TSGR1#7(99)e19

TS 25.213 V2.3.0 (1999-9)

Technical Specification

3rd Generation Partnership Project (3GPP);
Technical Specification Group (TSG)
Radio Access Network (RAN);
Working Group 1 (WG1);
Spreading and modulation (FDD)

Reference

<Workitem> (25_213-xxx.PDF)

Keywords

<keyword[, keyword]>

3GPP

Postal address

Office address

Internet

secretariat@3gpp.org

Individual copies of this deliverable
can be downloaded from
http://www.3gpp.org

Contents

Intellectual Property Rights

<IPR notice shall be provided once correct notice is available within 3GPP>

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project, Technical Specification Group Radio Access Network, Working Group 1.

The contents of this TS may be subject to continuing work within the 3GPP and may change following formal TSG approval. Should the TSG modify the contents of this TS, it will be re-released with an identifying change of release date and an increase in version number as follows:

Version m.t.e

where:

m indicates [major version number]

x the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

y the third digit is incremented when editorial only changes have been incorporated into the specification.

1 Scope

The present document describes spreading and modulation for UTRA Physical Layer FDD mode.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

[1] TS 25.201: “Physical layer - general description”

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply.

3.2 Symbols

For the purposes of the present document, the following symbols apply:

<symbol> <Explanation>

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AICH	Acquisition Indicator Channel
AP	Access Preamble
BCH	Broadcast Control Channel
CCPCH	Common Control Physical Channel
CD	Collision Detection
CPCH	Common Packet Channel
DCH	Dedicated Channel
DPCH	Dedicated Physical Channel
DPCCH	Dedicated Physical Control Channel
DPDCH	Dedicated Physical Data Channel
FDD	Frequency Division Duplex
Mcps	Mega Chip Per Second
OVSF	Orthogonal Variable Spreading Factor (codes)
PDSCH	Physical Dedicated Shared Channel
PICH	Page Indication Channel
PRACH	Physical Random Access Channel
RACH	Random Access Channel
SCH	Synchronisation Channel
SF	Spreading Factor
UE	User Equipment

4 Uplink spreading and modulation

4.1 Overview

Spreading is applied to the physical channels. It consists of two operations. The first is the channelization operation, which transforms every data symbol into a number of chips, thus increasing the bandwidth of the signal. The number of chips per data symbol is called the Spreading Factor (SF). The second operation is the scrambling operation, where a scrambling code is applied to the spread signal.

With the channelization, data symbol on so-called I- and Q-branches are independently multiplied with an OVSF code. With the scrambling operation, the resultant signals on the I- and Q-branches are further multiplied by complex-valued scrambling code, where I and Q denote real and imaginary parts, respectively.

4.2 Spreading

4.2.1 Uplink Dedicated Physical Channels (uplink DPDCH/DPCCH)

Figure 1 illustrates the principle of the uplink spreading of DPCCH and DPDCHs. The binary DPCCH and DPDCHs to be spread are represented by real-valued sequences, i.e. the binary value "0" is mapped to the real value +1, while the binary value "1" is mapped to the real value –1. The DPCCH is spread to the chip rate by the channelization code $C_{ch,0}$, while the n:th DPDCH called $DPDCH_n$ is spread to the chip rate by the channelization code $C_{ch,n}$. One DPCCH and up to six parallel DPDCHs can be transmitted simultaneously, i.e. $0 \le n \le 6$.

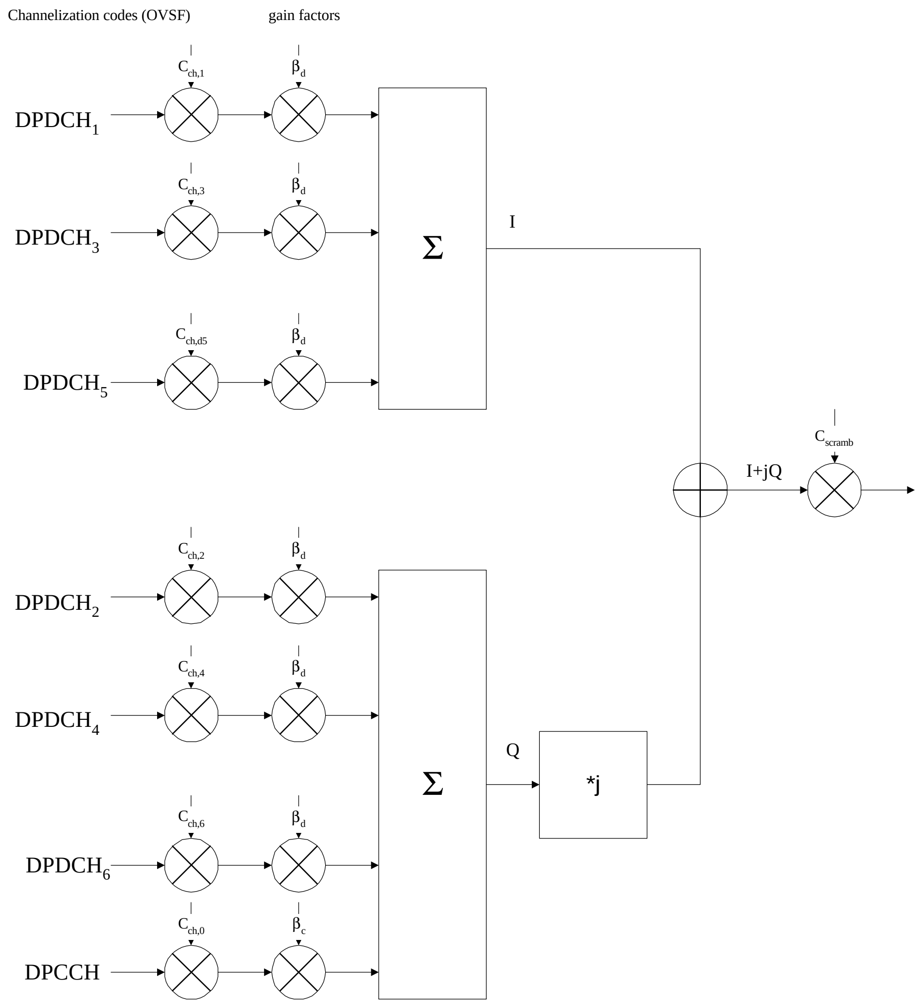

Figure 1. Spreading/modulation for uplink DPCCH and DPDCHs.

After channelization, the real-valued spread signals are weighted by gain factors, β_c for DPCCH and β_d for all DPDCHs.

At every instant in time, at least one of the values β_c and β_d has the amplitude 1.0. The β-values are quantized into 4 bit words. The quantization steps are given in Table 1.

Signalling values for β_c and β_d	Quantized amplitude ratios β_c and β_d
15	1.0
14	0.9333
13	0.8666
12	0.8000
11	0.7333
10	0.6667
9	0.6000
8	0.5333
7	0.4667
6	0.4000
5	0.3333
4	0.2667
3	0.2000
2	0.1333
1	0.0667
0	Switch off

Table 1: The quantization of the gain parameters.

After the weighting, the stream of real-valued chips on the I- and Q-branches are then summed and treated as a complex-valued stream of chips. This complex-valued signal is then scrambled by the complex-valued scrambling code C_{scramb}. After pulse-shaping (described in [1]), QPSK modulation is performed.

4.2.2 PRACH

The spreading and modulation of the message part of the random-access message part is basically the same as for the uplink dedicated physical channels, see section 4.2.1, where the uplink DPDCH and uplink DPCCH are replaced by the data part and the control part respectively. The scrambling code for the message part is chosen based on the preamble code.

4.3 Code generation and allocation

4.3.1 Channelization codes

The channelization codes of Figure 1 are Orthogonal Variable Spreading Factor (OVSF) codes that preserve the orthogonality between a user's different physical channels. The OVSF codes can be defined using the code tree of Figure 2.

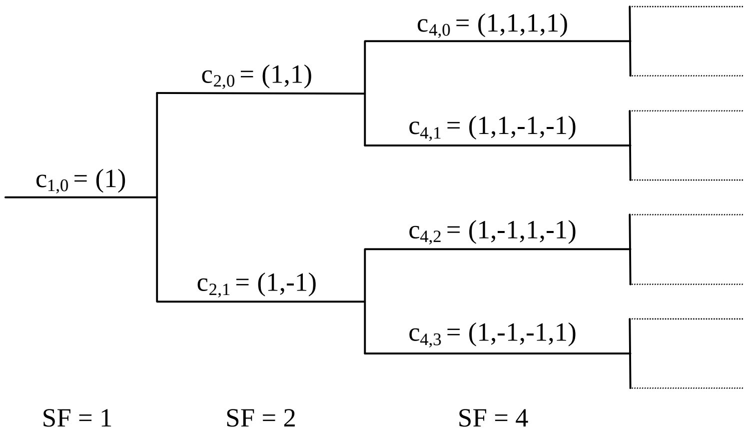

Figure 2. Code-tree for generation of Orthogonal Variable Spreading Factor (OVSF) codes.

In Figure 2, the channelization codes are uniquely described as $C_{SF,k}$, where SF is the spreading factor of the code and k is the code number, $0 \le k \le SF-1$.

Each level in the code tree defines channelization codes of length SF, corresponding to a spreading factor of SF in Figure 2

The generation method for the channelization code is defined as:

$$c_{1,0} = 1,$$

$$\begin{bmatrix} c_{2,0} \\ c_{2,1} \end{bmatrix} = \begin{bmatrix} c_{1,0} & c_{1,0} \\ c_{1,0} & -c_{1,0} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} c_{2^{(n+1)},0} \\ c_{2^{(n+1)},1} \\ c_{2^{(n+1)},2} \\ c_{2^{(n+1)},3} \\ \vdots \\ c_{2^{(n+1)},2^{(n+1)}-2} \\ c_{2^{(n+1)},2^{(n+1)}-1} \end{bmatrix} = \begin{bmatrix} c_{2^n,0} & c_{2^n,0} \\ c_{2^n,0} & -c_{2^n,0} \\ c_{2^n,1} & c_{2^n,1} \\ c_{2^n,1} & -c_{2^n,1} \\ \vdots & \vdots \\ c_{2^n,2^n-1} & c_{2^n,2^n-1} \\ c_{2^n,2^n-1} & -c_{2^n,2^n-1} \end{bmatrix}$$

The leftmost value in each channelization code word corresponds to the chip transmitted first in time.

For the DPCCH and DPDCHs the following applies:

- The DPCCH is always spread by code $c_{256,0}$ i.e. $C_{ch,0} = c_{256,0}$.
- When only one DPDCH is to be transmitted, $DPDCH_1$ is spread by code $C_{ch,1} = c_{SF,k}$ where SF is the spreading factor of $DPDCH_1$ and k= $SF_{d,1}$ / 4
- When more than one DPDCH is to be transmitted, all DPDCHs have spreading factors equal to 4. $DPDCH_n$ is spread by the the code $C_{ch,n} = c_{4,k}$, where $k = 1$ if $n \in \{1, 2\}$, $k = 3$ if $n \in \{3, 4\}$, and $k = 2$ if $n \in \{5, 6\}$.

4.3.2 Scrambling codes

4.3.2.1 General

There are 2^{24} uplink scrambling codes. Either short or long scrambling codes should be used on the uplink. Both short and long scrambling codes are represented with complex-value.

The uplink scrambling generator (either short or long) shall be initialised by a 25 bit value. One bit shall indicate selection of short or long codes (short = 1, long = 0). Twenty four bits shall be loaded into the scrambling generators as shown in sections 4.3.2.2 and 4.3.2.3.

MSB — Initialisation Code Short/Long flag + Value *v* — LSB

short /long	v(23)	v(22)	v(21)	v(20)	v(19)	v(18)	v(17)	v(16)	v(15)	v(14)	v(13)	v(12)	v(11)	v(10)	v(9)	v(8)	v(7)	v(6)	v(5)	v(4)	v(3)	v(2)	v(1)	v(0)

Figure 3 Initialisation Code for Uplink Scrambling generator

Both short and long scrambling codes are formed as follows:

$$C_{scramb} = c_1(w_0 + jc_2' w_1)$$

where w_0 and w_1 are chip rate sequences defined as repetitions of:

$w_0 = \{1 \quad 1\}$

$w_1 = \{1 \quad -1\}$

Also, c_1 is a real chip rate code, and c_2' is a decimated version of the real chip rate code c_2.

With a decimation factor 2, c_2' is given as:

$$c_2'(2k) = c_2'(2k+1) = c_2(2k), \quad k=0,1,2\ldots.$$

The constituent codes c_1 and c_2 are formed differently for the short and long scrambling codes as described in Sections 4.3.2.2 and 4.3.2.3.

4.3.2.2 Long scrambling code

The long scrambling codes are formed as described in Section 4.3.2, where c_1 and c_2 are constructed as the position wise modulo 2 sum of 38400 chip segments of two binary *m*-sequences generated by means of two generator polynomials of degree 25. Let *x*, and *y* be the two *m*-sequences respectively. The *x* sequence is constructed using the primitive (over GF(2)) polynomial $X^{25}+X^3+1$. The *y* sequence is constructed using the polynomial $X^{25}+X^3+X^2+X+1$. The resulting sequences thus constitute segments of a set of Gold sequences.

The code, c_2, used in generating the quadrature component of the complex spreading code is a 16,777,232 chip shifted version of the code, c_1, used in generating the in phase component.

The uplink scrambling code word has a period of one radio frame.

Let $n_{23} \ldots n_0$ be the 24 bit binary representation of the scrambling code number *n* (decimal) with n_0 being the least significant bit. The *x* sequence depends on the chosen scrambling code number *n* and is denoted x_n, in the sequel. Furthermore, let $x_n(i)$ and $y(i)$ denote the *i*:th symbol of the sequence x_n and *y*, respectively

The *m*-sequences x_n and *y* are constructed as:

Initial conditions:

$x_n(0)=n_0$, $x_n(1)= n_1$, … $=x_n(22)= n_{22}$,$x_n(23)= n_{23}$, $x_n(24)=1$

$y(0)=y(1)= … =y(23)= y(24)=1$

Recursive definition of subsequent symbols:

$x_n(i+25) =x_n(i+3) + x_n(i)$ *modulo 2, i=0,…,* 2^{25}-27,

$y(i+25) = y(i+3)+y(i+2) +y(i+1) +y(i)$ *modulo 2, i=0,…,* 2^{25}-27.

The definition of the *n*:th scrambling code word for the in phase and quadrature components follows as (the left most index correspond to the chip scrambled first in each radio frame):

$c_{1,n} = < x_n(0)+y(0), x_n(1)+y(1), …,x_n(N-1)+y(N-1) >$,

$c_{2,n} = < x_n(M)+y(M), x_n(M+1)+y(M+1), …, x_n(M+N-1) + y(M+N-1) >$,

again all sums being modulo 2 additions.

Where N is the period in chips and M = 16,777,232.

These binary code words are converted to real valued sequences by the transformation ‘0’ -> ‘+1’, ‘1’ -> ‘-1’.

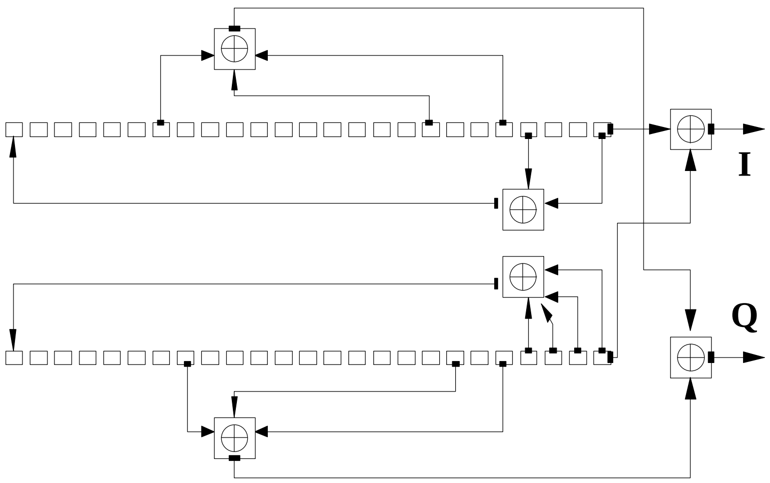

Figure 4. Configuration of uplink scrambling code generator

4.3.2.3 Short scrambling code

The short scrambling codes are formed as described in Section 4.3.2.1,where c1 and c2 are the real and imaginary components of a complex sequence from the family of periodically extended S(2) codes.

The uplink short codes $S_v(n)$, n=0,1,…255, of length 256 chips are obtained by one chip periodic extension of S(2) sequences of length 255. It means that the first chip ($S_v(0)$) and the last chip ($S_v(255)$) of any uplink short scrambling code are the same.

The quaternary S(2) sequence $z_v(n)$, $0 \le v \le 16,777,215$, of length 255 is obtained by modulo 4 addition of three sequences, a quaternary sequence $a_r(n)$ and two binary sequences $b_s(n)$ and $c_t(n)$, according to the following relation:

$z_v(n) = a_r(n) + 2 \cdot b_s(n) + 2 \cdot c_t(n) \quad (mod\ 4)\ , \quad n = 0, 1, …, 254.$

The user index v determines the indexes r, s, and t of the constituent sequences in the following way:

$v = t \cdot 2^{16} + s \cdot 2^{8} + r$,

$r = 0, 1, 2, \ldots, 255$,

$s = 0, 1, 2, \ldots, 255$,

$t = 0, 1, 2, \ldots, 255$.

The quaternary sequence $a_r(n)$ is generated by the recursive generator G_0 defined by the polynomial

$g_0(x)= x^8+x^5+3x^3+x^2+2x+1$ as

$a_r(n)= 3.a_r(n-3) +1.a_r(n-5) + 3.a_r(n-6) +2.a_r(n-7) + 3.a_r(n-8)$ (mod 4).

$n = 8\ldots254$.

The binary sequence $b_s(n)$ is generated by the recursive generator G_1 defined by the polynomial

$g_1(x)= x^8+x^7+x^5+x+1$ as

$b_s(n)= b_s(n-1)+ b_s(n-3)+ b_s(n-7)+ b_s(n-8)$ $(mod\ 2)$.

The binary sequence $c_t(n)$ is generated by the recursive generator G_2 defined by the polynomial

$g_2(x)= x^8+x^7+x^5+x^4+1$ as

$c_t(n)= c_t(n-1)+ c_t(n-3)+ c_t(n-4)+ c_t(n-8)$ $(mod\ 2)$.

An implementation of the short scrambling code generator is shown in Figure 5. The initial states for the binary generators G_1 and G_2 are the two 8-bit words representing the indexes s and t in the 24-bit binary representation of the user index v, as it is shown in Figure 6.

The initial state for the quaternary generator G_0 is according to Figure 6 obtained after the transformation of 8-bit word representing the index r. This transformation is given by

$a_r(0) = 2v(0)+1$ $(mod\ 4)$, $a_r(n) = 2v(n)$ $(mod\ 4)$, $n = 1,\ldots,7$.

The complex quadriphase sequence $S_v(n)$ is obtained from quaternary sequence $z_v(n)$ by the mapping function given in Table 2.

The Re{Sv(n)} and Im{Sv(n)} of the S(2) code are the pair of two binary sequences corresponding to input binary sequences c_1 and c_2 respectively described in 4.3.2.

zv(n)	Sv(n)
0	+1 + j1

1	-1 + j1
2	-1 - j1
3	+1 - j1

Table 2. Mapping between $S_v(n)$ and $z_v(n)$

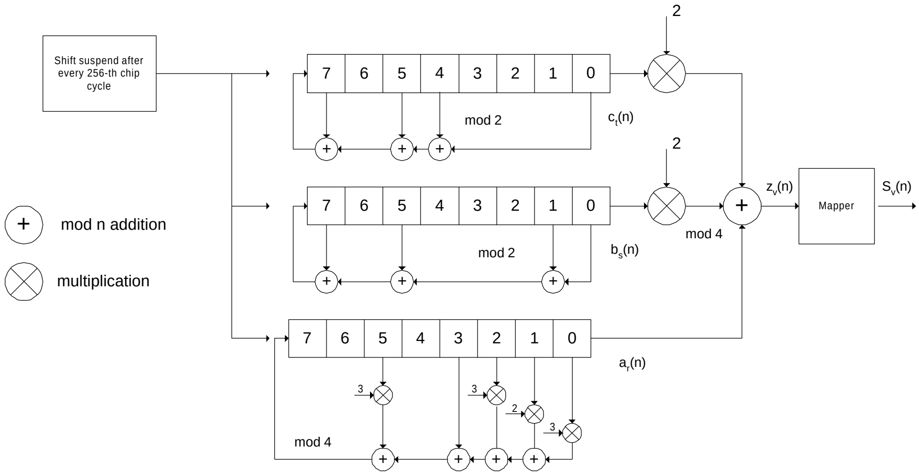

Figure 5. Uplink short scrambling code generator

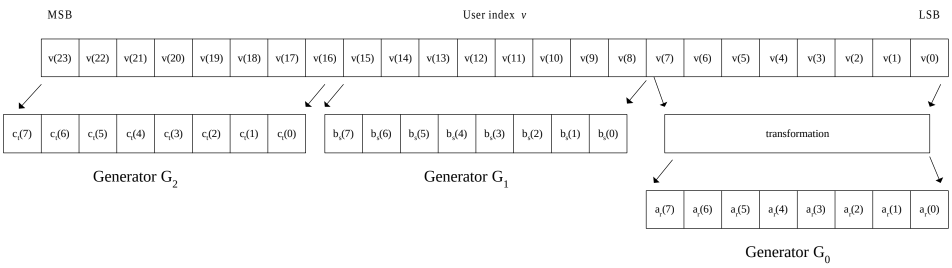

Figure 6. Uplink short scrambling code generator state initialisation

The short scrambling code may be changed during a connection.

4.3.3 Random access codes

4.3.3.1 Preamble scrambling code

The scrambling code for the preamble part is as follows.

The code generating method is the same as for the real part of the long codes on dedicated channels. Only the first 4096 chips of the code are used for preamble spreading with the chip rate of 3.84 Mchip/s. The long code c1 for the in-phase component is used directly on both in phase and quadrature branches without offset between branches. The preamble scrambling code is defined as the position wise modulo 2 sum of 4096 chips segments of two binary m-sequences generated by means of two generator polynomials of degree 25. Let x and y be the two m-sequences respectively. The x sequence is constructed using the primitive (over GF(2)) polynomial $X^{25}+X^3+1$. The y sequence is constructed using the polynomial $X^{25}+X^3+X^2+X+1$. The resulting sequences thus constitute segments of a set of Gold sequences.

Let $n_{23} \dots n_0$ be the binary representation of the code number n (decimal) with n_0 being the least significant bit. Code numbers between 0 and 255 are used for the random access channel. The *m*-sequences x_n and y are constructed as:

Initial conditions:

$x_n(0)=n_0$, $x_n(1)= n_1$, … $=x_n(22)= n_{22}$, $x_n(23)= n_{23}$, $x_n(24)=1$

$y(0)=y(1)= … =y(23)= y(24)=1$

Recursive definition of subsequent symbols:

$x_n(i+25) =x_n(i+3) + x_n(i)$ *modulo 2, i=0,…,* 4070*,*

$y(i+25) = y(i+3)+y(i+2) +y(i+1) +y(i)$ *modulo 2, i=0,…,* 4070*.*

The definition of the *n*:th code word follows (the left most index correspond to the chip transmitted first in each slot):

$C_{RACH,n} = \langle x_n(0)+y(0), x_n(1)+y(1), …,x_n(4095)+y(4095) \rangle$,

All sums of symbols are taken modulo 2.

The preamble spreading code is described in Figure 7.

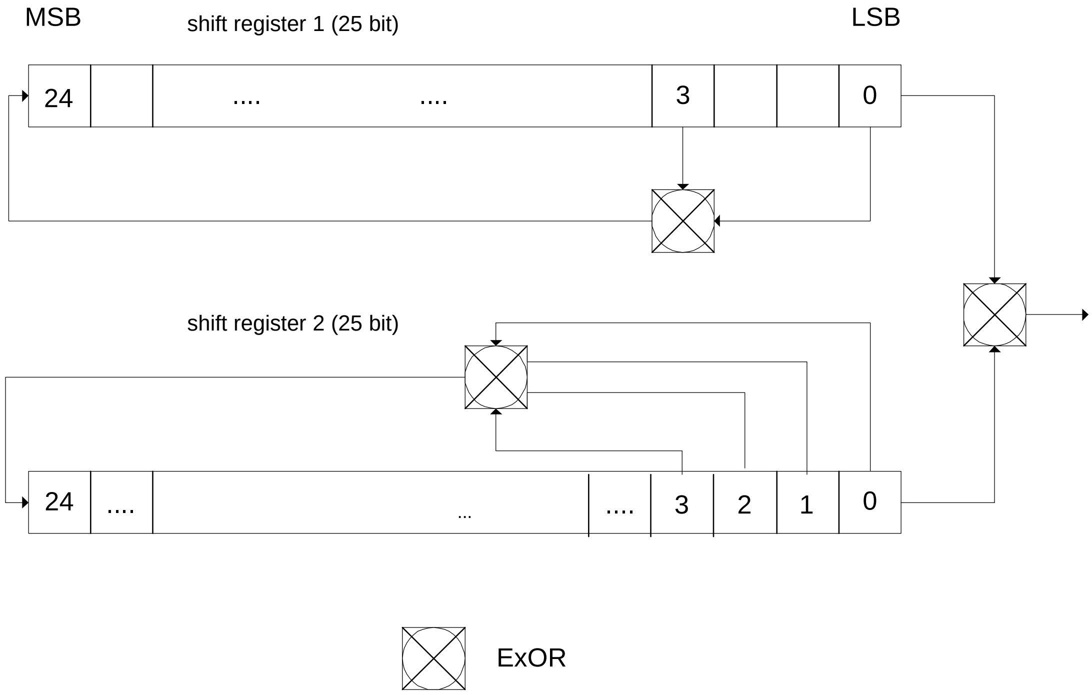

Figure 7. Preamble scrambling code generator

Before transmission these binary code words are converted to real valued sequences by the transformation ‘0’ -> ‘+1’,

4.3.3.2 Preamble signature

The preamble part consists of 256 repetitions of a length 16 signature,$\langle P_0,P_1,…,P_{15}\rangle$. Before scrambling the preamble is therefore

$$P_0, P_1, \cdots, P_{15}, P_0, P_1, \cdots, P_{15}, \cdots\cdots, P_0, P_1, \cdots, P_{15}$$

The signature is from the set of 16 Hadamard codes of length 16. These are listed in Table 3.

	Preamble symbols															
Signature	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
1	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
2	A	-A	A	-A	A	-A	A	-A	A	-A	A	-A	A	-A	A	-A
3	A	A	-A	-A	A	A	-A	-A	A	A	-A	-A	A	A	-A	-A
4	A	-A	-A	A	A	-A	-A	A	A	-A	-A	A	A	-A	-A	A
5	A	A	A	A	-A	-A	-A	-A	A	A	A	A	-A	-A	-A	-A
6	A	-A	A	-A	-A	A	-A	A	A	-A	A	-A	-A	A	-A	A
7	A	A	-A	-A	-A	-A	A	A	A	A	-A	-A	-A	-A	A	A
8	A	-A	-A	A	-A	A	A	-A	A	-A	-A	A	-A	A	A	-A
9	A	A	A	A	A	A	A	A	-A	-A	-A	-A	-A	-A	-A	-A
10	A	-A	A	-A	A	-A	A	-A	-A	A	-A	A	-A	A	-A	A
11	A	A	-A	-A	A	A	-A	-A	-A	-A	A	A	-A	-A	A	A
12	A	-A	-A	A	A	-A	-A	A	-A	A	A	-A	-A	A	A	-A
13	A	A	A	A	-A	-A	-A	-A	-A	-A	-A	-A	A	A	A	A
14	A	-A	A	-A	-A	A	-A	A	-A	A	-A	A	A	-A	A	-A
15	A	A	-A	-A	-A	-A	A	A	-A	-A	A	A	A	A	-A	-A
16	A	-A	-A	A	-A	A	A	-A	-A	A	A	-A	A	-A	-A	A

Table 3. Preamble signatures

The value of A = +1 in bipolar representation which is equivalent to 0 in boolean representation.

4.3.3.3 Preamble PAPR reduction

In order to reduce the PAPR during RACH preamble transmission the following technique is used.

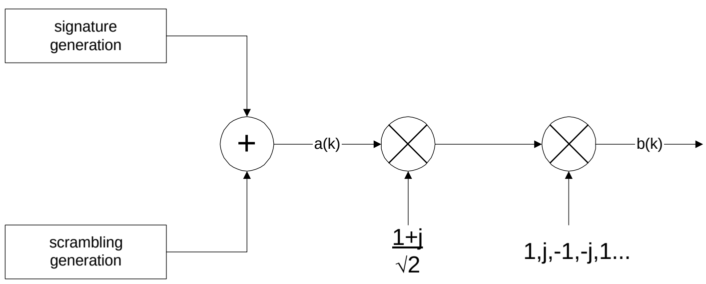

Figure 8 - Baseband modulator for RACH preamble.

The binary preamble $a(k)$ is modulated to get the complex valued preamble $b(k)$,

$$b(k) = a(k)\; e^{j(\frac{\pi}{4} + \frac{\pi}{2}k)}, \; k = 0, 1, 2, 3, \ldots, 4095.$$

4.3.3.4 Channelization codes for the message part

The preamble signature s, $1 \le s \le 16$, points to one of the 16 nodes in the code-tree that corresponds to channelization codes of length 16. The sub-tree below the specified node is used for spreading of the message part. The control part is spread with the channelization code $C_{ch,c}$ of spreading factor 256 in the lowest branch of the sub-tree, i.e. $C_{ch,c} = c_{256,m}$ where m = 16(s – 1) + 15. The spread control part is mapped to the Q-branch, similar to the DPCCH for dedicated channels.

The data part uses any of the channelization codes from spreading factor 32 to 256 in the upper-most branch of the sub-tree. To be exact, the data part is spread by channelization code $C_{ch,d}$, where $C_{ch,d} = c_{SF,m}$ and SF is the spreading factor used for the data part and m = SF×(s – 1)/16.

4.3.3.5 Scrambling code for the message part

In addition to spreading, the message part is also subject to scrambling with a 10 ms complex code. The scrambling code is cell-specific and has a one-to-one correspondence to the scrambling code used for the preamble part.

The scrambling codes used are formed from the continuation of the sequences x_n and y used for the preamble scrambling code and described in 4.3.3.1. Specifically, the values $x_n(4096)$, $x_n(4097),\ldots,x_n(4295)$ and $y(4096),y(4097),\ldots,y(4295)$ are generated according to the recursive relations in 4.3.3.1 and used to form the *n*th constituent codes,$c_{1,n}$, and $c_{2,n}$ (the left most index corresponds to the first chip scrambled in the message):

$c_{1,n}= <x_n(4096)+y(4096), x_n(4097)+y(4097), \ldots,x_n(42495)+y(42495) >$,

$c_{2,n}= <x_n(M+4096)+y(M+4096), x_n(M+4097)+y(M+4097), \ldots,x_n(M+42495)+y(M+42495) >$,

where M is defined in 4.3.2.2. The scrambling code for the message part is then

$C_{MSG,n}= c_{1,n} (w_0+j\, c'_{2,n}w_1)$

where w_0 and w_1 are defined in 4.3.2.1 and $c'_{2,n}$ is a decimated version of $c_{2,n}$ as described in 4.3.2.1.

The generation of these codes is explained in Section 4.3.2.2. The mapping of these codes to provide a complex scrambling code is also the same as for the other dedicated uplink channels and is described in Section 4.3.2.

4.3.4 Common packet channel codes

4.3.4.1 Access preamble scrambling code

The access preamble scrambling code generation is done in the same way as for the PRACH with a difference of the initialisation of the x m-sequence. The long code $c_{1,257}$ (see 4.3.3.2) for the in-phase component is used directly on both in phase and quadrature branches without offset between branches.

4.3.4.2 CD preamble spreading code

The scrambling code for the access preamble is also used as the CD preamble spreading code. The 4096 chips from 4096 to 8191 of the code are used for the CD preamble spreading with the chip rate of 3.84 Mchip/s. The long code c_{257} for the in-phase component is used directly on both in phase and quadrature branches without offset between branches.

4.3.4.3 CPCH preamble

4.3.4.3.1 Access preamble signature

The access preamble part of the CPCH-access burst carries one of the sixteen different orthogonal complex signatures identical to the ones used by the preamble part of the random-access burst.

4.3.4.3.2 CD preamble signature

The CD-preamble part of the CPCH-access burst carries one of sixteen different orthogonal complex signatures identical to the ones used by the preamble part of the random-access burst.

4.3.4.4 Channelization codes for the CPCH message part

The signature in the preamble specifies one of the 16 nodes in the code-tree that corresponds to channelization codes of length 16. The sub-tree below the specified node is used for spreading of the message part. The control part is always spread with a channelization code of spreading factor 256. The code is chosen from the lowest branch of the sub-tree. The data part may use channelization codes from spreading factor 4 to 64. A UE is allowed to increase its spreading factor during the message transmission by choosing any channelization code from the uppermost branch of the sub-tree code. For channelization codes with spreading factors less that 16, the node is located on the same sub-tree as the channelization code of the access preamble.

4.3.4.5 Scrambling code for the CPCH message part

In addition to spreading, the message part is also subject to scrambling. The scrambling code is cell-specific and has a one-to-one correspondence to the spreading code used for the preamble part.

The scrambling codes used are from the same set of codes as is used for the other dedicated uplink channels when the long scrambling codes are used for these channels. The long scrambling codes (c_{257} to c_{512}) of the uplink long scrambling code set are used for the CPCH message part (see section 4.3.2.2). The phases 8192 and above of the codes are used for the message part (phases 0 to 4095 of c_{257} are used in the access preamble spreading and phases 4096 to 8191 for the CD preamble) with the chip rate of 3.84 Mchips/s.

The mapping of these codes to provide a complex scrambling code is also the same as for the other dedicated uplink channels and is described in Section 4.3.2.

4.4 Modulation

4.4.1 Modulating chip rate

The modulating chip rate is 3.84 Mcps.

4.4.2 Modulation

In the uplink, the modulation of both DPCCH and DPDCH is BPSK.

5 Downlink spreading and modulation

5.1 Spreading

Figure 9 illustrates the spreading and modulation for the downlink DPCH. Data modulation is QPSK where each pair of two bits are serial-to-parallel converted and mapped to the I and Q branch respectively. The I and Q branch are then spread to the chip rate with the same channelization code c_{ch} (real spreading) and subsequently scrambled by the scrambling code C_{scramb} (complex scrambling).

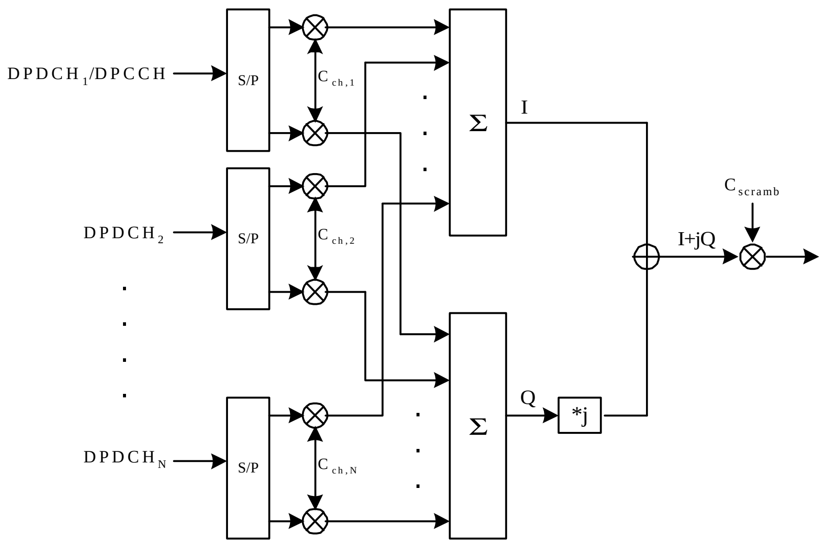

Figure 9. Spreading/modulation for downlink DPCH.

Spreading/modulation of the CPICH, Secondary CCPCH, PSCCCH, PDSCH, PICH and AICH is done in an identical way as for the downlink DPCH.

Spreading/modulation of the Primary CCPCH is done in an identical way as for the downlink DPCH, except that the Primary CCPCH is time multiplexed after spreading. As illustrated in Figure 10. Primary SCH and Secondary SCH are code multiplexed and transmitted simultaneously during the 1st 256 chips of each slot. The transmission power of SCH can be adjusted by a gain factor $G_{P\text{-}SCH}$ and $G_{S\text{-}SCH}$, respectively, independent of transmission power of P-CCPCH. The SCH is *non-orthogonal* to the other downlink physical channels.

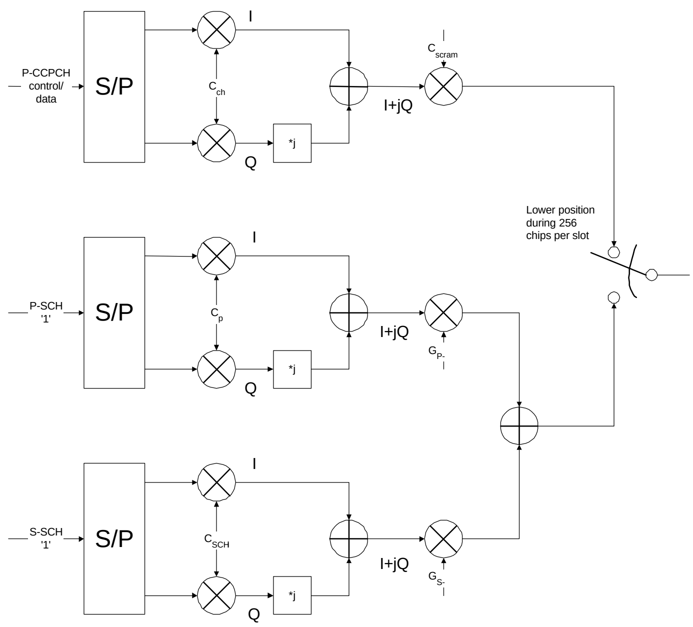

Figure 10. Spreading and modulation for SCH and P-CCPCH

Figure 11 illustrates the detailed generation of an AICH access slot. Note that this is an example implementation.

The AI-part of the access slot consists of the symbol-wise sum of up to 16 orthogonal code words w1-w16, multiplied by the value of the corresponding acquisition indicator AIi. The orthogonal code words w1,...,w16 are shown in Table 4.

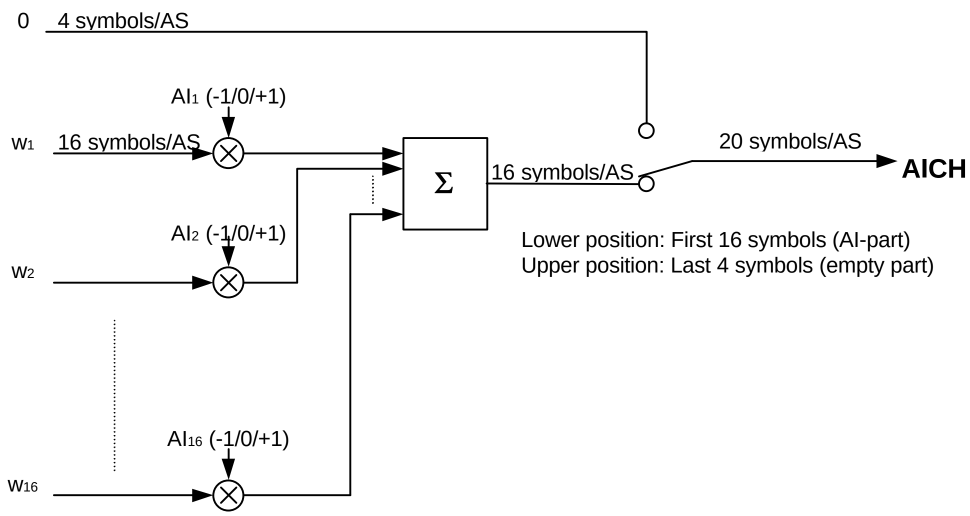

Figure 11. Schematic generation of AICH

I	w_I															
1	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
2	A	-A	A	-A	A	-A	A	-A	A	-A	A	-A	A	-A	A	-A
3	A	A	-A	-A	A	A	-A	-A	A	A	-A	-A	A	A	-A	-A
4	A	-A	-A	A	A	-A	-A	A	A	-A	-A	A	A	-A	-A	A
5	A	A	A	A	-A	-A	-A	-A	A	A	A	A	-A	-A	-A	-A
6	A	-A	A	-A	-A	A	-A	A	A	-A	A	-A	-A	A	-A	A
7	A	A	-A	-A	-A	-A	A	A	A	A	-A	-A	-A	-A	A	A
8	A	-A	-A	A	-A	A	A	-A	A	-A	-A	A	-A	A	A	-A
9	A	A	A	A	A	A	A	A	-A	-A	-A	-A	-A	-A	-A	-A
10	A	-A	A	-A	A	-A	A	-A	-A	A	-A	A	-A	A	-A	A
11	A	A	-A	-A	A	A	-A	-A	-A	-A	A	A	-A	-A	A	A
12	A	-A	-A	A	A	-A	-A	A	-A	A	A	-A	-A	A	A	-A
13	A	A	A	A	-A	-A	-A	-A	-A	-A	-A	-A	A	A	A	A
14	A	-A	A	-A	-A	A	-A	A	-A	A	-A	A	A	-A	A	-A
15	A	A	-A	-A	-A	-A	A	A	-A	-A	A	A	A	A	-A	-A
16	A	-A	-A	A	-A	A	A	-A	-A	A	A	-A	A	-A	-A	A

Table 4 Definition of orthogonal vectors w1-w16 used in AICH; A = (1+j)

5.2 Code generation and allocation

5.2.1 Channelization codes

The channelization codes of Figure 9 and Figure 10 are the same codes as used in the uplink, namely Orthogonal Variable Spreading Factor (OVSF) codes that preserve the orthogonality between downlink channels of different rates and spreading factors. The OVSF codes are defined in Figure 2 in Section 4.3.1.

The channelization code for the Primary CPICH is fixed to $c_{256,0}$ and the channelization code for the Primary CCPCH is fixed to $c_{256,1}$.The channelization codes for all other physical channels are assigned by UTRAN.

When compressed mode is implemented by reducing the spreading factor by 2, the OVSF code of spreading factor SF/2 on the path to the root of the code tree from the OVSF code assigned for normal frames is used in the compressed frames. For the case where the scrambling code is changed during compressed frames, an even numbered OVSF code used in normal mode results in using the even alternative scrambling code during compressed frames, while an odd numbered OVSF code used in normal mode results in using the odd alternative scrambling code during compressed frames. The even and odd alternative scrambling codes are described in the next section.

In case the OVSF code on the PDSCH varies from frame to frame, the OVSF codes shall be allocated such a way that the OVSF code(s) below the smallest spreading factor will be from the branch of the code tree pointed by the smallest spreading factor used for the connection. This means that all the codes for UE for the PDSCH connection can be generated according to the OVSF code generation principle from smallest spreading factor code used by the UE on PDSCH.

In case of multicode PDSCH allocation, the same rule applies, but all of the branches identified by the multiple codes, corresponding to the smallerst spreading factor, may be used for higher spreading factor allocation.

5.2.2 Scrambling code

A total of 2^{18}-1 = 262,143 scrambling codes, numbered 0…262,142 can be generated. However not all the scrambling codes are used. The scrambling codes are divided into 512 sets each of a primary scrambling code and 15 secondary scrambling codes.

The primary scrambling codes consist of scrambling codes n=16*i where i=0…511. The i:th set of secondary scrambling codes consists of scrambling codes 16*i+k, where k=1…15.

There is a one-to-one mapping between each primary scrambling code and 15 secondary scrambling codes in a set such that i:th primary scrambling code corresponds to i:th set of scrambling codes.

Hence, according to the above, scrambling codes k = 0, 1, …, 8191 are used. Each of these codes are associated with an even alternative scrambling code and an odd alternative scrambling code, that may be used for compressed frames. The even alternative scrambling code corresponding to scrambling code k is scrambling code number k + 8192, while the odd alternative scrambling code corresponding to scrambling code k is scrambling code number k + 16384.

The set of primary scrambling codes is further divided into 64 scrambling code groups, each consisting of 8 primary scrambling codes. The j:th scrambling code group consists of primary scrambling codes 16*8*j+16*k, where j=0..63 and k=0..7.

Each cell is allocated one and only one primary scrambling code. The primary CCPCH is always transmitted using the primary scrambling code. The other downlink physical channels can be transmitted with either the primary scrambling code or a secondary scrambling code from the set associated with the primary scrambling code of the cell.

The mixture of primary scrambling code and secondary scrambling code for one CCTrCH is allowable.

The scrambling code sequences are constructed by combining two real sequences into a complex sequence. Each of the two real sequences are constructed as the position wise modulo 2 sum of 38400 chip segments of two binary *m*-sequences generated by means of two generator polynomials of degree 18. The resulting sequences thus constitute segments of a set of Gold sequences. The scrambling codes are repeated for every 10 ms radio frame. Let x and y be the two sequences respectively. The x sequence is constructed using the primitive (over GF(2)) polynomial $1+X^7+X^{18}$. The y sequence is constructed using the polynomial $1+X^5+X^7+ X^{10}+X^{18}$.

The x sequence depends on the chosen scrambling code number n and is denoted x_n, in the sequel. Furthermore, let $x_n(i)$ and $y(i)$ denote the *i:*th symbol of the sequence x_n and y, respectively

The *m*-sequences x_n and y are constructed as:

Initial conditions:

x_0 *is constructed with* $x_0(0)= x_0 (1)=...= x_0 (16)=0, x_0 (17)=1$

$y(0)=y(1)= \dots =y(16)= y(17)=1$

Recursive definition of subsequent symbols:

$x_n(i+18) =x_n(i+7) + x_n(i) \text{ modulo } 2, i=0,\dots,2^{18}-20,$

$y(i+18) = y(i+10)+y(i+7)+y(i+5)+y(i) \text{ modulo } 2, i=0,\dots, 2^{18}-20.$

x_n is constructed with the following equation.

$x_n(i) =x_0((i+n) \text{ modulo } 2^{18}-1), \quad i=0,...,2^{18}-2$

The n:th Gold code sequence z_n is then defined as

$z_n(i) = x_n(i) + y(i) \text{ modulo } 2, i=0,\dots, 2^{18}-2.$

These binary code words are converted to real valued sequences by the transformation ‘0’ -> ‘+1’, ‘1’ -> ‘-1’.

Finally, the n:th complex scrambling code sequence C_{scramb} is defined as (the lowest index corresponding to the chip scrambled first in each radio frame): (where N is the period in chips and M is 131,072)

$C_{scramb}(i) = z_n(i) + j\, z_n(i+M),\ i=0,1,,N-1.$

Note that the pattern from phase 0 up to the phase of 38399 is repeated.

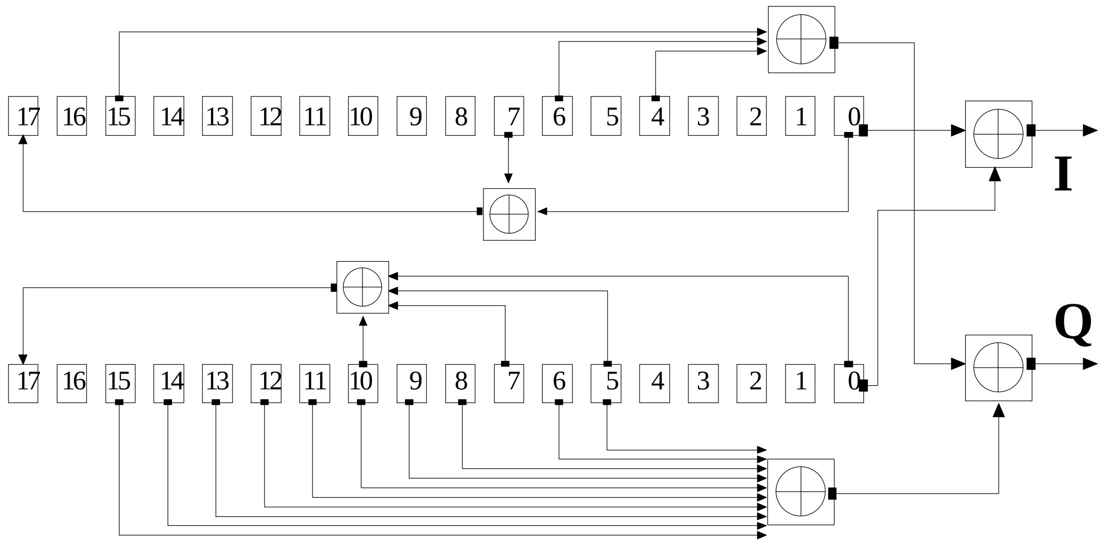

Figure 12. Configuration of downlink scrambling code generator

5.2.3 Synchronisation codes

5.2.3.1 Code Generation

The Primary code sequence, C_p is constructed as a so-called generalised hierarchical Golay sequence. The Primary SCH is furthermore chosen to have good aperiodic auto correlation properties.

Letting a = <x_1, x_2, x_3, …, x_{16}> = <0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0> and

$$b = < x_1, x_2, .., x_8, \overline{x}_9, \overline{x}_{10}, .., \overline{x}_{16} >$$

The PSC code is generated by repeating sequence 'a' modulated by a Golay complementary sequence.

Letting $y = < a, a, a, \overline{a}, \overline{a}, a, \overline{a}, \overline{a}, a, a, a, \overline{a}, a, \overline{a}, a, a >$

The definition of the PSC code word C_p follows (the left most index corresponds to the chip transmitted first in each time slot):

C_p=<y(0),y(1),y(2),...,y(255)>.

Let the sequence $z = \{b,b,b,\overline{b},b,b,\overline{b},\overline{b},b,\overline{b},b,\overline{b},\overline{b},\overline{b},\overline{b},\overline{b}\}$. Then the Secondary Synchronization code words, {C_1,…,C_{16}} are constructed as the position wise addition modulo 2 of a Hadamard sequence and the sequence *z*.

The Hadamard sequences are obtained as the rows in a matrix H_8 constructed recursively by:

$$H_0 = (0)$$

$$H_k = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & \overline{H_{k-1}} \end{pmatrix}, \quad k \geq 1$$

The rows are numbered from the top starting with row *0* (the all zeros sequence).

The Hadamard sequence *h* depends on the chosen code number *n* and is denoted h_n in the sequel.

This code word is chosen from every 16th row of the matrix H_8 implying 16 possible code words given by n =0,16,32,48,64,80,96,112,128,144,160,176,192,208,224,240.

Furthermore, let $h_n(i)$ and $z(i)$ denote the *i:*th symbol of the sequence h_n and *z*, respectively.

The definition of the *n*:th SCH code word follows (the left most index correspond to the chip transmitted first in each slot):

$C_{SCH,n} = < h_n(0)+z(0), h_n(1)+z(1), h_n(2)+z(2), \ldots, h_n(255)+z(255) >$,

All sums of symbols are taken modulo 2.

These PSC and SSC binary code words are converted to real valued sequences by the transformation '0' -> '+1', '1' ->

The Secondary SCH code words are defined in terms of $C_{SCH,n}$ and the definition of $\{C_1,\ldots,C_{16}\}$ now follows as:
$C_i = C_{SCH,i}$, i=1,…,16

5.2.3.2 Code Allocation

The 64 sequences are constructed such that their cyclic-shifts are unique, i.e., a non-zero cyclic shift less than 15 of any of the 64 sequences is not equivalent to some cyclic shift of any other of the 64 sequences. Also, a non-zero cyclic shift less than 15 of any of the sequences is not equivalent to itself with any other cyclic shift less than 15. The following sequences are used to encode the 64 different scrambling code groups (note that c_i indicates the i'th Secondary Short code of the 16 codes). Note that a Secondary Short code can be different from one time slot to another and that the sequence pattern can be different from one cell to another, depending on Scrambling Code Group the cell uses.

Scrambling	slot number														
Code Group	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15
Group 1	1	1	2	8	9	10	15	8	10	16	2	7	15	7	16
Group 2	1	1	5	16	7	3	14	16	3	10	5	12	14	12	10
Group 3	1	2	1	15	5	5	12	16	6	11	2	16	11	15	12
Group 4	1	2	3	1	8	6	5	2	5	8	4	4	6	3	7
Group 5	1	2	16	6	6	11	15	5	12	1	15	12	16	11	2
Group 6	1	3	4	7	4	1	5	5	3	6	2	8	7	6	8
Group 7	1	4	11	3	4	10	9	2	11	2	10	12	12	9	3
Group 8	1	5	6	6	14	9	10	2	13	9	2	5	14	1	13
Group 9	1	6	10	10	4	11	7	13	16	11	13	6	4	1	16
Group 10	1	6	13	2	14	2	6	5	5	13	10	9	1	14	10
Group 11	1	7	8	5	7	2	4	3	8	3	2	6	6	4	5
Group 12	1	7	10	9	16	7	9	15	1	8	16	8	15	2	2
Group 13	1	8	12	9	9	4	13	16	5	1	13	5	12	4	8
Group 14	1	8	14	10	14	1	15	15	8	5	11	4	10	5	4
Group 15	1	9	2	15	15	16	10	7	8	1	10	8	2	16	9
Group 16	1	9	15	6	16	2	13	14	10	11	7	4	5	12	3
Group 17	1	10	9	11	15	7	6	4	16	5	2	12	13	3	14

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

Group 55	5	6	11	7	10	8	5	8	7	12	12	10	6	9	11
Group 56	5	6	13	8	13	5	7	7	6	16	14	15	8	16	15
Group 57	5	7	9	10	7	11	6	12	9	12	11	8	8	6	10
Group 58	5	9	6	8	10	9	8	12	5	11	10	11	12	7	7
Group 59	5	10	10	12	8	11	9	7	8	9	5	12	6	7	6
Group 60	5	10	12	6	5	12	8	9	7	6	7	8	11	11	9
Group 61	5	13	15	15	14	8	6	7	16	8	7	13	14	5	16
Group 62	9	10	13	10	11	15	15	9	16	12	14	13	16	14	11
Group 63	9	11	12	15	12	9	13	13	11	14	10	16	15	14	16
Group 64	9	12	10	15	13	14	9	14	15	11	11	13	12	16	10
Sync BTS	9	12	16	16	10	15	11	13	14	15	13	12	10	9	14

Table 5 Spreading Code allocation for Secondary SCH Code, the index "i" of the code Ci

5.3 Modulation

5.3.1 Modulating chip rate

The modulating chip rate is 3.84 Mcps.

5.3.2 Modulation

QPSK modulation is used.

Annex A Generalised Hierarchical Golay Sequences

A.1 Alternative generation

The generalised hierarchical Golay sequences for the PSC described in 5.2.3.1 may be also viewed as generated (in real valued representation) by the following methods:

Method 1.

The sequence y is constructed from two constituent sequences x_1 and x_2 of length n_1 and n_2 respectively using the following formula:

$y(i) = x_2(i \bmod n_2) * x_1(i \text{ div } n_2), i = 0 \ldots (n_1 * n_2) - 1$

The constituent sequences x_1 and x_2 are chosen to be the following length 16 (i.e. $n_1 = n_2 = 16$) sequences:

- x_1 is defined to be the length 16 ($N^{(1)}=4$) Golay complementary sequence obtained by the delay matrix $D^{(1)} = [8, 4, 1,2]$ and weight matrix $W^{(1)} = [1, -1, 1,1]$.

- x_2 is a generalised hierarchical sequence using the following formula, selecting s=2 and using the two Golay complementary sequences x_3 and x_4 as constituent sequences. The length of the sequence x_3 and x_4 is called n_3 respectively n_4.

  $x_2(i) = x_4(i \bmod s + s*(i \text{ div } sn_3)) * x_3((i \text{ div } s) \bmod n_3), i = 0 \ldots (n_3 * n_4) - 1$

  x_3 and x_4 are defined to be identical and the length 4 ($N^{(3)}= N^{(4)}=2$) Golay complementary sequence obtained by the delay matrix $D^{(3)} = D^{(4)} = [1, 2]$ and weight matrix $W^{(3)} = W^{(4)} = [1, 1]$.

The Golay complementary sequences x_1,x_3 and x_4 are defined using the following recursive relation:

$$a_0(k) = \delta(k) \text{ and } b_0(k) = \delta(k)$$

$$a_n(k) = a_{n-1}(k) + W^{(j)}_n \cdot b_{n-1}(k-D^{(j)}_n) ,$$

$$b_n(k) = a_{n-1}(k) - W^{(j)}_n \cdot b_{n-1}(k-D^{(j)}_n) ,$$

$$k = 0, 1, 2, \ldots, 2^{**}N^{(j)} -1,$$

$$n = 1, 2, \ldots, N^{(j)}.$$

The wanted Golay complementary sequence x_j is defined by a_n assuming $n=N^{(j)}$. The Kronecker delta function is described by δ, k,j and n are integers.

Method 2

The sequence y can be viewed as a pruned Golay complementary sequence and generated using the following parameters which apply to the generator equations for a and b above:

(a) Let $j = 0$, $N^{(0)} = 8$

(b) $[D_1^0,D_2^0,D_3^0,D_4^0,D_5^0,D_6^0,D_7^0,D_8^0] = [128, 64, 16, 32, 8, 1, 4, 2]$

(c) $[W_1^0,W_2^0,W_3^0,W_4^0,W_5^0,W_6^0,W_7^0,W_8^0] = [1, -1, 1, 1, 1, 1, 1, 1]$

(d) For $n = 4, 6$, set $b_4(k) = a_4(k)$, $b_6(k) = a_6(k)$.

6 History

Document history		
draft	1999-02-12	New document merged from ETSI XX.05 and ARIB 3.2.4 sources.
0.0.1	1999-02-12	Corrected typo in table2.
0.0.2	1999-02-16	Added sec. SCH code table, option for HPSK on S(2) codes, scale on SCH.
0.0.3	1999-02-18	Reflected decision made on SCH multiplexing (see document titled 'Report from Ad Hoc #2 SCH multiplexing'.) and additional description on the use of S(2) for uplink short scrambling code.
0.1.0	1999-02-28	Raised to 0.1.0 after TSG RAN WG1#2 meeting (Yokohama).
1.0.0	1999-03-12	Raised to 1.0.0 when presented to TSG RAN.
1.0.1	1999-03-17	Raised to 1.0.1 incorporated Ad Hoc changes and errata from e-mail.
1.0.2	1999-03-23	Raised to 1.0.2 incorporated reports from Ad Hocs plus editorial matters.
1.0.3	1999-03-24	Raised to 1.0.3 incorporated actions from WG1#3 plenary..
1.1.0	1999-03-26	Raised to 1.1.0 changed as result of text proposal, Tdoc 298.
1.1.1	1999-04-12	Raised to 1.1.1 by incorporating 3GPP template and adding editor's note.
1.1.2	1999-04-12	Raised to 1.1.2 by entering editorial changes with revision marks.
1.1.3	1999-04-19	Rasied to 1.1.3 by Tdocs 347, 385 at WG1#4 meeting (Yokohama)
1.1.4	1999-04-20	Raised to 1.1.4 by Tdoc 397 at WG1#4 meeting (Yokohama)
2.0.0	1999-04-20	Raised to 2.0.0 at WG1#4 (Yokohama) for presentation to RSG RAN.
2.0.1	1999-04-27	Raised to 2.0.1 fixing references in section 4.3.2.3, fixed figures 10, 11.
2.0.2	1999-06-04	Raised to 2.0.2 at WG1#5 (Cheju) plenary.
2.1.0	1999-06-04	Raised to 2.1.0 at WG1#5 plenary for presentation to TSG RAN.
2.1.1	1999-06-22	Raised to 2.1.1 due to editorial changes noted after WG1#5.
2.1.2	1999-07-20	Raised to 2.1.2 due to editorial changes noted offline and proposals at WG1#6.
2.2.0	1999-08-30	Raised to 2.2.0 at WG1#7 (Hannover) plenary.
2.2.1	1999-09-03	Raised to 2.2.1 as a result of text proposals at WG1#7 (Hannover).
2.3.0	1999-09-03	Raised to 2.3.0 at WG1#7 (Hannover) plenary.

Editor for 25.213, spreading and modulation specification, is:

Peter Chambers
Siemens Roke Manor Research
Email: peter.chambers@roke.co.uk

This document is written in Microsoft Word 97.

ISBN <G-EEEE-NNNN-K>
Dépôt légal : <Mois AAAA>