The PSK2k 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 216 bits. Since the sinc-function is infinitely long, it is smoothely limited. Nevertheless, the generated signal has long wings outside the 216 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 1500 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 500 Hz to 2500 Hz with extremely sharp edges. Stations running PSK2k therefore can be spaced by 2.0 kHz without causing interference.

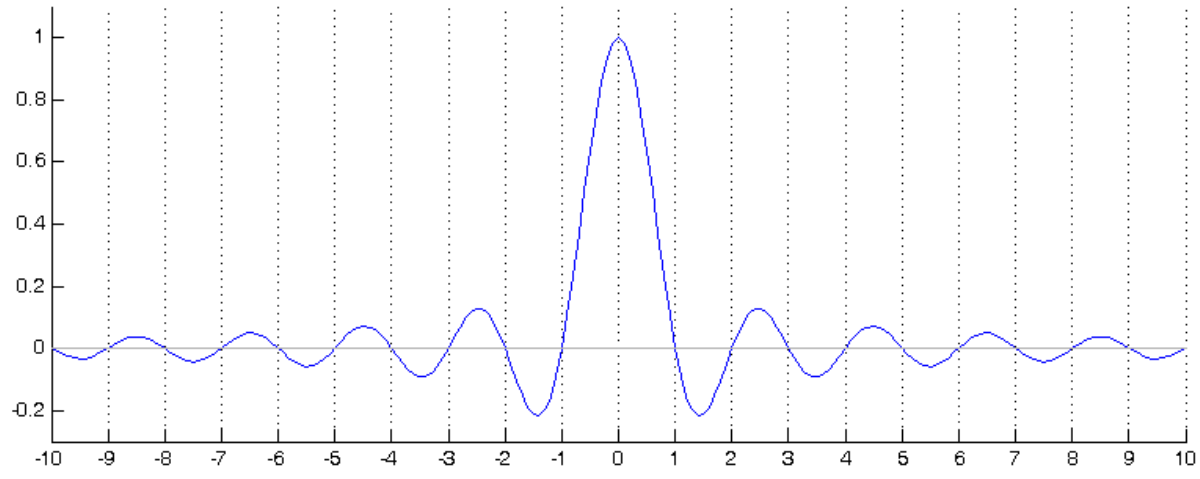
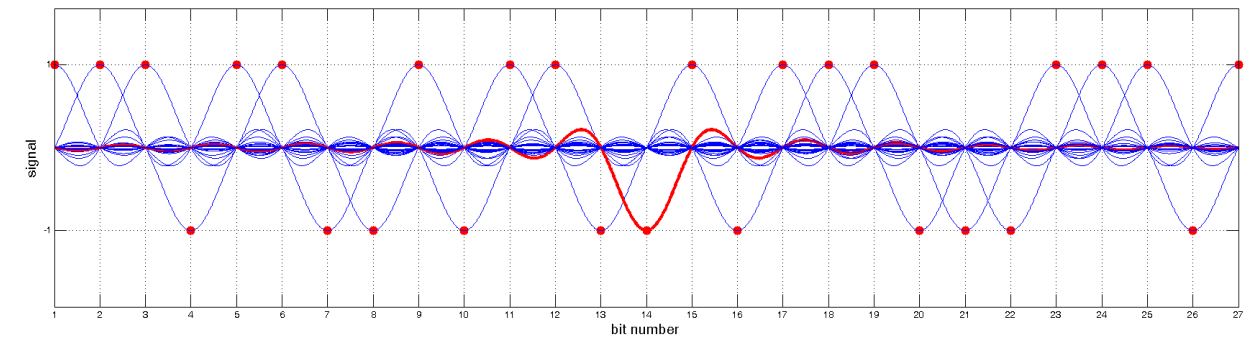
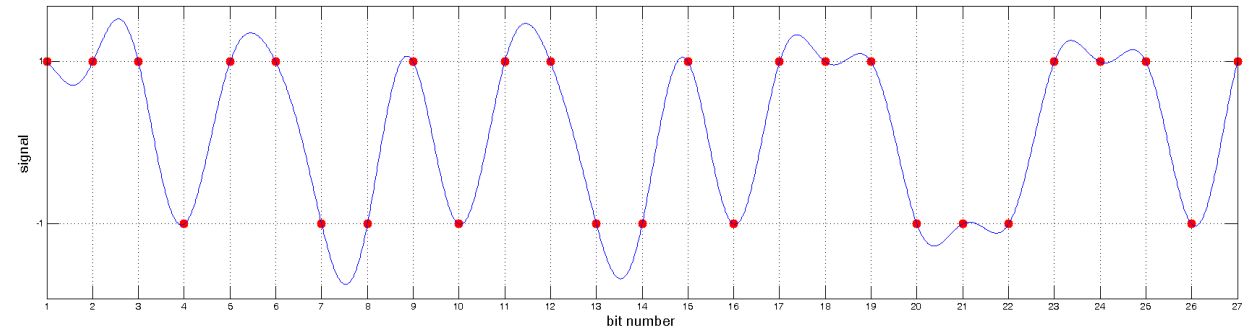


Figure 1. The sinc-pulse. It is infinitely long to both sides with slowly decreasing amplitude. It has zeros at $k = \pm 1, \pm 2, \pm 3, \ldots$



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.

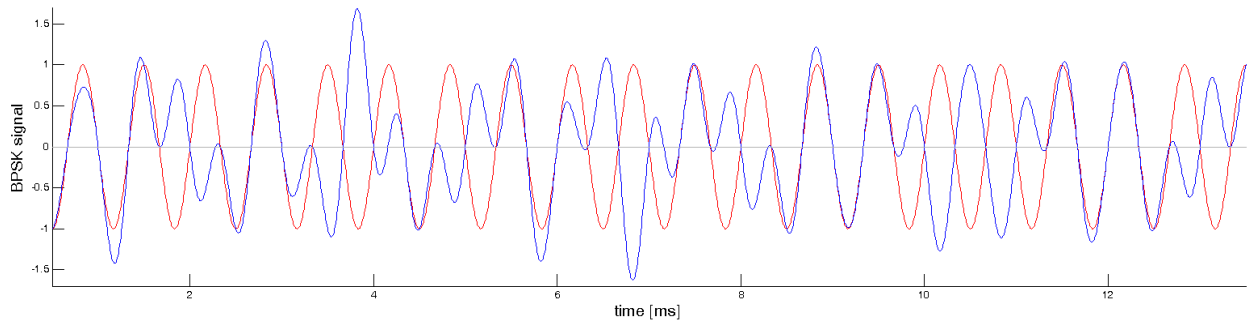


Figure 3. The BPSK-signal (blue) for the same binary data as in figure 2, and the 1500 Hz carrier (red).

2. The General Binary Packet Format

All PSK2k-packets contain 216 bits. 54 bits are header bits used for synchronization, and 162 bits are a codeword. The header bits are interleaved with the codeword: Bits 1,5,9,13,... of the 216 packet bits are the header bits.

3. The Headers

PSK2k uses two different binary header types of 54 bits. They are discussed in the following.

3.1. The General Header

The general header is used for all transmissions addressed to everybody, i.e. CQ calls, QRZ calls, and QSTs. The general header has the unique bitsequence

`001111011110010101000100111011000101011000010001001011`

3.2. The Address Header

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

`000011110101100101010110101001110010001110101010110011`

The encoding of the address header is specified in Chapter 7.1.

4. The Error Correcting Codes

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

4.1. Format 1 Code for Transmission of Callsigns and Stationinformation

This is a rate `1/2` code with constraint length `c=13`. It is tail-ended with a tail of `2*12` bits. So `162/2-c-1=69` information bits are encoded into a codeword of `162` 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: 0100011000000000101001100000100000000000...

replacements:

1101101010001 1101101010001

1101101010001 1101101010001

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: `01101000111110101000100110011001110010001`

The encoding of `n` information bits generates `n+c-1=81` bits for each polynomial.

4.2. Format 1 Data Interleaving

The two bit-arrays of the encoding and the `54` header bits are arranged in the following sequence. The bits are counted from left to right individually in the three arrays starting at 1. Header bits are colored red, 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	19	37	2	55	73	10	3	28	46	64	4	2	20	38	5	56	74	11
6	29	47	65	7	3	21	39	8	57	75	12	9	30	48	66	10	4	22	40
11	58	76	13	12	31	49	67	13	5	23	41	14	59	77	14	15	32	50	68
16	6	24	42	17	60	78	15	18	33	51	69	19	7	25	43	20	61	79	16
21	34	52	70	22	8	26	44	23	62	80	17	24	35	53	71	25	9	27	45
26	63	81	18	27	36	54	72	28	10	28	46	29	64	1	19	30	37	55	73
31	11	29	47	32	65	2	20	33	38	56	74	34	12	30	48	35	66	3	21
36	39	57	75	37	13	31	49	38	67	4	22	39	40	58	76	40	14	32	50
41	68	5	23	42	41	59	77	43	15	33	51	44	69	6	24	45	42	60	78
46	16	34	52	47	70	7	25	48	43	61	79	49	17	35	53	50	71	8	26
51	44	62	80	52	18	36	54	53	72	9	27	54	45	63	81				

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 following table marks by grey background where these 26 codeword bits are located in the PSK2k-packet.

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

4.3. 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.

4.4. Format 2 Data Interleaving

The 9 bit-arrays of the encoding and the 54 header bits are arranged in the following sequence. The bits are counted from left to right individually in the 9 arrays starting at 1. Header bits are colored red. 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	10	2	2	11	3	12	3	4	13	5	4	14	6	15	5	7	16	8
6	17	9	18	7	1	10	2	8	11	3	12	9	4	13	5	10	14	6	15
11	7	16	8	12	17	9	18	13	1	10	2	14	11	3	12	15	4	13	5
16	14	6	15	17	7	16	8	18	17	9	18	19	1	10	2	20	11	3	12
21	4	13	5	22	14	6	15	23	7	16	8	24	17	9	18	25	1	10	2
26	11	3	12	27	4	13	5	28	14	6	15	29	7	16	8	30	17	9	18
31	1	10	2	32	11	3	12	33	4	13	5	34	14	6	15	35	7	16	8
36	17	9	18	37	1	10	2	38	11	3	12	39	4	13	5	40	14	6	15
41	7	16	8	42	17	9	18	43	1	10	2	44	11	3	12	45	4	13	5
46	14	6	15	47	7	16	8	48	17	9	18	49	1	10	2	50	11	3	12
51	4	13	5	52	14	6	15	53	7	16	8	54	17	9	18				

The reason for the interleaving is to spread the codeword bits that inluence 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. The following table marks by grey background where these 90 codeword bits are located in the PSK2k-packet.

1	1	10	2	2	11	3	12	3	4	13	5	4	14	6	15	5	7	16	8
6	17	9	18	7	1	10	2	8	11	3	12	9	4	13	5	10	14	6	15
11	7	16	8	12	17	9	18	13	1	10	2	14	11	3	12	15	4	13	5
16	14	6	15	17	7	16	8	18	17	9	18	19	1	10	2	20	11	3	12
21	4	13	5	22	14	6	15	23	7	16	8	24	17	9	18	25	1	10	2
26	11	3	12	27	4	13	5	28	14	6	15	29	7	16	8	30	17	9	18
31	1	10	2	32	11	3	12	33	4	13	5	34	14	6	15	35	7	16	8
36	17	9	18	37	1	10	2	38	11	3	12	39	4	13	5	40	14	6	15
41	7	16	8	42	17	9	18	43	1	10	2	44	11	3	12	45	4	13	5
46	14	6	15	47	7	16	8	48	17	9	18	49	1	10	2	50	11	3	12
51	4	13	5	52	14	6	15	53	7	16	8	54	17	9	18				

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 13 parity bits are generated with $r=8009$. The source information is encoded into 56 bits.

The special Contest Messages use 15 parity bits generated with $r=32051$.

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=32051. 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 Chapters 7.2 and 7.3 can be assumed as error-free.

7.1. The Address Header Code

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 9101112131415161718192021221324252627282930313233343536
```

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 writen 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	1110	1101	1100	1011	1010	1001	1000	0
length code	0111	1011	0011	1101	0101	1001	0001	0

The binary representation of z with 27 bits is the binary code of DJ5HG in the address header:

```
000011110101100101010110101
```

The 54-bit header code starts with this binary representation 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 address header. For clarity, the segments are colored here:

`000011110101100101010110101001100100001101010101100011`

------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 adress header for OH0M/DJ5HG for example is

`00111110111110010111110111011011010000110011011110001 0`

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

The address header is only used at the receiving end for correlation. It is not of great relevance. The sense of all the encoding only is to get address headers that differ in many bits even if the callsigns are not very different.

7.2. 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 9101112131415161718192021222324252627282930313233343536373839404 1
```

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

00110000001110101000011111 and 10001010101100000011000 1100

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

00110000001110101000011111100010101011000000110001100

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 10characters, 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 : 60s) 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 QRG 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:

00011001010000110000010011111111001010001100001000111 1

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 :

00011001010000110000010011111111001010001100001000111101

7.3. Addressed Messages

There are two data formats for addressed messages:

(1) the normal message

(2) the short message.

7.3.1. The Normal Message

A normal message transports either 52 bits or 56 bits of information plus 17 or 13 parity bits. These messages are encoded similar to the messages to all stations as described in Chapter 7.2. There are a few subformats:

7.3.1.1. 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 header and additionally in the parity bits of the information.

Example: `SM2CEW de DJ5HG 25`

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

`001`00011001010000110000010011111111001010001100001001111 1

This is exactly the same as that of `CQ de DJ5HG` (see Chapter 7.2.). But the `CQ` is not addressed, so it has a different header. And, more imortant, the parity bits differ.

The 12 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:

`001`00011001010000110000010011111111001010001100001001111 1...

00101000000010010001111001111000000101001001010101110 1

The value in decimal notation is `3565416832451987934213442385442 21`(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=4093` is `545`, and it's binary representation in 12 bits is

`001000100001`.

The message to be encoded with the outer code is the concatenation of the binary information of the message with these 12 parity bits:

`001`00011001010000110000010011111111001010001100001001111 1`001000100001`

7.3.1.2. 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 72. The message type bits are 11. The parity bits are generated similar as above in 7.3.1.1, but with r=8009.

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...

00101000000010010001111001111000000101001001010101110 1...

00001111010110010101011010100111001000111010101011 0011

Or in decimal notation: 2029578668907684030004176690418412671994908068523 5.

The remainder of the division by r=8009 is 6882 or in binary notation 1101011100010. The message then is

11011110001100001110100011010001000001000010100100000001 1101011100010

7.3.1.3. 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 { 10W, 25W, 50W, 100W, 250W, 500W, 1kW, 2kW }

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 600W, 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

100000001100010110011001101011110100011111010001101110

The parity bits are generated as above with r=32051 from the concatenation of this bit pattern with those of both callsigns of the QSO.

7.3.2. 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

011001010000001001000111100111100000010100100101010111101...

000011110101100101010110101001110010001110101010110011

by r=32051.

8. The Receiver

This Chapter is still under construction.

The receiver is the most complex subsystem in the PSK2k-program. We will discuss only some aspects here. The first stage of the receiver is a Hilbert filter, which transforms the real input signal to a complex signal. The next stages are the very simple birdie-blanker and the noise blanker. Then the demodulation follows. It is briefly explained in the following chapter.

8.1. 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 unknown 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 very robust. 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 $0 ... 2\pi$ we can reconstruct φ only to $\varphi = (2\varphi)/2$ in the region $0 ... \pi$. But φ in reality also could be in the region $\pi ... 2\pi$ which would lead to 2φ in the region $2\pi ... 4\pi$ what is the same as $0 ... 2\pi$. 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 $216*8 = 1728$ samples. In respect to a simple realization of the packet shift of ±1/3 packet length as shown in figure 8, the receiver analyses the concatenation of two packets at a time. So the input to the receiver is an array of $2*1728 = 3456$ samples. This array is reshaped to a matrix of size $(3456/8/4) * (8*4)$. This matrix is filtered column-wise with all 6 address patterns yielding $12*32=384$ filtered signals of length 108. The absolut maximum value of each signal points to the last packet bit, and the search for the maximum of all these absolut maximum values differentiates between general address and private address and also between the thee possible packet shits, and it determines which samples should be taken for the read-out of the bits.

The number 6 of different address patterns is the product of 2 for the two different addresses, the genneral address and the private address, and 3 for the three possible shifts. For example, the three general addresses rotated in steps of 18 bits are:

shift	pattern
0	001111011110010101000100111011000101011000010001001011
-18	000100111011000101011000010001001011001111011110010101
+18	011000010001001011001111011110010101000100111011000101

The output of the synchronization stage is an array of 216 real values. They are called soft bits, because they represent the received 216 packet bits, but they are corrupted by noise, and there is no decision to their binary values. If the best correlation was with an address pattern shifted by s bits, then the whole packet of 216 bits is rotated by -4s bits to get the correct bit sequence in the codeword.

14.3. 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 69 bits. The first 54 bits now are used to generate 15 checkbits as discussed in chapter 6. These generated bits are compared to the decoded bits 55...69. If there is any difference, the 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.