subcarrier. Runtime BAT can be defined by the transmitter or receiver. Transmitter-defined BATs are typically pre-defined. For runtime BATs the number of bits is variable in general and depends on the signal-to-noise ratio (SNR) on each subcarrier. The BAT is always suggested by the receiver and communicated to the transmitter using the feedback protocol. A runtime BAT uses subcarrier grouping of *G* = 1 (no grouping as default), 2, 4, 8, and 16 subcarriers on consecutive frequencies where all subcarriers of the same group use the same bit loading. If subcarriers in a group are masked, bit loading is applied to the supported subcarriers (SSCs) only.



**Figure 5 LFSR for modulation of unloaded supported subcarriers.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| k | Sk | k | Sk | k | Sk | k | Sk |
| 1 | 7FFFFF16 | 17 | 07628716 | 33 | 03714416 | 49 | 1BEDE616 |
| 2 | 26B48916 | 18 | 3E1A3116 | 34 | 27858716 | 50 | 608D6B16 |
| 3 | 278A9116 | 19 | 05DE6D16 | 35 | 2CF7F716 | 51 | 4B75D316 |
| 4 | 15F4ED16 | 20 | 5C5B4E16 | 36 | 027D4616 | 52 | 22BA6416 |
| 5 | 5B4CB116 | 21 | 59641316 | 37 | 70A7EB16 | 53 | 7D064616 |
| 6 | 2F021F16 | 22 | 0613D916 | 38 | 4C622C16 | 54 | 7F56E616 |
| 7 | 7A64C116 | 23 | 19504A16 | 39 | 54DC6816 | 55 | 61433316 |
| 8 | 414CD716 | 24 | 50FDE016 | 40 | 01715E16 | 56 | 4F136816 |
| 9 | 649D5E16 | 25 | 5CD04816 | 41 | 274A7B16 | 57 | 7359EF16 |
| 10 | 13482616 | 26 | 66C64616 | 42 | 55238D16 | 58 | 2D86A916 |
| 11 | 2A3DFC1 | 27 | 7169B316 | 43 | 008B0616 | 59 | 25373D16 |
| 12 | 2B957016 | 28 | 48049716 | 44 | 3FA25516  | 60 | 25846616 |
| 13 | 3C677716 | 29 | 053FE316 | 45 | 777A6A16 | 61 | 4CE92A16 |
| 14 | 75798616 | 30 | 51F1B116 | 46 | 5154DD16 | 62 | 6B7E3D16 |
| 15 | 10396216 | 31 | 7D2BA016 | 47 | 55C20316 | 63 | 760B3416 |
| 16 | 0DB87B16 | 32 | 11E4D816 | 48 | 0D21F916 | 64 | 761EA616 |

**Table 5 Example LFSR seeds for an initial seed of 7FFFFF16**

Unloaded SSC are loaded with a pseudorandom binary sequence defined by the linear feedback shift register (LFSR) generator with the polynomial *p*(*x*) *= x*23 *+ x*18 *+* 1 shown in Figure 2.

The LFSR generator is initialized at the beginning of each OFDM symbol by using an initial seed assigned by the MAC. The *i*th payload symbol uses the modified seed *Sk* where *k* is equal to (*i*–1, modulo 64) + 1, where *i* = 1, 2, 3, 4,.... *Sk* is generated by advancing the LFSR by 8192\*(*k*-1) from the original seed.

An example of LFSR seeds for an initial seed of 7FFFFF16 is provided in Table 3. Seeds *S*1 to *S*64 are used to initialize the LFSR for payload symbols 1-64, 65-128 and so on. The LSB of the seed S*k* corresponds to *c*1.

The LFSR is advanced by two bits for each subcarrier (for both SSC and MSC) of each symbol of the payload. Two LFSR bits corresponding to the subcarrier index 0 are (*c*1, *c*2) of the initialization seed. Two LFSR bits corresponding to the subcarrier index 1 are (*c*1, *c*2) after two shifts, and so on. For modulation of unloaded subcarriers, symbols in optional fields are treated in the same way as payload symbols.

The modulation of inactive subcarriers (ISC, not loaded with encoded payload bits) is as follows

1) Starting at the beginning of the first payload OFDM symbol, each subcarrier from the ISC set is modulated with the two bits which are the LSBs of the LFSR, *c*1, and *c*2 using the 2-bit constellation mapping defined in 1.1.3 (*c*1 is transmitted first).

2) In every OFDM symbol of payload, if the number of bits in the symbol frame does not fill the entire symbol, the bits from the LFSR is used to fill the remainder of the symbol frame, by taking the sequential groups of *m* LSBs of the LFSR and mapping them on to the remaining subcarriers so that LSB of LFSR is transmitted first and in the order defined by the current BAT, where *m* is the number of bits allocated for that subcarrier by the BAT. For the first padded subcarrier, if *n* bits of the *m* loaded bits are data bits (*n* < *m*), these *n* data bits are loaded as the LSBs of the group of bits mapped on the constellation point, and the *m-n* bits of the LFSR are used as the MSBs of the group of bits mapped on the constellation point starting from LSB of LFSR.

3) In the case of a PROBE frame, starting at the beginning of the first payload OFDM symbol, each subcarrier from the ISC set is modulated with the two bits which are the LSBs of the LFSR, *c*1 and *c*2, using 2-bit constellation mapping defined in 1.1.3 (*c*1 is transmitted first).

The bits from LFSR are loaded on subcarriers in the order of logical indices (i.e., in the same way as data is loaded over payload symbols), according to subcarrier indexing defined in 1.1.1. Modulation of unloaded subcarriers starts from the unloaded SSC with the lowest logical index of the first payload symbol, continue in ascending order of logical indices until the unloaded SSC with the highest logical index of the first payload symbol, continue with the unloaded SSC with the lowest logical index of the second payload symbol, continue in ascending order of logical indices until the unloaded SSC with the highest logical index of the second payload symbol, and continue until the unloaded SSC with the highest logical index of the last payload symbol.

The ASCs are loaded according to the corresponding BAT as defined above.

### **Constellation encoder**

Constellation mapping associates every data symbol with the values of *I* (in-phase component) and *Q* (quadrature-phase component) of a constellation point in the plane of complex numbers. Each group of bits consists of *b* bits {*d*b–1, *d*b–2, … *d*0} mapped onto the constellation mapper with the LSB bit, *d*0, first.

For **even number numbers of bits**, *b, i.e.* (2, 4, 6, 8, 10, 12), square-shaped constellations are used. Support of all even order constellations is mandatory at both the transmitter and the receiver. With square-shaped constellations, 2*b* constellation points are set as a square.

For *b*=2, bit *d0* is mapped to I branch as {0, 1}🡪{-1, 1} and same way d1 is mapped to Q branch.

For *b*=4, bits [d1 d0]aremapped to the I branch as {00, 10, 11, 01}🡪{-3, -1, 1, 3} and same way bits [d3 d2]aremapped to the Q branch.

For *b* ≥ 4 constellations are derived as follows:

1) Divide the incoming group of *b* bits into two equal subgroups, so that *b*/2 LSBs form the *I*-subgroup and *b*/2 MSBs form the *Q*-subgroup.

2) Compute values of *I* and *Q* for the incoming group {*db*–1, *db*–2, … *d*0} as:

*I* = *sgnI* × *valI*

*Q* = *sgnQ* × *valQ* where

*sgnI = 2 × d0 – 1,*

*sgnQ* = 2 × db/2 –1,

*valI* = *|Ib*–2 – 2*b*/2–1| and *valQ* = *|Qb*–2 – 2*b*/2–1|

where |.| is the absolute value. *Ib–*2 and *Qb*–2 are the values of *I* and *Q* for the incoming (*b*-2)-bit group {*db*-–1, *db*–2, … *db*/2+1, *db*/2–1, … *d*1}, i.e., with *d*0 and *d*b/2 being removed.

For **odd numbers of bits**, (1, 3, 5, 7, 9, 11), specific constellations are used. Support of all odd constellations is mandatory at the transmitter, while with *b* ≥ 5 they are optional at the receiver.

For *b*=1, bit *d0* is mapped to I branch as {0, 1}🡪{-1, 1}.

For *b*=3, if *d1=*1, bit *d0* is mapped to I branch as {0, 1}🡪{-1, 1} and same way d2 is mapped to Q branch. If *d1=*0, bit groups [*d2* *d0*] map as [0 1] 🡪 {3, -1}, [0 0] 🡪 {-1, -3}, [1 1] 🡪 {1, 3}, and [1 0] 🡪 {-3, 1}.

For *b* > 3, cross-shaped constellations are used. First, 2*b* constellation points form a rectangle, with *MI* = 2*B*1 columns (*MI* points on the *I*-axis) and *MQ* = 2*B*2 rows (*MQ* points on the *Q*-axis), where *B*1 = *ceiling*(*b*/2) and *B*2 = *floor*(*b*/2). The mapping of points uses following steps

1. Divide the incoming group of bits into two subgroups, so that *B*1 LSBs form the first subgroup (*I*-group) and *B*2 MSBs form the second subgroup (*Q*-group); both subgroups are incoming LSBs (which are *d*0 and *dB*2+1, respectively) first.
2. Compute values of *I* and *Q* for the incoming group {*db*–1, *db*–2, … *d*0} as

*I* = *sgnI* × *valI*

*Q* = *sgnQ* × *valQ*

*sgnI* = 2 × *d*0 – 1

*sgnQ* = 2 × *d*B1 – 1

*valI* = *|I*2*×B*1| and *valQ* = *|Q2×B2*|

*I*2*×B*1 is the value of *I* for (2×*B*1)-bit group {0, *db*–1, *db*–2, … *d*0} and *Q*2*×B*2 is the value of *Q* for (2×*B*2)-bit group {*db*–1, *db*–2, … *d*1} computed as defined for even constellations.

1. Transform *s* = (*MI* – *MQ*)/4 columns of constellation points in each quadrant having highest absolute values of *I* (positive or negative) into rows of *Q* by changing their {*I*, *Q*} coordinates to {*I'*, *Q'*} in the following way:

|*Q'*| = |*I*| - 2*s*, and sign (*Q'*) = sign (*I*)

|*I'*| = *MQ* – |*Q*|, and sign (*I'*) = sign (*Q*).

 **Constellation point scaling** is done as follows. Each constellation point (*I*, *Q*), corresponding to the complex value *I + jQ* at the output of the constellation mapper, is scaled by the power-normalization factor χ(*b*) and the frequency-domain spectrum shaping coefficient *tss*

The values (*I, Q*) for each constellation point are scaled so that all constellations have the same average power. The **power normalization factor**, χ(*b*), for *b*-bit loading is given in Table 4.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *b* | 1 | 2 | 3 | 4 | 5 | 6 |
| χ(*b*) | 1 | 1/√2 | 1/√6 | 1/√10 | 1/√20 | 1/√40 |
| *b* | 7 | 8 | 9 | 10 | 11 | 12 |
| χ(*b*) | 1/√82 | 1/√170 | 1/√330 | 1/√682 | 1/√1322 | 1/√2730 |

**Table 4 Power normalization factor**

**Frequency-domain transmit spectrum shaping** is achieved by a scaling factor *tss* defined for each subcarrier. The *tss* values are set by the transmitter between 0 and -30 dB in steps of -0.5 dB. Smaller values of *tss* provide more attenuation. The value *tss* = 0 dB corresponds to no attenuation on the particular subcarrier. If no spectrum shaping is applied, all *tss* values shall be equal to 0 dB. The values of *tssi* are relevant only for supported and ignored for masked subcarriers.



**Figure 6 LFSR generator for the constellation scrambler.**

### **LFSR for constellation scrambling**

The phase of constellation points obtained from the constellation mapper are shifted according to the pseudorandom bit sequence (PRBS) generated by a linear feedback shift register (LFSR), as shown in Figure 2.

The LFSR generator implements the polynomial *g*(*x*) = *x*13 + *x*12 + *x*11 + *x*8 + 1. The LFSR is advanced by two bits for supported and masked subcarriers in the order specified in Figure 2. The two LSBs of the LFSR are taken to determine the phase shift of the constellation scrambler as shown in Table 2.

|  |  |
| --- | --- |
| **LFSR output** | **Phase shift (rad)** |
| s2 | s1 |  |
| 0 | 0 | 0 |
| 0 | 1 | π/2 |
| 1 | 0 | π |
| 1 | 1 | 3 π/2 |

**Table 6 Constellation phase shift depending on LFSR output**

* 1. **OFDM Modulator**

The functional diagram of the OFDM modulator is presented in Figure 4. The OFDM modulator consists of an *N*-point IDFT, prepending of a cyclic prefix (CP) of length *N*CP and frequency up-shift by *F*US where *N, NCP* and *F*us all depend on the OCR as indicated in Table 1. All aspects of the signal processing used in the modulator complies with the following equations and text.



**Figure 7 OFDM modulator for HB PHY**

The IDFTconverts the stream of the *N* complex numbers *Zi,l* at its input into the stream of *N* complex time-domain samples *Xn,l*. The number of IDFT samples *N* is the same for all symbols of the frame. The input numbers *Zi,l* represent *N* mapped data symbols, where the *i*th data symbol is mapped onto the *i*th modulated subcarrier of the *l*th OFDM signal, where *i* = 0, 1, … *N*–1 is the subcarrier index and *l* is the number of the OFDM symbol. *N*-point IDFT is given by

.

The value of *Zi,l* for masked subcarriers (MSC) is set to 0.

The CP provides a guard interval between adjacent OFDM symbols to protect against inter-symbol interference (ISI). The CP is implemented by prepending the last *NCP* samples of the IDFT output to its output *N* samples. The order of samples is as follows. The first sample is the IDFT output sample *N*-*N*CP*.* The last sample of the CP is the IDFT output sample *N-*1. The next sample is the IDFT output sample 0. Total OFDM symbol duration is *NW* = *N* + *NCP* samples. The value of *NCP* varies during a frame. In the header and first two symbols after the header, *NCP=NGI-HD=N/4*. All the rest of the payload has the same value *NCP=NGI.*

Frequency up-shift moves the spectrum of the transmit signal by *FUS* = *m*\**FSC*given in Table 1 where *m*=*N*/2. Real and imaginary components of the signal after frequency up-shift are given by

.

The transmitted real-valued output signal is given by

where *Xn*/*p* is *Xn* after interpolation with interpolation factor *p* being equal to or higher than 2. The phase of the up-shift should be initialized to zero at the first sample of the preamble and advanced by per each sample after interpolation.

The OFDM modulator provides a bipolar output signal. Conversion into a unipolar waveform is performed in the optical frontend and not in the OFDM modulator. For driving an LED, unipolar DC-biased OFDM signals are obtained by i) adding a constant bias (which may include the dimming), ii) setting the average modulation amplitude as a constant fraction of the bias also denoted as the modulation index µ≤1. The optical frontend will inherently force below-zero signals to zero optical output, clipping negative parts of the waveform.

* 1. **PPDU format**

Preamble

Channel estimation

SHR

PHY header

HCS

Optional Fields

PHR

PSDU

PHY payload

**Figure 8 PPDU format for High Bandwidth PHY**

The PM PHY uses the PPDU format shown in Figure 1. It consists of a synchronization header (SHR), physical layer header (PHR) and PHY payload (PSDU).

* 1. **Transmission**
		1. **Synchronization Header (SHR)**
			1. **Preamble**

The HB PHY synch preamble consists of two sections. The first section comprises 10 repetitions of an OFDM symbol ***S*** employing subcarrier spacing *4* × *FSC*, where *F*SC denotes the subcarrier spacing. The second section comprises 4 repetitions of the inverted symbol **-*S***.

The synch preamble uses the subcarriers with indices 4, 8, 12, 16, …, N-4, except those being masked. On supported subcarriers, at first, a bit sequence of all ones is mapped using the 1-bit constellation. Next, the LFSR generator of the constellation scrambler is initialized as 16E616=1011011100110[[5]](#footnote-5). The LFSR is advanced by two bits per subcarrier in the order specified in Figure 2. The output of the mapper is rotated using the two LSBs of the LFSR, s1 and s2, as defined in Table 2 resulting in the constellation points *Zi* for subcarrier index *i*. The resulting constellation sequence is finally fed into the OFDM modulator, see 1.1.2.

* + - 1. **Channel estimation**

The channel estimation (CE) sequence is needed for equalization and subsequent detection of header and data. CE sequence allows frequency-domain equalization. The channel estimation sequence comprises 2.5 repetitions of an OFDM symbol ***S*** employing subcarrier spacing 1× *FSC*, where *F*SC denotes the subcarrier spacing. The number of repetitions is a non-integer number to indicate that two OFDM symbols are used so that the CP is the same as in the PHY header, i.e. *NCP=NGI-HD=N/4*. In the channel estimation preamble, all subcarriers with indices 0, 1, 2, …, N-1 are used, except those being masked. On supported subcarriers, at first a bit sequence of all ones is mapped using the 1-bit constellation. Next, the LFSR generator of the constellation scrambler is initialized as 110516= 1000100000101. The LFSR is advanced by two bits for supported and masked subcarriers in the order specified in Figure 2. The output of the mapper is rotated using the two LSBs of the LFSR, s1 and s2, as defined in Table 2 resulting in the constellation points *Zi* for subcarrier index *i*. The resulting constellation sequence is finally fed into the OFDM modulator, see 1.1.2.

* + 1. **Physical Layer Header (PHR)**
			1. **PHY header**

The core PHY header is PHYH bits long. It is transmitted over D=1 or D=2 consecutive OFDM symbols. The core header is composed of a common part being the same for all PHY frame types and a variable part which differs according to the PHY frame type. The PHY frame type is indicated by the FT field. The PAD fields fit the length of the header of different PHY frame types to the standard value of PHYH bits. The content of the core PHY header is protected by the 16-bit header check sum (HCS). The fields of the core header are defined in the following Table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Octet** | **Bits** | **Description** | **Reference** |
| **common part** |
| FT | 0 | [3:0] | Frame type | 1.6.2.1.1. |
| OID | [7:4] | OWPAN ID | 1.6.2.1.2. |
| SID | 1 | [7:0] | Source ID  | 1.6.2.1.3. |
| DID | 2 | [7:0] | Destination ID | 1.6.2.1.4. |
| MI | 3 | [0] | Multicast indication | 1.6.2.1.5. |
| DRI | [1] | Duration indication | 1.6.2.1.6. |
| EHI | [2] | Ext. header indication | 1.6.2.1.7. |
| HSI | [3] | Header segmentation indication | 1.6.2.1.8. |
| Reserved | [7:4] | Reserved |  |
| **variable part** |
| FTSF | 4-18 | [119:0] | Frame-type specific field | defined by MAC |
| **common part** |
| HCS | 19-20 | [15:0] | Header check sum | 1.6.2.3. |

If the extended header indication EHI=”1”, additional PHYH bits are appended to the core header representing the extended header. The extended header is encoded and segmented exactly the same way as the core header. The content of the extended header is protected by the 16-bit extended header check sequence (E\_HCS). Core and extended headers are transmitted over separate OFDM symbols.

* + - * 1. **Frame type**

The FT field indicates the PHY frame type. The following table shows different PHY frame types.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FT** | **Value (b3b2b1b0)** | **Description** | **DRI** | **ERI** |
| MAP/RMAP | 0000 | MAP/RMAP frame | 1 | 0 |
| MSG | 0001 | Data and management frame | 1 | 0 |
| ACK | 0010 | ACK control frame | 0 | 0 or 1 |
| RTS | 0011 | RTS control frame | 1 | 0 |
| CTS | 0100 | CTS control frame | 1 | 0 |
| CTMG | 0101 | Short control frame | 0 | 0 or 1 |
| PROBE | 0110 | PROBE control frame | 1 | 0 |
| ACKRQ | 0111 | ACK retransmission request frame | 0 | 0 |
| BMSG | 1000 | Bidirectional MSG frame with data, mgmt., ACK | 1 | 0 or 1 |
| BACK | 1001 | Bidirectional ACK frame with ACK data, mgmt  | 1 | 0 or 1 |
| ACTMG | 1010 | Acknowledgment for CTMG frame | 0 | 0 |
| Reserved | 1011…1110 | Reserved |  |  |
| FTE | 1111 | Frame type extension pointer to further frame types | 0 or 1 | 0 or 1 |

* + - * 1. **OWPAN ID**

OID field contains the OWPAN ID to which the source and destination devices of the PHY frame belong. OID is an unsigned 4-bit integer from 0 to 15. Value 0 is reserved for communication between OWPANs.

* + - * 1. **Source ID**

SID identifies the source node of the PHY frame during its registration in the OWPAN. SID is an 8-bit unsigned integer from 0 to 251. Value 0 is used if a node attempts to join the OWPAN. Value 251 is reserved for communication between OWPANs.

* + - * 1. **Destination ID**

DID identifies the destination node of the PHY frame. DID is an 8-bit unsigned integer from 0 to 250.

* + - * 1. **Multicast indication**

If MI=”0”, the DID identifies the destination node for unicast transmission. If MI=”1”, however, the DID is a MULTICAST\_ID or BROADCAST\_ID of the destination nodes.

* + - * 1. **Duration indication**

If DRI=”1”, the FTSF starts with a duration field. If DRI=”0”, the PHY frame contains no payload. DRI is the duration of a single PHY frame or PHY frame sequence. DRI is a 16-bit unsigned integer in steps of 0.25 μs. DRI is the smallest integer larger than or equal to the actual duration. DRI depends on the frame type as shown in the table above. If a node detects a PHY frame with unknown frame type, the node assumes for its virtual carrier sense that the channel is occupied for that duration. After that time, an inter-frame gap equal to TIFG\_MIN. applies.

* + - * 1. **Extended header indication**

If EHI=”0”, the PHY header contains PHYH information bits. If EHI=”1”, the PHY header contains 2×PHYH information bits. The EHI depends on the frame type as shown in the Table above.

* + - * 1. **Header segmentation indication**

The HSI has same value as the header segmentation field in the TXOP descriptor extension in the MAP.

* + - 1. **Header check sum (HCS)**

The header check sequence (HCS) uses CRC-16 as defined in Annex C. The HCS bits shall be processed in the transmitted order. The registers shall be initialized to all ones.

* + - 1. **Optional fields**

Optional fields contain reference symbols for multiple-input multiple-output (MIMO) channel estimation. For MIMO RS, forward error correction and HCS do not apply. MIMO RS are defined in the frequency domain. The use of MIMO RS is configurable by the MAC.

* + - * 1. **MIMO RS**

**MIMO** RSs allow orthogonal detection of multiple data streams or signals from multiple transmitters. MIMO RS are orthogonal in the frequency domain. A specific comb of subcarriers identifies a particular stream or transmitter.

Construction of FD RS starts from the LFSR also used in the synchronization preamble. FD RS for the first transmitter use the subcarriers with indices Δ, 2Δ, 3Δ, 4Δ, …, N-Δ, except those being masked, where Δ is the comb spacing and a power of 2. The value of Δ is defined by the MAC taking the fundamental relation Δ*≤N/NCP* into account. On supported subcarriers, at first, a bit sequence of all ones is mapped using the 1-bit constellation. Next, the LFSR generator of the constellation scrambler is initialized as 16E616=1011011100110[[6]](#footnote-6). The LFSR is advanced by two bits per subcarrier in the order specified in Figure 2. The output of the mapper is rotated using the two LSBs of the LFSR, s1 and s2, as defined in Table 2 resulting in the constellation points *Zi* for subcarrier index *i*.

The resulting constellation sequence is finally fed into the OFDM modulator described in 1.1.2.

The variable *CS*in the PHY header identifies other transmitters at the same coordinator. For *CS≥1*, a cyclic shift of the comb is performed such that now the subcarriers with indices *CS*, Δ*+CS*, 2Δ*+CS*, 3Δ*+CS*, 4Δ*+CS*, …, N-Δ*+CS* are used*,* except those being masked.By using a single RS, up to *CSmax* ≤Δ-1 streams or transmitters can be identified. The MAC layer will add more RSs for more streams or transmitters as indicated by NRS being a power of 2. In this way, up to NRS\*(Δ-1) streams can be identified. Decompose the identifier of the *i*th stream or transmitter as

*i*=*a*\*(Δ-1)+*b* where *b*<Δ-1.

The comb shift is then *CS*=*b* and the original RS is multiplied with the entries in the *a*th row of the MxM Hadamard matrix **H**K where M=2K. **H**K is obtained by incrementing k from k=1…K

.

* + 1. **Header encoding**



**Figure 9 Header encoder**

The header encoder is shown in Figure 9. It contains an FEC encoder and a header repetition encoder (HRE). The bits of the PHY header shall be input into the header FEC encoder in their original order and encoded as described in 1.2. The size of the FEC codeword and the coding rate of the header FEC encoder are described in Table 3. Since the coding rate used for header encoding is 1/2, the number of bits in the FEC codeword is always even, and the number of bits in the encoded header block is even.

* + - 1. **Header repetition encoder**

The FEC codeword enters the HRE operating as follows. The FEC codeword is first copied *M* times, where *M* = ceiling (*kH/NFEC*), *kH* is the number of bits to be loaded on to the OFDM symbol carrying the header. The first encoded header block is formed by concatenation of *M* copies of the header FEC encoder output. The bits (*bi*) within each codeword are cyclically shifted by 2 bits as follows:

• 1st FEC codeword copy is {*b*0, *b*1, …, *bNFEC*–2, *bNFEC*–1}.

• 2nd FEC codeword copy is {*b*2, *b*3, …, *bNFEC* –1, *b*0, *b*1}.

• 3rd FEC codeword copy is {*b*4, *b*5, …, *bNFEC*–1, *b*0, *b*1, *b*2, *b*3}.

• …

• *M*th FEC codeword copy, where *M* > 3, is {*b*(2×*M–*2), *b*(2×*M*–1), …, *bNFEC*–1, *b*0, *b*1, …, *b*(2×*M*–4), *b*(2×*M*–3)}.

The second encoded header block is formed by cyclic shifting of each copy by NFEC/2 bits and concatenation of *M* copies of the shifted FEC codeword. The bits (*bi*) within each codeword is cyclically shifted by 2 bits as follows:

• 1st FEC codeword copy is {*bNFEC/2*, *bNFEC*/2+1, …, *bNFEC*–2, *bNFEC*–1, *b*0, *b*1, …, *bNFEC*/2–2, *bNFEC*/2–1}.

• 2nd FEC codeword copy is {*bNFEC*/2+2, *bNFEC*/2+3,…, *bNFEC*–2, *bNFEC*-1, *b*0, *b*1, …, *bNFEC*/2, *bNFEC*/2+1}.

• 3rd FEC codeword copy is { *bNFEC*/2+4, *bNFEC*/2+5, …, *bNFEC*–2, *bNFEC*-1, *b*0, *b*1, …, *bNFEC*/2+2, *bNFEC*/2+3}.

• …

• *M*th FEC codeword copy, where *M*>3, is {*bNFEC*/2+(2×M-2), *bNFEC*/2+(2×M–1), …, *bNFEC*–1, *b*0, *b*1, …, *bNFEC*/2+(2×M–4), *bNFEC*/2+(2×M–3)}.

* + - 1. **Header segmentation**

The encoded header block from the output of the header encoder is segmented into symbol frames. The maximum number of bits in the symbol frame does not exceed the values of *kH* for header symbol frames. Header symbol frames shall be passed to the adaptive bitloading by using a fixed constellation of 2 bits per subcarrier as shown in Figure 1 and described in 1.3.

The encoded header block is segmented into *D* symbol frames (*D*= 1, 2). The value of *D* is selected by the coordinator and indicated in the PHY header. The first *kH* bits of the first encoded header block is mapped into the first symbol frame, so that *b*0 is transmitted first. If *D*= 2, the first *kH* bits of the second encoded header block is mapped into the second symbol frame, so that *bNFEC*/2 is transmitted first. The rest of the bits of the first and the second encoded header blocks are discarded. Header segmentation is illustrated in Figure 10.



**Figure 10 Header segmentation**

* + - 1. **Spatial precoder for the header**

The spatial precoder is the same as for the payload, see 1.6.4.2.

* + 1. **PHY payload encoding**

The payload encoder is shown in Figure 11. It contains an FEC encoder and a payload repetition encoder (PRE) to support robust communication.



**Figure 11 Payload encoder**

The incoming PHY-frame payload is divided into sequential blocks of information bits, *K* bits per block. Each block of information bits is encoded by the FEC, as described in 1.2. The valid values of *K*, the coded block size *NFEC*, and the coding rate *R*, are given in Table 3. The bits of each information block are in the same order as they are in the payload; the payload bit to be transmitted first is the first in the corresponding information block. In normal mode, indicated by REP = 001 in the PHY header, PRE is disabled. The FEC codewords are passed directly to the output of the payload encoder and concatenated into the encoded payload block; their order are the same as the order of corresponding information blocks at the input of the payload encoder. For robust communication, each FEC codeword is further encoded by the PRE where the FEC codewords are repeated and concatenated into the encoded payload block.



**Figure 12 Mapping of a FEC codeword onto sections**

* + - 1. **Payload repetition encoder**

The Payload repetition encoder (PRE) performs the number of repetitions *NREP* specified in Table 7.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **REP value** (*b*7*b*6*b*5) | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| **Interpretation** (NREP) | reserved | 1 | 2 | 3 | 4 | 6 | 8 | reserved |

**Table 7 Allowed values of REP**

The used number of repetitions is advertised in the REP field in the PHY header. PRE operates as follows. Each incoming FEC codeword is first copied *NREP* times. Each copy is divided into *S* sections, numbered from 0 to *S–*1, with *B* bits in each section. Bits of the FEC codeword are mapped into sections in ascending sequential order; the bit of the FEC codeword to be transmitted first shall be the first bit (*b*0) of Section 0. If, after all bits of the FEC codeword are mapped, the last *q* bit positions of the last section remain empty, these positions shall be filled by the first *q* bits of Section 0 in ascending sequential order. Mapping of an FEC codeword on to sections is shown in Figure 12.



**Figure 13 Format of the encoded payload block (payload consists of J FEC codewords)**

If *floor*(*kP/NREP*) is divisible by 4, the number of bits per section is set to *B* = *floor*(*kP/NREP*)-1; otherwise, *B* = *floor*(*kP/NREP*), where *kP* is the total number of bits that can be loaded on to the payload OFDM symbol according to the current BAT. The number of sections per FEC codeword is *S = ceil* (*NFEC*/*B*). If the computed value of *S* is 1, *H* consecutive FEC codewords may be concatenated. The number of sections in this case is *S = ceil*(*H*×*NFEC/B*), where *H* is selected to provide *S* > 1 for the given values of *NFEC, N*REP and *kP*. Concatenation of codewords may only be applied when an FEC information block size is 960. The total size of concatenated codewords cannot exceed the maximum FEC codeword size. PRE parameters NREP and *H* are so that q < *H* x NFEC.

If the number of FEC codewords in the payload is not a multiple of *H*, the necessary *z* < *H* dummy FEC codewords are added. Dummy codewords are copies of the last FEC codeword of the same payload. The values of *H* (1, 2 and 4) and *z* (0 to *H*-1) are indicated in the FEC concatenation factor (FCF) field of the PHY header (see Table 8).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |
| --- |
| **FCF value** (*b*2*b*1*b*0) |

 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| H | 1 | reserved | 2 | 2 | 4 | 4 | 4 | 4 |
| z | 0 | reserved | 0 | 1 | 0 | 1 | 2 | 3 |

**Table 4 FEC concatenation factor (FCF)**

The PRE outputs sections sequentially, in groups of *S* sections. Each group carries a copy of the FEC codeword. The number of groups per each FEC codeword is *NREP*. The order of bits in each section is the same as these bits appear in the incoming FEC codeword. The format of the encoded payload block with PRE is shown in Figure 13. The total number of sections in the encoded payload block is *NREP*×*S*.

The order of sections in the first group is ascending, from 0 to *S*–1; the order of sections in all subsequent groups is cyclically shifted. The shift is defined by the cyclic section shift (CSS) vector {0 CSS2 CSS3 … CSS*NREP*} with a length of *NREP*, where CSS*L* is the sequential number of the section to be transmitted first in the *L*th group of sections. The value of CSS shall be computed using the following rule:

*NREP*=2: if (*S* mod 2)=0 CSS:={0,1} else CSS:={0,0}

*NREP*=3: if (*S* mod 3) = 0 CSS:= {0,1,2} else CSS:= {0,0,0}

*N*REP=4: if (*S* mod 4)=0 CSS:= {0,1,2,3} else if (*S* mod 2)=0 CSS:={0,0,1,1} else CSS:={0,0,0,0}

*NREP*=6: if (*S* mod 6)= 0 CSS:={0,1,2,3,4,5} else if (*S* mod 3)=0 CSS:={0,0,1,1,2,2} else if (*S*mod 2)=0 CSS:={0,0,0,1,1,1} else CSS:={0,0,0,0,0,0}

*NREP*=8: if (*S* mod 8)=0 CSS:={0,1,2,3,4,5,6,7}else if (*S* mod 4)=0 CSS:= {0,0,1,1,2,2,3,3} else if (*S*mod 2)=0 CSS:={0,0,0,0,1,1,1,1}else CSS:= {0,0,0,0,0,0,0,0}

As an example, with CSS = 3L for a group of *S*=4 sections, sections will be transmitted in the following order: 3, 0, 1, 2. The first group of sections, for comparison, is transmitted: 0, 1, 2, 3.

* + - 1. **Spatial Precoder for the Payload**

The spatial precoder is a matrix-vector operation ***P****·****x*** operating subcarrier-wise when using MIMO RS.

**If FT=0** (probe frame), the transmitter multiplies the 1x1 scalar stream of header symbols ***x*** with the *N*RSx1 vector ***P*** which contains all ones, i.e. the coordinator sends the same information to all selected transmitters (global transmission). All transmitters send in a synchronous manner. How to realize synchronization of multiple OWC transmitters is out of scope for this standard.

**If FT=1** (transport frame), the transmitter multiplies the 1x1 stream of header information symbols ***x***with the NRSx1 precoding vector ***P*** which contains ones for all active transmitters in a coordinated transmission cluster and zeros elsewhere, i.e. the coordinator sends the same information to all selected transmitters in the same cluster (regional transmission). All transmitters send in a synchronous manner. How to realize synchronization of multiple distributed OWC transmitters is out of scope for this standard.

1. Significant parts of the HB PHY text are taken over from ITU-T recommendation G. 9660/G.9964-2011 and -2015 in the coax baseband (CB) mode. Corresponding simplifications have been taken into account in this specification. [↑](#footnote-ref-1)
2. k=1,2,3,…7 [↑](#footnote-ref-2)
3. For OCR≤200 MHz, the lowest carrier indexes (0, 10) are all unused. [↑](#footnote-ref-3)
4. Parameters are under discussion in ITU-T G.hn2. They will be added as soon as consented. [↑](#footnote-ref-4)
5. Initial zeros are dropped when converting hex to binary numbers. [↑](#footnote-ref-5)
6. Initial zeros are dropped when converting hex to binary numbers. [↑](#footnote-ref-6)