The PSK2k V5 Codes

Klaus von der Heide, DJ5HG

PSK2k is an advanced specialized digital mode for amateur radio meteor scatter communication. This paper is a documentation of it's technical details.

1. The Modulation

PSK2k uses Binary Phase Shift Keying (BPSK) at a rate of 2000 bits/s. The digital-to-analog conversion of the bitsequence is done by replacement of every bit by a pulse-function. The pulse-function is sampled at a standard sampling rate.The pulse function used in PSK2k is a sinc function. It is shown in Figure 1. Figure 2 explains how the PSK2k-signal is generated from a given bitsequence. All information packets of PSK2k have 258 bits. Since the sinc-function is infinitely long, it is smoothely limited. Nevertheless, the generated signal has long wings outside the 258 bits. But these bits are repeated for a period. Therefore the left wing is added at the right end of the sequence and the right wing at the left end.

The signal of figure 2b is multiplied with a sine-wave of 193*2000/258=1496.1 Hz. The result is the Double-Side-Band signal shown in Figure 3. This is the audiosignal that is shifted by the SSB-transceiver to the target frequency. It is called a BPSK signal because the phase is 0° where the binary values (the red dots in figure 2) are +1, and 180° where they are -1.

The spectrum of a PSK2k-signal is rectangular, and it extends from 496.1 Hz to 2496.1 Hz with extremely sharp edges. Stations running PSK2k therefore can be spaced by 2.0 kHz without causing interference.

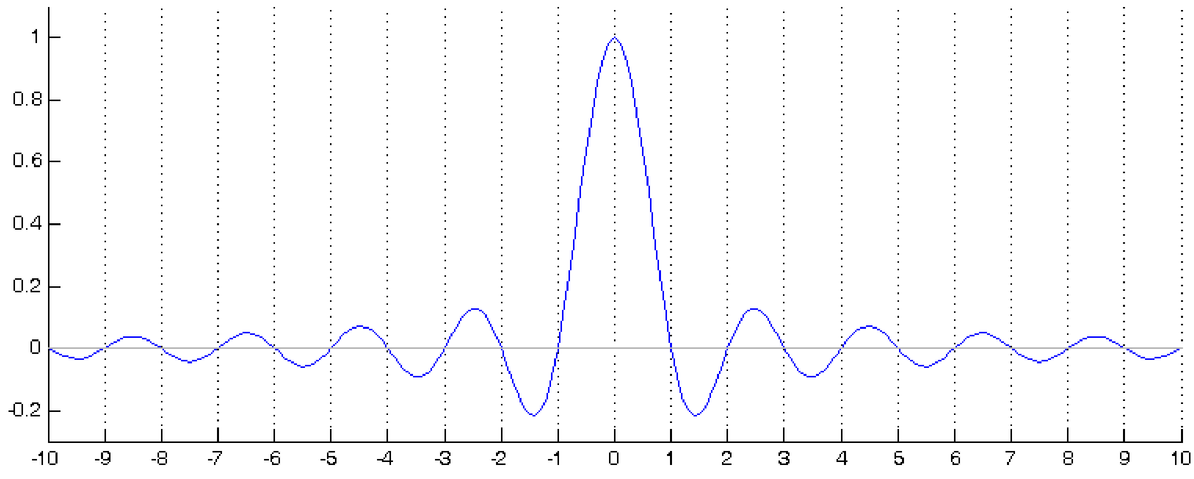
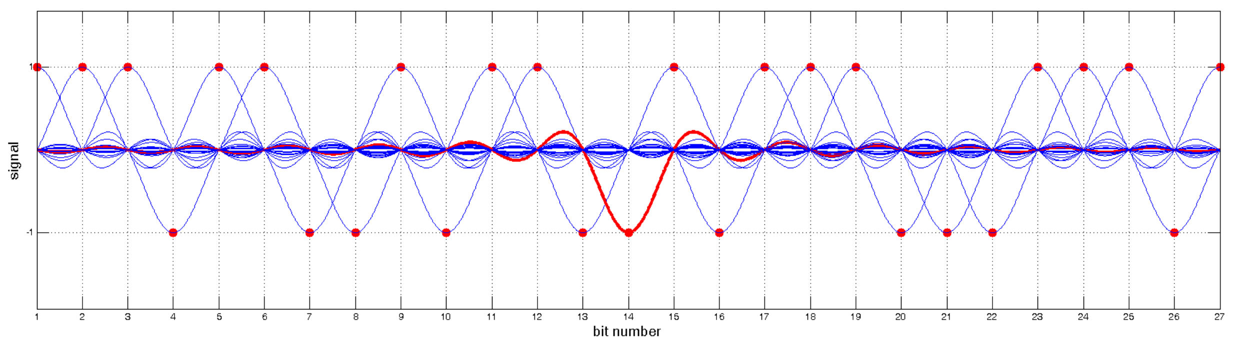
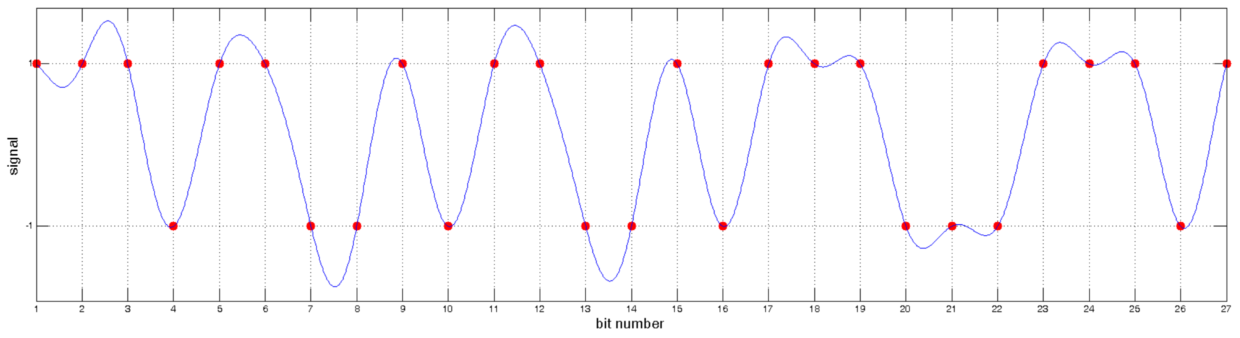


Figure 1. The sinc-pulse. It is infinitely long to both sides with slowly decreasing amplitude. It has zeros at k = ±1, ±2, ±3, ...



a.



b.

Figure 2. Generation of the PSK2k baseband signal. The example bits [1 1 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 1 0 1] are replaced by the values ±1. These values `v` are replaced by `v` times the sinc-function. All these functions are shown as blue lines in figure 2a except for one which is colored in red for demonstration. It is obvious that all other functions have a zero where one function has ist value `1` or `-1`. The sum of all sinc-functions in figure 2b therefore goes through all the values v which are marked as red dots.

The horizontal spacing of the bits (1.0 in the figure) corresponds to 0.5 ms because of the bitrate of 2000 bits/s. The PSK2k-signal is an analog signal. But the PSK2k program generates it at a user-chosen sampling rate. If the sampling rate is 32000 for example, the spacing is 16 samples.

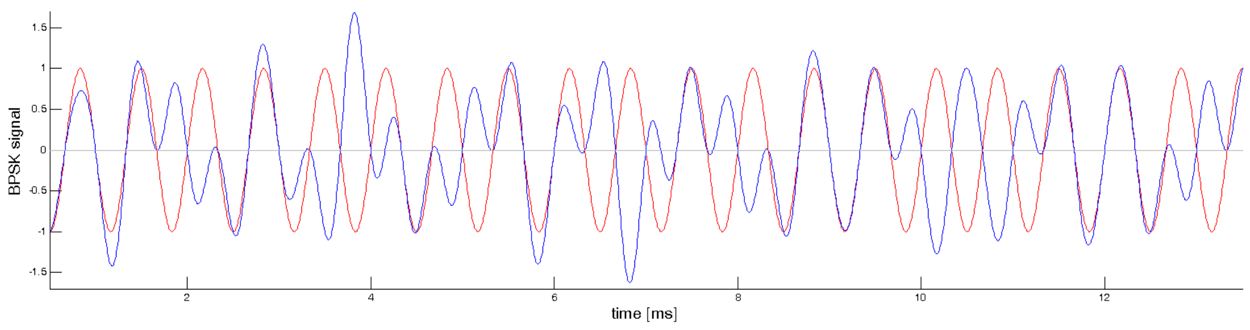


Figure 3. The BPSK-signal (blue) for the same binary data as in figure 2, and the 1496.1 Hz carrier (red).

2. The General Binary Packet Format

All PSK2k-packets contain 258 bits. 43 bits are used for synchronization followed by 49 address bits and 166 bits of encoded information. All bits are interleaved.

3. The Synchronization Pattern

PSK2k uses a Hadamard code of 43 bits for synchronization:

```
0100101001110111110001011100000100011010110
```

This pattern is interleaved with the address and data bits such that every sixth bit of a packet is a synchronization bit. Hadamard codes have optimal autocorrelation features if they are repeated. Figure 4 shows the autocorrelation of the HD43 code. The SNR of PSK2k-packets must be larger than -3dB to be decodable. As figure 4 shows on it's right side, the synchronization is perfect even at very much lower SNR. This guarantees synchronization of very short pings.

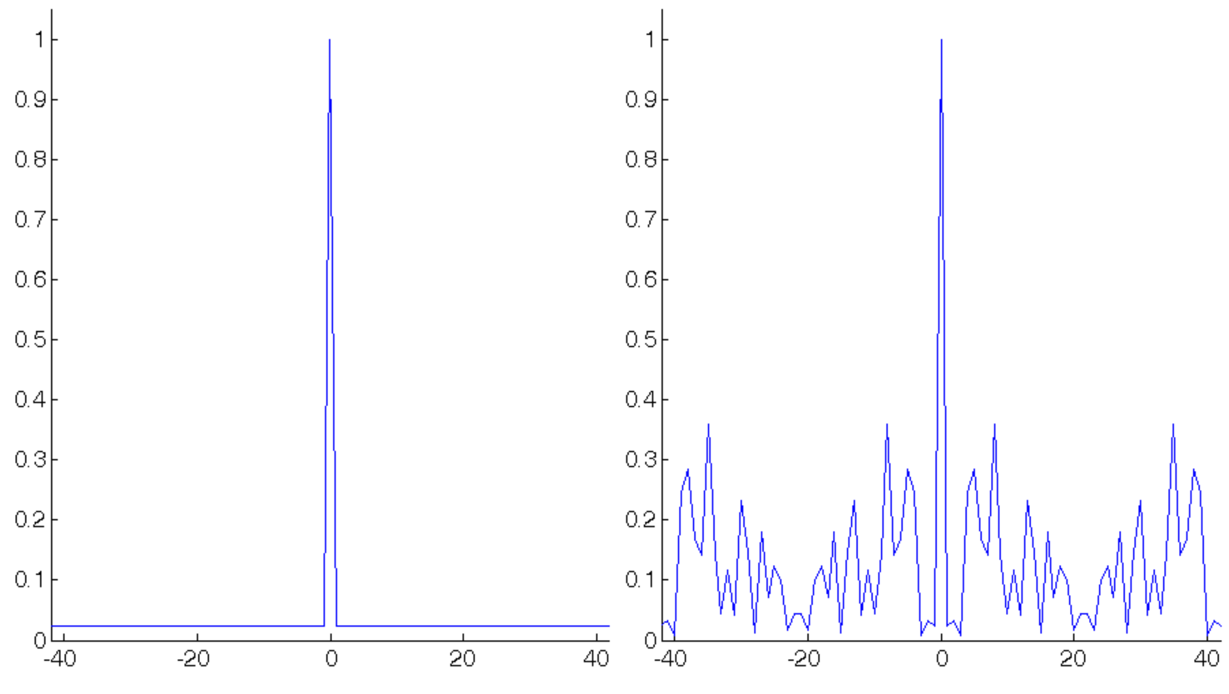


Figure 4. Autocorrelation of the Hadamard-43 code (this is the output of a digital filter which uses the reverse HD-43 pattern as it's filter coefficients with -1 and +1 instead of 0 and 1).

Left: noisefree, right: SNR = –10 dB.

4. The Address Codes

PSK2k uses two different types of addresses. Both are 49 bits long. They are discussed in the following.

4.1. The General Address

The general address is used for all transmissions addressed to everybody, i.e. CQ calls, QRZ calls, and QSTs. The general address has the binary pattern

`1101010000001111100110011011011011011011011011011`

This pattern corresponds to the private address of the non-existing callsign **`8305636 ?6`**. It's cross-correlation with the synchronization pattern is very low.

4.2. The Private Address

The private address is used to address a target callsign. It is unique for every callsign. The address for DJ5HG for example is

`0000111101011001010101101010011100100011101010101`

The encoding of the private address is specified in Chapters 7.1. and 7.2.

5. The Error Correcting Codes

Two different sizes of the source information necessitate the use of two different Convolutional Codes:

5.1. Format 1 Code for Transmission of Callsigns and Station Information

This is a rate `1/2` code with constraint length `c=13`. It is tail-ended with a tail of `2*12` bits. So `166/2-c-1=71` information bits are encoded into a codeword of `166` bits. The polynomials are

`1101101010001`

`1000110111111`

The length of the polynomials is called the constraint length `c` (one bit of the information array influences `c` consecutive bits in the encoded output).

The encoding process of binary convolutional codes works as follows:

Do for each of the polynomials:

(1) All ones of the information bit sequence are replaced by the polynomial bit array.

```
information bits:   010001100000000010100110000010000000000...
replacements:        1101101010001  1101101010001
                         11011010100011101101010001
```

```
1101101010001  1101101010001
                1101101010001
                      1101101010001
```

(2) Count the ones in each column of the replacements. If it is an even number the result is a `0` and a `1` otherwise.

Result of the example: `0110100011111010100010011001100111001000 1`

The encoding of `71` information bits generates `71+c-1=83` bits for each polynomial.

5.2. Format 1 Data Interleaving

The two bit-arrays of the encoding (2*83) bits, and the 43 synchronization bits plus the 49 address bits are arranged in the following sequence. The bits are counted from left to right individually in the four arrays starting at 1. Synchronization bits are colored red, the address is colored magenta, bits from the first polynomial are colored green, and the bits from the second polynomial are colored cyan. The bits are sent row-wise in the following table:

1	1	1	19	36	54	2	15	2	33	50	68	3	29	3	47	64	82	4	43
4	61	78	12	5	57	5	75	9	26	6	71	6	5	23	40	7	2	7	19
37	54	8	16	8	33	51	68	9	30	9	47	65	82	10	44	10	61	79	13
11	58	11	75	9	27	12	72	12	6	23	41	13	2	13	20	37	55	14	16
14	34	51	69	15	30	15	48	65	83	16	44	16	62	79	13	17	58	17	76
10	27	18	72	18	6	24	41	19	3	19	20	38	55	20	17	20	34	52	69
21	31	21	48	66	83	22	45	22	62	80	14	23	59	23	76	10	28	24	73
24	7	24	42	25	3	25	21	38	56	26	17	26	35	52	70	27	31	27	49
66	14	28	45	28	63	80	28	29	59	29	77	11	42	30	73	30	7	25	56
31	4	31	21	39	70	32	18	32	35	53	1	33	32	33	49	67	15	34	46
34	63	81	29	35	60	35	77	11	43	36	74	36	8	25	57	37	4	37	22
39	71	38	18	38	36	53	44	39	32	39	50	67	45	40	46	40	64	81	46
41	60	41	78	12	47	42	74	42	8	26	48	43	5	43	22	40	49		

The reason for the interleaving is to spread the codeword bits that influence the decision upon an information bit. For example, the first information bit is encoded into the first 13 bits of the convolutional encoding with both polynomials. The decision upon that bit only depends on the codeword (soft) bit, if the corresponding polynomial has a 1 at that place. The two polynomials alltogether have 16 ones. So the decision upon an information bit depends on 16 soft bits. The following table marks by grey background where these 16 softbits bits are located, which determine the decision upon the first information bit.

```
 1  1  1 19 36 54  2 15  2 33 50 68  3 29  3 47 64 82  4 43
 4 61 78 12  5 57  5 75  9 26  6 71  6  5 23 40  7  2  7 19
37 54  8 16  8 33 51 68  9 30  9 47 65 82 10 44 10 61 79 13
11 58 11 75  9 27 12 72 12  6 23 41 13  2 13 20 37 55 14 16
14 34 51 69 15 30 15 48 65 83 16 44 16 62 79 13 17 58 17 76
10 27 18 72 18  6 24 41 19  3 19 20 38 55 20 17 20 34 52 69
21 31 21 48 66 83 22 45 22 62 80 14 23 59 23 76 10 28 24 73
24  7 24 42 25  3 25 21 38 56 26 17 26 35 52 70 27 31 27 49
66 14 28 45 28 63 80 28 29 59 29 77 11 42 30 73 30  7 25 56
31  4 31 21 39 70 32 18 32 35 53  1 33 32 33 49 67 15 34 46
34 63 81 29 35 60 35 77 11 43 36 74 36  8 25 57 37  4 37 22
39 71 38 18 38 36 53 44 39 32 39 50 67 45 40 46 40 64 81 46
41 60 41 78 12 47 42 74 42  8 26 48 43  5 43 22 40 49
```

5.3. Format 2 Code for Transmission of Short Messages like R 6dB, RRR, 73

This is a rate 1/9 code with constraint length 10. It is tail-byting. So 162/9=18 bits are encoded into a codeword of 162 bits. The polynomials are

```
1111001001 , 1010111101 , 1101100111
1101010111 , 1111001001 , 1010111101
1101100111 , 1110111001 , 1010011011
```

The covolutional encoding works for each of the 9 polynomials as described in the previous chapter. The only difference is tail-byting. This means that there are as many encoded bits for each polynomial as there are information bits. The tail is added to the beginning:

(1) All ones of the information bit sequence are replaced by the polynomial bit array.

```
18 information bits:   010001100000100010
replacements:           1111001001
                            1111001001
                             1111001001
                                   111100 tail wrapped around
                       1001
                                       11 tail wrapped around
                       11001001
```

(2) Count the ones in each column of the replacements. If it is an even number the result is a `0` and a `1` otherwise.

Result of the example: `001001000111010011`

The encoding of `n=18` information bits generates `n=18` bits for each polynomial.

5.4. Format 2 Data Interleaving

The `9` bit-arrays of the encoding and the 43 synchronization bits plus the 49 address bits are arranged in the following sequence. The bits are counted from left to right individually in the `11` arrays starting at `1`. Synchronization bits are colored red and address bits are colored magenta. The bits from the `9` polynomials are individually colored in the `9` colors

🟩🟨🟦🟧🟪🟩⬜🟪🟦

The bits are sent row-wise in the following table:

1	1	1	13	8	11	2	10	2	5	17	3	3	2	3	14	9	12	4	11
4	6	18	4	5	3	5	15	1	13	6	12	6	7	10	5	7	4	7	16
2	14	8	13	8	8	11	6	9	5	9	17	3	15	10	14	10	9	12	7
11	6	11	18	4	16	12	15	12	1	13	8	13	7	13	10	5	17	14	16
14	2	14	9	15	8	15	11	6	18	16	17	16	3	15	1	17	9	17	12
7	10	18	18	18	4	16	2	19	1	19	13	8	11	20	10	20	5	17	3
21	2	21	14	9	12	22	11	22	6	18	4	23	3	23	15	1	13	24	12
24	7	10	5	25	4	25	16	2	14	26	13	26	8	11	6	27	5	27	17
3	15	28	14	28	9	12	7	29	6	29	18	4	16	30	15	30	1	13	8
31	7	31	10	5	17	32	16	32	2	14	9	33	8	33	11	6	18	34	17
34	3	15	-	35	9	35	12	7	-	36	18	36	4	16	-	37	1	37	13
8	-	38	10	38	5	17	44	39	2	39	14	9	45	40	11	40	6	18	46
41	3	41	15	1	47	42	12	42	7	10	48	43	4	43	16	2	49		

The reason for the interleaving is to spread the codeword bits that influence the decision upon an information bit. For example, the first information bit is encoded into the first `10` bits of the convolutional encoding of all `9` polynomials. But the decision upon that bit only depends on the codeword (soft) bit, if the corresponding polynomial has a 1 at that place. The 9 polynomials alltogether have 60 ones. So the decision upon an information bit depends on 60 soft bits. The following table marks by grey background where the `60` bits are located in the PSK2k-packet, which determine the decision upon the first information bit.

```
  1  1  1 13  8 11  2 10  2  5 17  3  3  2  3 14  9 12  4 11
  4  6 18  4  5  3  5 15  1 13  6 12  6  7 10  5  7  4  7 16
  2 14  8 13  8  8 11  6  9  5  9 17  3 15 10 14 10  9 12  7
 11  6 11 18  4 16 12 15 12  1 13  8 13  7 13 10  5 17 14 16
 14  2 14  9 15  8 15 11  6 18 16 17 16  3 15  1 17  9 17 12
  7 10 18 18 18  4 16  2 19  1 19 13  8 11 20 10 20  5 17  3
 21  2 21 14  9 12 22 11 22  6 18  4 23  3 23 15  1 13 24 12
 24  7 10  5 25  4 25 16  2 14 26 13 26  8 11  6 27  5 27 17
  3 15 28 14 28  9 12  7 29  6 29 18  4 16 30 15 30  1 13  8
 31  7 31 10  5 17 32 16 32  2 14  9 33  8 33 11  6 18 34 17
 34  3 15  -  35  9 35 12  7  -  36 18 36  4 16  -  37  1 37 13
  8  -  38 10 38  5 17 44 39  2 39 14  9 45 40 11 40  6 18 46
 41  3 41 15  1 47 42 12 42  7 10 48 43  4 43 16  2 49
```

6. Error Detection

6.1. Introduction to Error Detecting Codes

The error correcting capability of codes ist limited. If the signal is more corrupted than the code can correct then the decoding result is worse compared to that of an uncoded transmission. These faulty messages can be recognized by an additional Error Detecting Code. This means that, at a first step, the `k` information bits are encoded by such a code (also called the inner code) into `m>k` bits. And these `m` bits are then, as the second coding step, encoded by an Error Correcting Code (also called the outer code) resulting in `n>m` bits.

Error Detecting Codes usually are systematic codes. A systematic code is one that starts each codeword with the information bits unmodified. The additional bits are called parity bits. These bits are generated from the information bits by a fixed algorithm. The receiver first decodes the outer code. Then it simply generates the parity bits from the decoded information bits and compares them to the decoded parity bits. If there is any difference, the message is discarded.

The capability of error detection also is limited. If only one parity bit is used then on average every second faulty message would be accepted by random. In the case of `n` parity bits the probability of accepting a faulty message is not lower than 2^{-n}.

6.2. Residual Codes

The error detection in PSK2k generally is based on residual codes. This means that the information bit array is interpreted as a natural number `z`. The parity bits are computed as the binary representation of the residual of the division of `z` by a fixed quotient `r`.

Examples with r=31.

binary arrays	0000010001	1000111101	1111010010
z	17	573	978
remainder	17	15	17
parity bits	10001	01111	10001

If r is a prime number then the chance to get acceptable parity bits in case of a received random bit sequence is 1/r .

6.3. Error Detection in Format 1

For CQ, QRZ, QST messages, of stations, and individual messages to calls generally 15 parity bits are generated with r =32749. The source information is encoded into 56 bits.

Calls of a station with or without a report like SM2CEW de DJ5HG 3dB transport a full callsign and the report, which needs 57 bits for the source code. In that case 14 parity bits are generated with r = 16381.

The special Contest Messages use 17 parity bits generated with r =131071.

The source from which the parity bits are generated is changing:

Messages to all stations:	the information bit array
Calls of stations:	the information bit array plus the bitarray of the addressed callsign
Messages in a QSO:	the information bit array plus the bitarrays of the addressed callsign and of the sending station

6.4. Error Detection in Format 2

The short messages for report, roger, and 73 use 15 parity bits generated with r =32749. The parity bits are generated from the information to be sent plus the bitpatterns of both callsigns.

7. The Source Codes

The Error Correcting Code plus the elimination of unwanted false decodes by the Error Detecting Code has the consequence that all the bit arrays discussed in the following Chapters 7.2 and 7.3 can be assumed as error-free, when they are processed in the receiver.

7.1. The Binary Code of Callsigns

The alphabet for callsigns is given by

/	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36

Each character of a callsign to be encoded is replaced by it's index into this alphabet which is below the characters. So an A has to be replaced by 1, a Z by 26, and a / by 0. For example the callsign `DJ5HG` results in the sequence of indices `4,10,32,8,7.` This sequence is interpreted as a number `z` written in base-37 notation instead of the usual base-10 notation. To get the value of the number `z`, we add the last number of the sequence, the second-last times the base, the third-last times the square of the base, plus the fourth-last times the cube of the base and so on. In the example, we get

```
z = 7 + 8*37 + 32*37*37 + 10*37*37*37 + 4*37*37*37*37 = 8047285
```

The callsigns differ in length from 3 up to 10 characters. The length of the binary array is as follows:

length of callsign:	3	4	5	6	7	8	9	10
length of binary pattern:	16	21	27	32	37	42	47	see below
length code	0111	1011	0011	1101	0101	1001	0001	0

The binary representation of `z` with 27 bits is the binary code of DJ5HG:

```
000011110101100101010110101
```

One or more residual codes are appended to this code such that the final representation of a callsign has 54 bits independently of the length of the callsign. The last four (or the last one) bits of the 54-bit code are the length code. The remaining bits between are used for parity bits generated as follows.

7 different prime numbers `r` are used to generate parity bits from `z` by the method discussed in chapter 6.2. These are

prime number `r`	7	13	23	29	31	59	61
number of parity bits	3	4	5	5	5	6	6

Which of these actually are used depends on the length of the callsign:

length of callsign				`r` in use			
3	7	13	23	29	31	59	61
4	7	13		29	31	59	61
5	7	13		29	31		61
6	7	13		29	31		
7	7			29	31		
8	7				31		
9	7						
10	no parity						

Example: In the case of the callsign DJ5HG discussed above we have `z=8047285.` The length of the callsign is `5`. So we have to compute the remainders of the division of `z` by `r=7, 13, 29, 31, 61.` These are `1, 12, 17, 26, 43.` The binary representations of these remainders are

`001 , 1100 , 10001 , 11010 , 101011`. These all are concatenated and inserted between the binary representation of the callsign and the length code. This results in the 54 bit binary code. For clarity, the segments are colored here:

```
000011110101100101010110101001110010001110101010110011
```

------binary representation of DJ5HG------- ----------------parity bits----------------length code

If the length of the callsign is 10 characters then it's binary representation is the concatenation of the binary representations of the first 6 characters (32 bits) and of the last 4 characters (21 bits) plus the length code which is a `0` in this case. The the binary code of OH0M/DJ5HG for example is

```
00111110111110010111110111011011010000110011011110001 0
```

---------representation of OH0M/D--------------- -----representation of J5HG-----length code

7.2. The Private Address

The private address is only 49 bits long. It is generated from the binary code of a callsign (see 7.1) by erasure of 5 bits. If the length code is 0 then bits 50:53 are erased, otherwise bits 47:50. As a consequence, the private address cannot represent more than 9 characters. That is no problem because the full address additionally is encoded in the check bits of any private message. The receiver therefore can well differentiate between messages to the callsigns OH0M/DJ5HG and OH0M/DJ5HH although they effectively have the same private address. The only disadvantage in this case is that both receivers will activate the Viterbi decoder.

Example: The binary address of DJ5HG is (compare with the binary representation in 7.1):

```
000011110101100101010110101 001 1100 10001 11010 10 0011
```

7.3. Messages to All Stations

QST, CQ, and QRZ messages are all built the same way:

The first 54 bits encode a text message TEXT of 10 characters of the alphabet

```
 / A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 . , - ?
 0 1 2 3 4 5 6 7 8 91011121314151617181920212223242526272829303132333435363738394041
```

The binary representation is generated similar to what has been described in chapter 7.1. But the base used is 42, and the representations of the first 5 characters and of the last 5 characters are concatenated. The text is filled up with blanks (index 0) if it is shorter than 10 characters.

Example: encode the text HELLOWORLD. The indices of the characters are

```
8, 5, 12, 12, 15, 23, 15, 18, 12,  4.
```

The numbers are

```
z1=15+12*42+12*42*42+ 5*42*42*42+ 8*42*42*42*42 = 25285695
z2= 4+12*42+18*42*42+15*42*42*42+23*42*42*42*42 = 72712588
```

Their binary representations are

`001100000011101010000111111` and `100010101011000000110001100`

The Format 1 Code is the concatenation of these two binary arrays:

```
001100000011101010000111111100010101011000000110001100
```

Two bits are appended to these 54 bits to define the message type. The type codes are

type code	message
`00`	QST : arbitrary TEXT
`01`	CQ de MyCall (TEXT = MyCall, length of MyCall is 10 characters)
`10`	QRZ de MyCall (TEXT = MyCall)
`11`	CQ de MyCall QTF PERIOD

If the length of the callsign of the calling station is less than 10 characters, a CQ generally uses type code `11`. The QTF and the period are encoded in the last character of the TEXT. The index of the last character is generated from the QTF in degrees and the periodindex (1 : 15s; 2 : 30s; 3 : 2.5s) as follows:

```
characterindex = 3*mod(round(QTF/30),12) + periodindex
```

Example: `CQ de DJ5HG QTF=240° period=15s` is encoded the same way as `CQ de DJ5HG` but the index of the last character of the text to be encoded is now the QTF index `25` and not `0` (the appended blank). And the type code `11` is appended instead of `01` :

The indices of the characters of `DJ5HG` (with 4 trailing blanks) are `4,10,32,8,7,41,41,41,41`. With the QTF and period index `25` as the last of the 10 indices the computation of 7.1.1 now yields

$$z1 = 7 + 8*42 + 32*42^2 + 10*42^3 + 4*42^4 = 13244455$$

$$z2 = 25 + 41*42 + 41*42^2 + 41*42^3 + 41*42^4 = 130691215$$

The binary representations of `z1` and `z2` have to be concatenated:

`000110010100001100000100111111110010100011000010001111`

With the two type bits `11` added we get the final source code of

`CQ de DJ5HG QTF=240° period=15s:`

`00011001010000110000010011111111001010001100001000111111`

For comparison the code of `CQ de DJ5HG:`

`00011001010000110000010011111111001010001100001001111101`

15 parity bits are added following the algorithm of 6.3.

7.4. Addressed Messages

There are several data formats for addressed messages:

7.4.1. Addressed call of a station with or without report

The information to transport is the callsign of the sending station plus a possible report. The callsign of the addressed station is encoded in the private address and additionally in the parity bits of the information.

Example: SM2CEW de DJ5HG 0dB

This is a message addressed to SM2CEW. The callsign of the caller DJ5HG is encoded in the same way as it is done in a CQ call. The message bits are preceeded by three message type bits. They encode the possible reports:

no report	000
0dB	001
3dB	010
6dB	011

The binary information of the message therefore is

0010001100101000001100000100111111110010100011000010011111

This is exactly the same as that of CQ de DJ5HG (see Chapter 7.3.). But the CQ uses the general address instead of the private address of SM2CEW. And, more imortant, the parity bits differ.

The 14 parity bits of the addressed message are generated by the algorithm of Chapter 6.2. But not the encoded information of 57 bits is used to determine the residual. It's concatenation with the callsign of the receiving station is taken, in the example:

0010001100101000001100000100111111110010100011000010011111...

001010000000100100011110011110000001010010010101011101

The value in decimal notation is 356541683245198793421344238544221(which is much more than a million times the number of nanoseconds since the bing bang). The remainder of the division of this huge number by r=16381 is 13615, and it's binary representation in 14 bits is

111010100101111.

The message to be encoded with the outer code is the concatenation of the binary information of the message with these 14 parity bits:

0010001100101000001100000100111111110010100011000010011111111010100101111

7.4.2. Addressed free message of up to 10 characters

The 10 arbitrary characters are converted into 54 bits in the same way as described in Chapter 7.1. The message type bits are 11. The parity bits are generated as described in chapter 6 with r=32749.

Example: The addressed message SM2CEW de DJ5HG : TNX PETER

11011110001100001110100011010001000001000010100100000001

The parity bits are generated as above from this bit array plus the callsign of the receiving station, but plus the callsign of the sending station too:

11011110001100001110100011010001000001000010100100000001...

001010000000100100011110011110000001010010010101011101...

000011110101100101010110101001110010001110101010110011

Or in decimal notation: 20295786689076840300041766904184126719949080685235.

The remainder of the division by r=32749 is 7041 or in binary notation 001101110000001.

The message then is

`110111100011000011101000110100010000010000101001000000001001101110000001`

7.4.1.3. Addressed contest message with report, roger, number, locator, QTF, power, antgain

The bit array starts with the message type bits `10`. The information is encoded into the following 52 bits:

Let be

R	the roger bit (0: no roger; 1: roger)
s	a report index (starting index is `0`). The list of possible reorts is `{ 0dB, 3dB, 6dB}`
i	the index into the list of the number of decodes (starting index is `0`). The list of possible number of decodes is `{ 1, <4, <8, <16, >15 }`
n	the contest QSO-number (0<n<1416)
q	the index into the list of possible QFT-values in degrees (starting index is `0`). The list of possible QTF-values is the QTF in steps of 5 degrees { 0°, 5°, 10°, ... , 355°, 'round horizontally polarized', 'round vertically polarized')
L	an array of 6 indices that defines the Maidenhead locator (starting indices are `0`). The indexed alphabet of the first two indices is `ABCDEFGHIJKLMNOPQR` The indexed alphabet of the last two indices is `ABCDEFGHIJKLMNOPQRSTUVWX` The indexed alphabet of the two indices between is `0123456789`
p	an index which specifies the actual transmitter power (starting index is `0`). The indexed list is `{ QRP, 10W, 25W, 50W, 100W, 250W, 500W, QRO }`
g	an index which specifies the actual antenna gain (starting index is `0`) The indexed list is `{ 0dB, 3dB, 6dB, 9dB, 12dB, 15dB, 18dB, 21dB }`

The information bit array now is assembled:

`bits(1...2)`	`1 0`
`bit(3)`	R
`bits(4...22)`	binary representation of `370*(n-1) + 5*q + i`
`bits(23...48)`	binary representation of the following number:

```
18662400*s + 777600*L(6) + 32400*L(5) + 3240*L(4) + 324*L(3) + 18*L(2) + L(1)
```

`bits(49...51)`	binary representation of p
`bits(52...54)`	binary representation of g

Example: Le the actual data be no roger, report is 3dB, only 1 ping decoded, contest number is 35, QTF is 65°, the locator is JO53IM, transmitting power is 300W, antenna gain is 17dB. The above variables are:

`R=0, s=1, i=1, n=35, q=65/5=13, L=[ 9, 14, 5, 3, 8, 12 ], p=5, g=6.`

With these values we finally get the bit array

`1000000001100010110011001101011110100011111010001101110`

The 17 parity bits are generated as above with `r=131071` from the concatenation of this bit pattern with those of both callsigns of the QSO.

7.4.2. The Short Message

A short message transports only 3 bits of information plus 15 parity bits. These parity bits depend on both callsigns in the contact. They function as a password for the display of the message.

There are 5 different messages. They are listed below with their bitpatterns:

Information	encoding
`R 0dB`	`000`
`R 3dB`	`001`
`R 6dB`	`010`
`RRRR`	`011`
`TNX 73`	`100`

Example: `SM2CEW de DJ5HG RRRR` is encoded by the following bitarray:

`011101001100010010`

The parity bits are the binary representation of the remainder of the division of the binary number

`011001010000000100100011110011110000001010010010101011101...`

`00001111010110010101011010100111001000111010101010110011`

by `r=32749`.

8. The Receiver

8.1. The General Structure

The receiver is the most complex subsystem in the PSK2k-program. It's general structure is shown in figure 5. We will discuss only some aspects here. The first stages of the receiver are a bandpass and a Hilbert filter. The Hilbert filter transforms the real input signal into a complex signal with the real part equal to it's input. The complex signal is shifted down to zero frequency by multiplying with the complex conjugate carrier. The next stages are the matched lowpass filter, the very simple birdie-blanker and the noise blanker. After a usual carrier synchronization, which is described in 8.2, the demodulation follows. The demodulated signal is filtered with the synchronization filter. The largest peaks at the output of the synchronization filter indicate the end of received packets. According to the position of these peaks, the last 258 bits are read from the demoduated signal. They are decoded by the Viterbi decoder. If all received checkbits are identical to those that are generated from the decoded information, then the decoded information is accepted, otherwise it is discarded.

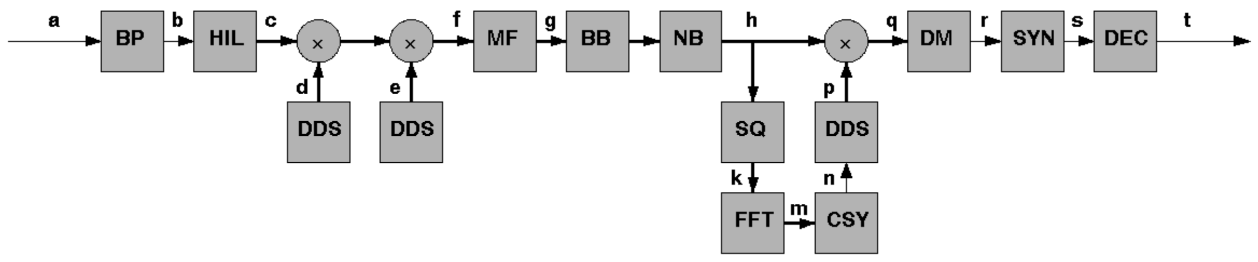


Figure 5. The structure of the PSK2k receiver with the following components:

BP	Input bandpass 200 - 2800 Hz
HIL	Hilbert filter which transforms the real signal into a comples signal
DDS	Direct Digital Synthesis of complex sine waves (helix signals)
MF	Matched filter for the actual sincpulses at a rate of 2000 per second
BB	Birdie blanker which suppresses unwanted carriers
NB	Noise blanker which suppresses unwanted pulse-type noise
SQ	Square of signal
FFT	Fast Fourier Transform
CSY	Carrier syncronization
DM	Demodulator
SYN	Packet synchronization and bit synchronization
DEC	Decoder
a	real input signal (audio output of the SSB-receiver)
b	real signal after input bandpass
c	complex signal
d	carrier helix of -1496 Hz
e	helix of actual frequeny offset
f	complex input signal shifted to mean at 0 Hz
g	complex signal after matched filter
h	complex signal cleaned from birdies and pulses
k	h squared
m	k transformed into frequency domain
n	frequency and phase of largest absolute peak of m
p	n in time domain
q	signal with carrier at 0.000 Hz and phase 0.0
r	real part of q
s	packet of 258 bits
t	decoded text and accompanying information (or empty string if an error is detected)

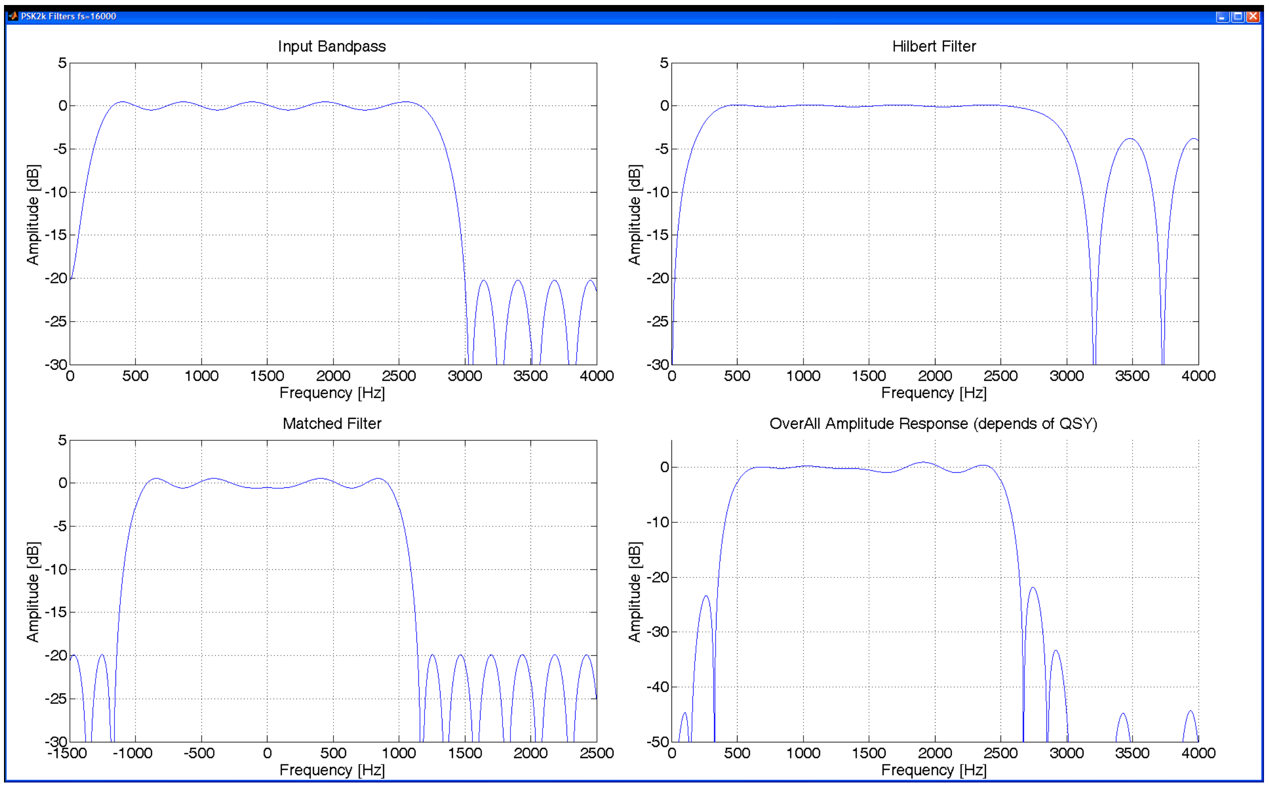


Figure 6. The frequency response of the three filters before the demodulator. The input bandpass has a passband of 300-2800 Hz which allows for a frequency offset of the input signal of up to ±300 Hz. The Hilbert filter generates the imaginary part of the signal. It should be as flat as possible within the frequency range of the signal. The matched filter for a sinc pulse is a lowpass filter with steep edges at halve the bitrate. The response of the complete filter sequence is shown at the lower right.

8.2. Detecting the Carrier Frequency and Demodulation

Let $m(t)$ be the baseband signal shown in figure 4b and let the carrier frequency and the phase be f and φ . f and φ are not precisely known at the receiving end. The received signal - for simplicity without noise here - has the form (amplitude-modulated carrier of frequency f and phase φ).

$$x = m(t) * \exp(i (2 \pi f t + \varphi))$$

The receiver computes the square of all samples:

$$x^2 = m^2(t) * \exp(2 i (2 \pi f t + \varphi)) = m^2(t) * \exp(i (2 \pi (2f) t + 2\varphi))$$

m(t) is a real function. It oscillates in the region -1.3 ... +1.3. The modulated signal x has a suppressed carrier of frequency f. But the square of $m(t)$ is positive. Therefore, the square of x is an amplitude-modulated signal with carrier of frequency $2f$. The detection of the carrier frequency of an amplitude modulated signal is simple. The PSK2k-receiver computes the Fast Fourier Transform (FFT) of the square of x. The maximum peak within this spectrum is at $2f$. Fortunately the FFT also gives 2φ . With these values the receiver then computes

$$x * \exp(-i (2 \pi f t + \varphi)) = m(t) * \exp(i (2 \pi f t + \varphi)) * \exp(-i (2 \pi f t + \varphi)) = m(t)$$

which is the demodulated real baseband signal.

We have neglected a problem here: if we know 2φ, which is in the range of $0 \ldots 2\pi$ we can reconstruct φ only to $\varphi = (2\varphi)/2$ in the region $0 \ldots \pi$. But φ in reality also could be in the region $\pi \ldots 2\pi$ which would lead to 2φ in the region $2\pi \ldots 4\pi$ what is the same as $0 \ldots 2\pi$. The consequence of this ambiguity is that we do not know the sign of the reconstructed $m(t)$, i.e. we finally get a sequence of bits, but we do not know whether we should take them as they are or invert all bits.

But the sign of all bits of the address is known. If the selected maximum correlation value is negative, then the sign is -1, otherwise +1.

8.2. Bit-Synchronization and Packet Synchronization

Let the input sampling rate be 16000 samples/second. Then the length of one bit (0.5ms) is 8 samples. The baseband signal $m(t)$ of one packet then has $258*8 = 2064$ samples. In respect to a simple realization of the packet shift of ±1/3 packet length, the receiver analyses the concatenation of two packets at a time. So the input to the receiver is an array of $2*2064 = 4128$ samples. This array is reshaped to a matrix of size $(4128/8/6) * (8*6) = 86*48$. This matrix is filtered column-wise with three shifted synchronization patterns yielding 8*6=48 filtered signals of length 86.

The reason for using three synchronization patterns at the receiving end -although only one is sent - is the fact that pings often are so short that they do not contain a packet from it's beginning to it's end. But the ping may start somewhere in a packet and end in the next packet. This situation is detected with the shifted synchronization patterns, and the packet bits then are read partly from the first packet and the rest from the subsequent packet.

The three synchronization patterns are:

shift	pattern
0	010010100111011111000101110000010001101011 0
14	001000110101100100101001110111110001011100 0
29	111100010111000001000110101100100101001110 1

For clarity, the three segments are differently colored.

The location of the absolut maximum value of all correlation signals determines which 258 samples have to be taken as the soft bits of the packet. These samples represent the bits of the packet. But they are real values. There is no decision to a binary value at this stage (that is done by the Viterbi decoder). That is the reason for the term 'soft bits'. Once the packet soft bits are found, the 49 soft address bits of this packet are correlated with both, the general address and (if there is a ToCall) the actual private address. If one of these correlations is above a given significance level, the packet is sent to the Viterbi decoder.

8.3. Error Correction, Error Detection, and Source Decoder

A Viterbi decoder decodes the packet. It is this decoder, which makes the decision upon what has been received. The result is a pattern of 71 bits. The first 54 bits now are used to generate 14, 15, or 17 checkbits as discussed in chapter 6. These generated bits are compared to the decoded bits. If there is any difference, the complete packet is discarded.

If the check is successful, the source information is reconstructed from bits 1...54 using the reverse algorithm to that of chapter 7. The result is the textstring of the decoded information. Some information on pingtime, number of decodes, frequency deviation (taken from 8.1), and SNR is appended at the front of the decoded information. The resulting textstring is displayed in the selected decoder window.