ETSI TS 125 223 V7.3.0 (2007-03)

Technical Specification

**Universal Mobile Telecommunications System (UMTS);
Spreading and modulation (TDD)
(3GPP TS 25.223 version 7.3.0 Release 7)**

Reference

RTS/TSGR-0125223v730

Keywords

UMTS

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Specification (TS) has been produced by the 3rd Generation Partnership Project (3GPP).

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

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

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

2 References

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

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 25.201: "Physical layer - general description".

[2] 3GPP TS 25.211: "Physical channels and mapping of transport channels onto physical channels (FDD)".

[3] 3GPP TS 25.212: "Multiplexing and channel coding (FDD)".

[4] 3GPP TS 25.213: "Spreading and modulation (FDD)".

[5] 3GPP TS 25.214: "Physical layer procedures (FDD)".

[6] 3GPP TS 25.215: "Physical layer – Measurements (FDD)".

[7] 3GPP TS 25.221: "Physical channels and mapping of transport channels onto physical channels (TDD)".

[8] 3GPP TS 25.222: "Multiplexing and channel coding (TDD)".

[9] 3GPP TS 25.102: "UTRA (UE) TDD; Radio Transmission and Reception".

[10] 3GPP TS 25.105: "UTRA (BS) TDD; Radio Transmission and Reception".

[11] 3GPP TS25.308: "High Speed Downlink Packet Access (HSDPA); Overall description; Stage 2".

[12] 3GPP TS25.224: 'Physical Layer Procedures (TDD)'

[13] 3GPP TS25.321: 'Medium Access Control (MAC) protocol specification'

3 Symbols and abbreviations

3.1 Symbols

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

C_p:	PSC
C_i:	i:th secondary SCH code
$C_{CSC,m}^{(k)}$:	CSC derived as *k*:th offset version from *m*:th applicable constituent Golay complementary pair

3.2 Abbreviations

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

16QAM	16 Quadrature Amplitude Modulation
CCTrCH	Coded Composite Transport Channel
DPCH	Dedicated Physical Channel
CDMA	Code Division Multiple Access
CSC	Cell Synchronisation Code
FDD	Frequency Division Duplex
HS-PDSCH	High Speed Physical Downlink Shared Channel
MIB	Master Information Block
OVSF	Orthogonal Variable Spreading Factor
P-CCPCH	Primary Common Control Physical Channel
PN	Pseudo Noise
PRACH	Physical Random Access Channel
PSC	Primary Synchronisation Code
QPSK	Quadrature Phase Shift Keying
RACH	Random Access Channel
SCH	Synchronisation Channel
SF	Spreading Factor
SFN	System Frame Number
TDD	Time Division Duplex
TFC	Transport Format Combination
UE	User Equipment
UL	Uplink

4 General

In the following, a separation between the data modulation and the spreading modulation has been made. The data modulation for 3.84Mcps TDD and 7.68Mcps TDD is defined in clause 5 'Data modulation for the 3.84 Mcps and 7.68Mcps options', the data modulation for 1.28Mcps TDD is defined in clause 5A 'Data modulation for the 1.28 Mcps option' and the spreading modulation in clause 6 'Spreading modulation'.

Table 1 shows the basic modulation parameters for the 7.68Mcps, 3.84Mcps and 1.28Mcps TDD options.

Table 1: Basic modulation parameters

Chip rate	7.68 Mchip/s	same as FDD basic chiprate: 3.84 Mchip/s	Low chiprate: 1.28 Mchip/s
Data modulation	QPSK,16QAM (HS-PDSCH, E-PUCH only)	QPSK,16QAM (HS-PDSCH, E-PUCH only)	QPSK, 8PSK,16QAM (HS-PDSCH, E-PUCH only)
Spreading characteristics	Orthogonal Q chips/symbol, where $Q = 2^p$, $0 <= p <= 5$	Orthogonal Q chips/symbol, where $Q = 2^p$, $0 <= p <= 4$	Orthogonal Q chips/symbol, where $Q = 2^p$, $0 <= p <= 4$

5 Data modulation for the 3.84 Mcps and 7.68Mcps options

5.1 Symbol rate

The symbol duration T_S depends on the spreading factor Q and the chip duration T_C: $T_s = Q \times T_c$, where $T_c = \frac{1}{chiprate}$.

5.2 Mapping of bits onto signal point constellation

5.2.1 Mapping for burst type 1 and 2

5.2.1.1 QPSK modulation

The data modulation is performed to the bits from the output of the physical channel mapping procedure in [8] and combines always 2 consecutive binary bits to a complex valued data symbol. Each user burst has two data carrying parts, termed data blocks:

$$\underline{\mathbf{d}}^{(k,i)} = \left(\underline{d}_1^{(k,i)}, \underline{d}_2^{(k,i)}, \ldots, \underline{d}_{N_k}^{(k,i)}\right)^T, \quad i = 1,2; k = 1,\ldots,K_{Code} \qquad (1)$$

K_{Code} is the number of used codes in a time slot: for 3.84Mcps, max K_{Code} =16; for 7.68Mcps, max K_{Code} =32. N_k is the number of symbols per data field for the code k. This number is linked to the spreading factor Q_k [7].

Data block $\underline{\mathbf{d}}^{(k,1)}$ is transmitted before the midamble and data block $\underline{\mathbf{d}}^{(k,2)}$ after the midamble. Each of the N_k data symbols $\underline{d}_n^{(k,i)}$; i=1, 2; k=1,...,K_{Code}; n=1,...,N_k; of equation 1 has the symbol duration $T_S^{(k)}=Q_k.T_C$ as already given.

The data modulation is QPSK, thus the data symbols $\underline{d}_n^{(k,i)}$ are generated from two consecutive data bits from the output of the physical channel mapping procedure in [8]:

$$b_{l,n}^{(k,i)} \in \{0,1\}, \quad l = 1,2; k = 1,\ldots,K_{Code}; n = 1,\ldots,N_k; i = 1,2 \qquad (2)$$

using the following mapping to complex symbols:

consecutive binary bit pattern	complex symbol
$^{(k,i)}_{1,n}$ $^{(k,i)}_{2,n}$	$\underline{d}_n^{(k,i)}$
00	+j
01	+1
10	-1
11	-j

The mapping corresponds to a QPSK modulation of the interleaved and encoded data bits $b_{l,n}^{(k,i)}$ of equation 2.

5.2.1.2 16QAM modulation

The data modulation is performed to the bits from the output of the physical channel mapping procedure. In case of 16QAM, modulation 4 consecutive binary bits are represented by one complex valued data symbol. Each user burst has two data carrying parts, termed data blocks:

$$\underline{\mathbf{d}}^{(k,i)} = (\underline{d}_1^{(k,i)}, \underline{d}_2^{(k,i)}, \ldots, \underline{d}_{N_k}^{(k,i)})^{\mathrm{T}} \quad \mathrm{i} = 1, 2; \mathrm{k} = 1,\ldots,\mathrm{K}. \qquad (2b)$$

N_k is the number of symbols per data field for the user k. This number is linked to the spreading factor Q_k.

Data block $\underline{\mathbf{d}}^{(k,1)}$ is transmitted before the midamble and data block $\underline{\mathbf{d}}^{(k,2)}$ after the midamble. Each of the N_k data symbols $\underline{d}_n^{(k,i)}$; i=1, 2; k=1,...,K; n=1,...,N_k; of equation 2b has the symbol duration $T_S^{(k)}=Q_k.T_C$ as already given.

The data modulation is 16QAM, thus the data symbols $\underline{d}_n^{(k,i)}$ are generated from 4 consecutive data bits from the output of the physical channel mapping procedure in [8]:

$$b_{l,n}^{(k,i)} \in \{0,1\}, \quad l=1,2,3,4;\ k=1,...,K_{code};\ n=1,...N_k;\ i=1,2 \tag{2c}$$

using the following mapping to complex symbols:

Consecutive binary bit pattern	complex symbol
$b_{1,n}^{(k,i)}\ b_{2,n}^{(k,i)}\ b_{3,n}^{(k,i)}\ b_{4,n}^{(k,i)}$	$\underline{d}_n^{(k,i)}$
0000	$j\frac{1}{\sqrt{5}}$
0001	$-\frac{1}{\sqrt{5}}+j\frac{2}{\sqrt{5}}$
0010	$\frac{1}{\sqrt{5}}+j\frac{2}{\sqrt{5}}$
0011	$j\frac{3}{\sqrt{5}}$
0100	$\sqrt{\frac{1}{5}}$
0101	$\frac{2}{\sqrt{5}}-j\frac{1}{\sqrt{5}}$
0110	$\frac{2}{\sqrt{5}}+j\frac{1}{\sqrt{5}}$
0111	$\frac{3}{\sqrt{5}}$
1000	$-\frac{1}{\sqrt{5}}$
1001	$-\frac{2}{\sqrt{5}}+j\frac{1}{\sqrt{5}}$
1010	$-\frac{2}{\sqrt{5}}-j\frac{1}{\sqrt{5}}$
1011	$-\frac{3}{\sqrt{5}}$
1100	$-j\frac{1}{\sqrt{5}}$
1101	$\frac{1}{\sqrt{5}}-j\frac{2}{\sqrt{5}}$
1110	$-\frac{1}{\sqrt{5}}-j\frac{2}{\sqrt{5}}$
1111	$-j\frac{3}{\sqrt{5}}$

The mapping corresponds to a 16QAM modulation of the interleaved and encoded data bits $b_{l,n}^{(k,i)}$ of the table above and $\underline{d}_n^{(k,i)}$ of equation 2b.

5.2.2 Mapping for burst type 3

In case of burst type 3, the definitions in subclause 5.2.1.1 and subclause 5.2.1.2 apply with a modified number of symbols in the second data block. For the burst type 3, the number of symbols in the second data block $\underline{\mathbf{d}}^{(k,2)}$ is decreased by $\frac{96}{Q_K}$ symbols for 3.84Mcps TDD and is decreased by $\frac{192}{Q_k}$ symbols for 7.68Mcps TDD.

5A Data modulation for the 1.28 Mcps option

5A.1 Symbol rate

The symbol duration T_S depends on the spreading factor Q and the chip duration T_C: $T_s = Q \times T_c$, where $T_c = \frac{1}{chiprate}$.

5A.2 Mapping of bits onto signal point constellation

5A.2.1 QPSK modulation

The mapping of bits onto the signal point constellation for QPSK modulation is the same as in the 3.84Mcps TDD cf. [5.2.1.1 QPSK modulation].

5A.2.2 8PSK modulation

The data modulation is performed to the bits from the output of the physical channel mapping procedure. In case of 8PSK modulation 3 consecutive binary bits are represented by one complex valued data symbol. Each user burst has two data carrying parts, termed data blocks:

$$\underline{\mathbf{d}}^{(k,i)} = \left(\underline{d}_1^{(k,i)}, \underline{d}_2^{(k,i)}, \ldots, \underline{d}_{N_k}^{(k,i)}\right)^T, \quad i = 1,2; k = 1,\ldots, K_{Code} \tag{1a}$$

N_k is the number of symbols per data field for the code k. This number is linked to the spreading factor Q_k.

Data block $\underline{\mathbf{d}}^{(k,1)}$ is transmitted before the midamble and data block $\underline{\mathbf{d}}^{(k,2)}$ after the midamble. Each of the N_k data symbols $\underline{d}_n^{(k,i)}$; i=1, 2; k=1,...,K_{Code}; n=1,...,N_k; of equation 1 has the symbol duration $T_S^{(k)}=Q_k.T_C$ as already given.

The data modulation is 8PSK, thus the data symbols $\underline{d}_n^{(k,i)}$ are generated from 3 consecutive data bits from the output of the physical channel mapping procedure in [8]:

$$b_{l,n}^{(k,i)} \in \{0,1\} \quad l = 1,2,3; k = 1,\ldots, K_{Code}; n = 1,\ldots, N_k; i = 1,2 \tag{2a}$$

using the following mapping to complex symbols:

Consecutive binary bit pattern	complex symbol
$b_{1,n}^{(k,i)}\ b_{2,n}^{(k,i)}\ b_{3,n}^{(k,i)}$	$\underline{d}_n^{(k,i)}$
000	cos(11pi/8)+ jsin(11pi/8)
001	cos(9pi/8)+ jsin(9pi/8)
010	cos(5pi/8)+ jsin(5pi/8)
011	cos(7pi/8)+ jsin(7pi/8)
100	cos(13pi/8)+ jsin(13pi/8)
101	cos(15pi/8)+ jsin(15pi/8)
110	cos(3pi/8)+ jsin(3pi/8)
111	cos(pi/8)+ jsin(pi/8)

The mapping corresponds to a 8PSK modulation of the interleaved and encoded data bits $b_{l,n}^{(k,i)}$ of the table above and $\underline{d}_n^{(k,i)}$ of equation 1a.

5A.2.3 16QAM modulation

The mapping of bits onto the signal point constellation for 16QAM modulation is the same as in the 3.84Mcps TDD cf. [5.2.1.2 16QAM modulation].

6 Spreading modulation

6.1 Basic spreading parameters

Spreading of data consists of two operations: Channelisation and Scrambling. Firstly, each complex valued data symbol $\underline{d}_n^{(k,i)}$ of equation 1 (or $e_n^{(k,i)}$ of equation 8 in the case of E-HICH) is spread with a real valued channelisation code $\mathbf{c}^{(k)}$ of length Q_k: for 3.84Mcps TDD and 1.28Mcps TDD, $Q_k \in \{1,2,4,8,16\}$; for 7.68Mcps TDD, $Q_k \in \{1,2,4,8,16,32\}$. The resulting sequence is then scrambled by a complex sequence $\underline{\mathbf{v}}$: the sequence is $\underline{\mathbf{v}}$ of length 16 for the 3.84Mcps and 1.28Mcps options; it is of length 32 for the 7.68Mcps option.

6.2 Channelisation codes

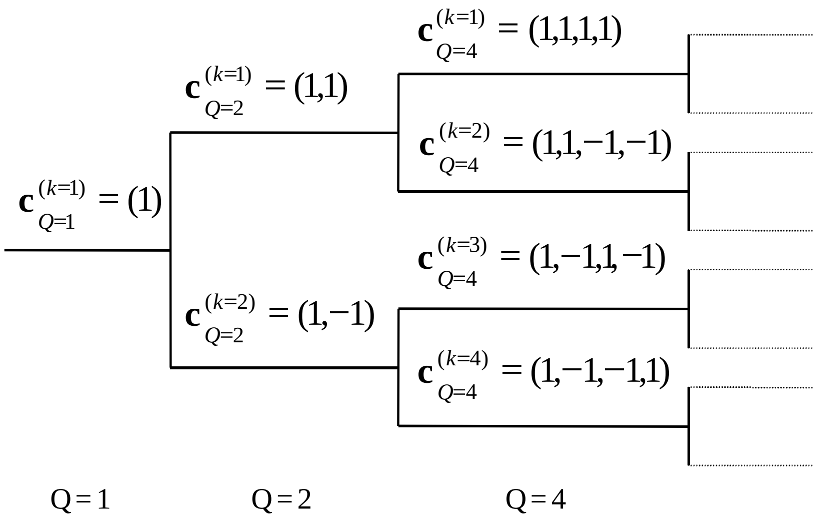
The elements $c_q^{(k)}$; k=1,...,K_{Code}; q=1,...,Q_k; of the real valued channelisation codes

$$\mathbf{c}^{(k)}=(c_1^{(k)},c_2^{(k)},...,c_{Q_k}^{(k)})\ ;\ k=1,...,K_{\text{Code}};$$

shall be taken from the set

$$\mathrm{V_c} = \{1, -1\} \qquad (3)$$

The $\mathbf{c}_{Q_k}^{(k)}$ are Orthogonal Variable Spreading Factor (OVSF) codes, allowing to mix in the same timeslot channels with different spreading factors while preserving the orthogonality. The OVSF codes can be defined using the code tree of figure 1.



$\mathbf{c}_{Q=4}^{(k=1)} = (1,1,1,1)$
$\mathbf{c}_{Q=2}^{(k=1)} = (1,1)$
$\mathbf{c}_{Q=4}^{(k=2)} = (1,1,-1,-1)$
$\mathbf{c}_{Q=1}^{(k=1)} = (1)$
$\mathbf{c}_{Q=4}^{(k=3)} = (1,-1,1,-1)$
$\mathbf{c}_{Q=2}^{(k=2)} = (1,-1)$
$\mathbf{c}_{Q=4}^{(k=4)} = (1,-1,-1,1)$
Q = 1
Q = 2
Q = 4

Figure 1: Code-tree for generation of Orthogonal Variable Spreading Factor (OVSF) codes for Channelisation Operation

Each level in the code tree defines a spreading factor indicated by the value of Q in the figure. All codes within the code tree cannot be used simultaneously in a given timeslot. A code can be used in a timeslot if and only if no other code on the path from the specific code to the root of the tree or in the sub-tree below the specific code is used in this timeslot. This means that the number of available codes in a slot is not fixed but depends on the rate and spreading factor of each physical channel.

For the 3.84Mcps and 1.28Mcps TDD options, the spreading factor goes up to Q_{MAX}=16; for the 7.68Mcps TDD option, the spreading factor goes up to Q_{MAX}=32.

6.3 Channelisation Code Specific Multiplier

Associated with each channelisation code is a multiplier $w_{Q_k}^{(k)}$ taking values from the set $\left\{e^{j\pi/2 \cdot p_k}\right\}$, where p_k is a permutation of the integer set {0, ..., Q_k -1} and Q_k denotes the spreading factor. The multiplier is applied to the data sequence modulating each channelisation code. The values of the multiplier for each channelisation code are given in the table below:

k	$w_{Q=1}^{(k)}$	$w_{Q=2}^{(k)}$	$w_{Q=4}^{(k)}$	$w_{Q=8}^{(k)}$	$w_{Q=16}^{(k)}$	$w_{Q=32}^{(k)}$
1	1	1	-j	1	-1	-j
2		+j	1	+j	-j	-1
3			+j	+j	1	-1
4			-1	-1	1	1
5				-j	+j	-1
6				-1	-1	-j
7				-j	-1	j
8				1	1	1
9					-j	-1
10					+j	1
11					1	1
12					+j	-j
13					-j	j
14					-j	-1
15					+j	j
16					-1	-j
17						-j
18						-j
19						1
20						j
21						-1
22						-j
23						-j
24						-j
25						-1
26						-1
27						j
28						-1
29						-j
30						1
31						-1
32						-1

NOTE: the multiplier $w_{Q=32}^{(k)}$ may only be applied in the 7.68Mcps TDD option.

If the UE autonomously changes the SF, as described in [7], it shall always use the multiplier associated with the channelisation code allocated by higher layers.

6.4 Scrambling codes for the 3.84Mcps and 1.28Mcps options

The spreading of data by a real valued channelisation code $\mathbf{c}^{(k)}$ of length Q_k is followed by a cell specific complex scrambling sequence $\underline{\mathbf{v}} = (\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_{16})$. The elements $\underline{v}_i; i = 1,\ldots,16$ of the complex valued scrambling codes shall be taken from the complex set

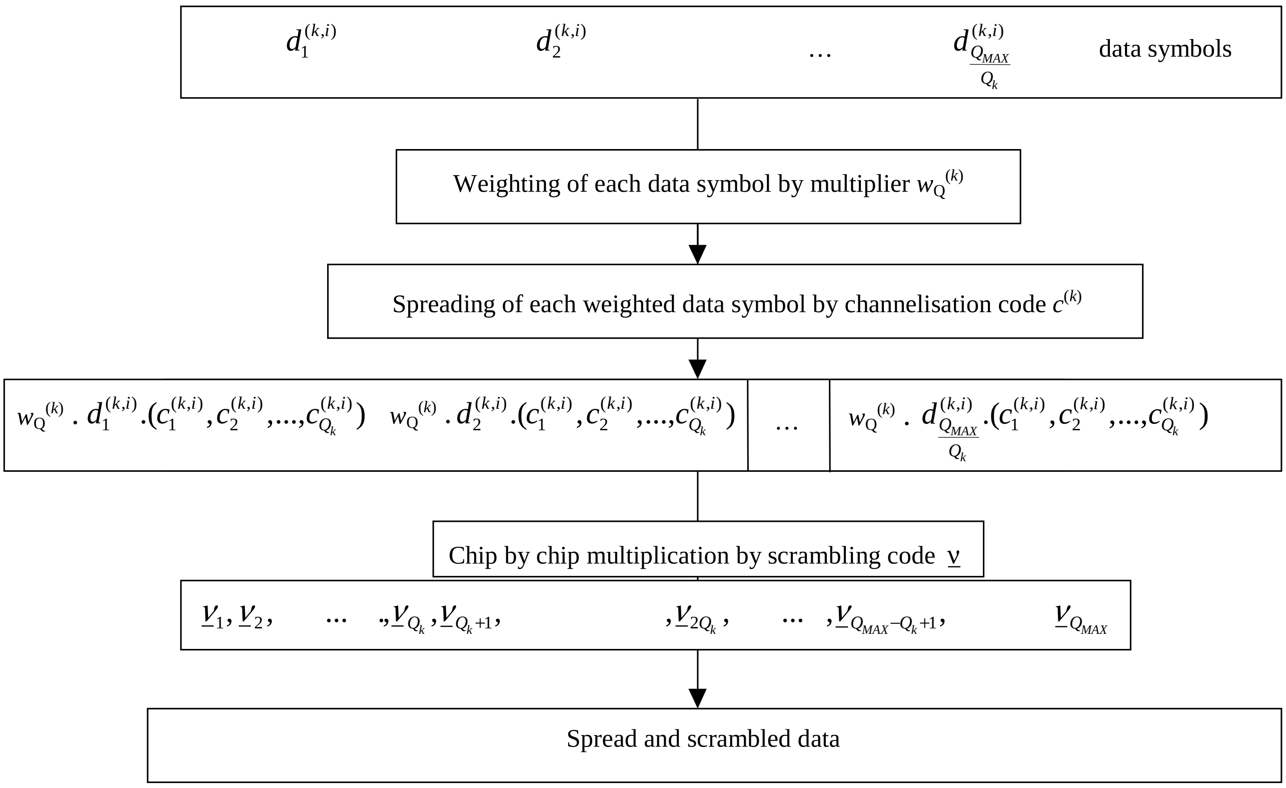
$$\underline{V}_{\underline{v}} = \{1, \mathrm{j}, -1, -\mathrm{j}\} \tag{4}$$

In equation 4 the letter j denotes the imaginary unit. A complex scrambling code $\underline{\mathbf{v}}$ is generated from the binary scrambling codes $\mathbf{v} = (v_1, v_2, \ldots, v_{16})$ of length 16 shown in Annex A. The relation between the elements $\underline{\mathbf{v}}$ and $\mathbf{v}$ is given by:

$$\underline{v}_i = (\mathrm{j})^i \cdot v_i \qquad v_i \in \{1, -1\},\ \mathrm{i} = 1,\ldots,16 \tag{5}$$

Hence, the elements $\underline{v}_i$ of the complex scrambling code $\underline{\mathbf{v}}$ are alternating real and imaginary.

The length matching is obtained by concatenating Q_{MAX}/Q_k spread words before the scrambling. The scheme is illustrated in figure 2 and is described in more detail in subclause 6.5.



$d_1^{(k,i)}$ $d_2^{(k,i)}$ … $d_{\frac{Q_{MAX}}{Q_k}}^{(k,i)}$ data symbols
Weighting of each data symbol by multiplier $w_Q^{(k)}$
Spreading of each weighted data symbol by channelisation code $c^{(k)}$
$w_Q^{(k)} \cdot d_1^{(k,i)}.(c_1^{(k,i)},c_2^{(k,i)},...,c_{Q_k}^{(k,i)})$ $w_Q^{(k)} \cdot d_2^{(k,i)}.(c_1^{(k,i)},c_2^{(k,i)},...,c_{Q_k}^{(k,i)})$ … $w_Q^{(k)} \cdot d_{\frac{Q_{MAX}}{Q_k}}^{(k,i)}.(c_1^{(k,i)},c_2^{(k,i)},...,c_{Q_k}^{(k,i)})$
Chip by chip multiplication by scrambling code $\underline{v}$
$\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_{Q_k}, \underline{v}_{Q_k+1}, \quad , \underline{v}_{2Q_k}, \ldots, \underline{v}_{Q_{MAX}-Q_k+1}, \quad \underline{v}_{Q_{MAX}}$
Spread and scrambled data

Figure 2: Spreading of data symbols

6.4a Scrambling codes for the 7.68Mcps option

The spreading of data by a real valued channelisation code $\mathbf{c}^{(k)}$ of length Q_k is followed by a cell specific complex scrambling sequence $\underline{v} = (\underline{v}_1, \underline{v}_2, ...\underline{v}_{32})$. The elements $\underline{v}_i; i = 1,...,32$ of the complex valued scrambling codes shall be taken from the complex set

$$\underline{V}_{\underline{v}} = \{1, j, -1, -j\} \tag{4a}$$

In equation 4a the letter j denotes the imaginary unit. A complex scrambling code $\underline{\mathbf{v}}$ is generated from the binary scrambling codes $v = (v_1, v_2, ...v_{32})$ of length 32 that are generated according to the method described in section 6.4a.1. The relation between the elements $\underline{\mathbf{v}}$ and $\mathbf{v}$ is given by:

$$\underline{v}_i = (j)^i \cdot v_i \qquad v_i \in \{1, -1\}, \quad i = 1,...,32 \tag{5a}$$

Hence, the elements $\underline{v}_i$ of the complex scrambling code $\underline{\mathbf{v}}$ are alternating real and imaginary.

The length matching is obtained by concatenating Q_{MAX}/Q_k spread words before the scrambling. The scheme is illustrated in figure 2 and is described in more detail in subclause 6.5.

6.4a.1 Generation of binary scrambling codes

The binary scrambling code, $c_{7.68}^n$, for cell parameter *n* in the 7.68Mcps TDD option is formed from the concatenation of the binary scrambling codes $c_{3.84}^n$ and $c_{3.84}^{(n+2)\bmod 128}$ shown in Annex A:

$$v = (v_1, v_2, ...v_{32}) = c_{7.68}^{n} = \left\{c_{3.84}^{n}, c_{3.84}^{(n+2)\bmod 128}\right\}$$

6.5 Spread signal of data symbols and data blocks

The combination of the user specific channelisation and cell specific scrambling codes can be seen as a user and cell specific spreading code $\mathbf{s}^{(k)} = \left(s_p^{(k)}\right)$ with

$$_{p}^{(k)} = c^{(k)}_{1+[(p-1)\bmod Q_k]} \cdot \underline{v}_{1+[(p-1)\bmod Q_{MAX}]}$$, k=1,…,K$_{\mathrm{Code}}$, p=1,…,N$_{\mathrm{k}}$Q$_{\mathrm{k}}$.

With the root raised cosine chip impulse filter Cr$_0$(t) the transmitted signal belonging to the data block $\underline{\mathbf{d}}^{(k,1)}$ of equation 1 transmitted before the midamble is

$$d^{(k,1)}(t) = \sum_{n=1}^{N_k} d_n^{(k,1)} w_{Q_k}^{(k)} \sum_{q=1}^{Q_k} s_{(n-1)Q_k+q}^{(k)} . Cr_0(t-(q-1)T_c-(n-1)Q_kT_c) \qquad (6)$$

and for the data block $\underline{\mathbf{d}}^{(k,2)}$ of equation 1 transmitted after the midamble

$$d^{(k,2)}(t) = \sum_{n=1}^{N_k} d_n^{(k,2)} w_{Q_k}^{(k)} \sum_{q=1}^{Q_k} s_{(n-1)Q_k+q}^{(k)} . Cr_0(t-(q-1)T_c-(n-1)Q_kT_c-N_kQ_kT_c-L_mT_c) \qquad (7)$$

where L_m is the number of midamble chips.

6.6 Modulation for the 3.84Mcps and 7.68Mcps options

The complex-valued chip sequence is modulated as shown in figure 3.

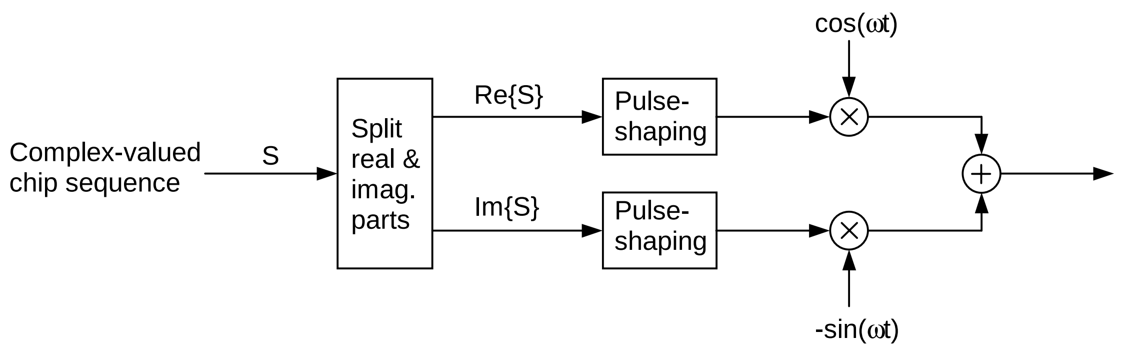


Figure 3: Modulation of complex valued chip sequences

The pulse-shaping characteristics are described in [9] and [10].

6.6.1 Combination of physical channels in uplink

Figure 4 illustrates the principle of combination of two different physical uplink channels within one timeslot. In the case of E-PUCH, only a single uplink physical channel is transmitted per timeslot and the procedures of subclause 6.6.1a shall instead apply).

The DPCHs to be combined belong to same CCTrCH, did undergo spreading as described in sections before and are thus represented by complex-valued sequences. First, the amplitude of all DPCHs is adjusted according to UL open loop power control as described in [10]. Each DPCH is then separately weighted by a weight factor γ_i and combined using complex addition. After combination of Physical Channels the gain factor β_j is applied, depending on the actual TFC as described in [10].

In case of different CCTrCH, principle shown in Figure 4 applies to each CCTrCH separately.

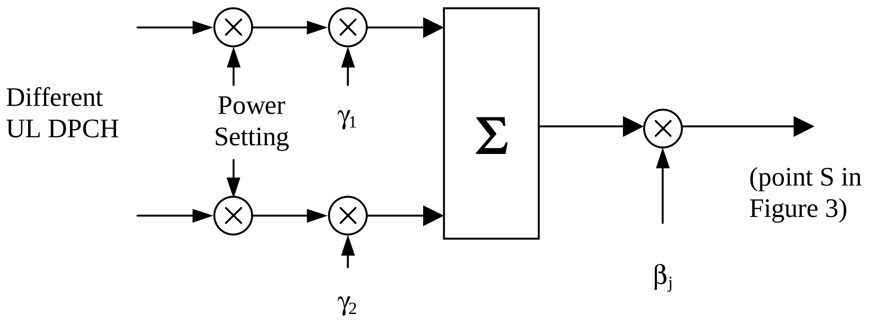


Figure 4: Combination of different physical channels in uplink

The values of weight factors γ_i are depending on the spreading factor SF of the corresponding DPCH:

SF of DPCH$_i$	γ_i
32	$\sqrt{2}/8$
16	$1/4$
8	$\sqrt{2}/4$
4	$1/2$
2	$\sqrt{2}/2$
1	1

NOTE: in the above table, SF = 32 is only supported in the 7.68Mcps TDD option.

In the case that β_j (corresponding to the *j*–th TFC) has been explicitly signalled to the UE, the possible values that β_j can assume are listed in the table below. In the case that β_j has been calculated by the UE from a reference TFC, β_j shall not be restricted to the quantised values.

Signalling value for β_j	Quantized value β_j
15	16/8
14	15/8
13	14/8
12	13/8
11	12/8
10	11/8
9	10/8
8	9/8
7	8/8
6	7/8
5	6/8
4	5/8
3	4/8
2	3/8
1	2/8
0	1/8

6.6.1a Physical channel transmission for E-PUCH

Figure 4a illustrates the principle of E-PUCH transmission. In a timeslot in which an E-PUCH is transmitted by a UE, no other physical channels may be transmitted by the same UE.

The amplitude of the E-PUCH is adjusted in accordance with the E-PUCH UL power control procedure described in [12]. The power setting procedure of [12] includes appropriate power adjustment factors for the E-PUCH spreading factor and for the E-TFC selected by higher layers [13]. Quantisation of the gain factor used to set the E-PUCH power is not specified.

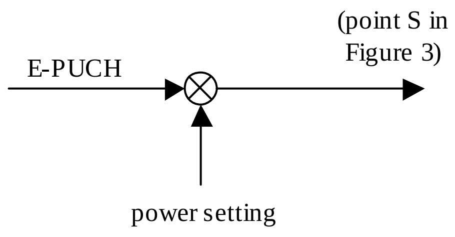


Figure 4a: Combination of different physical channels in uplink

6.6.2 Combination of physical channels in downlink

Figure 5 illustrates how different physical downlink channels are combined within one timeslot. Each complex-valued spread channel is separately weighted by a weight factor G_i. If a timeslot contains the SCH, the complex-valued SCH, as described in [7] is separately weighted by a weight factor G_{SCH}. All downlink physical channels are then combined using complex addition.

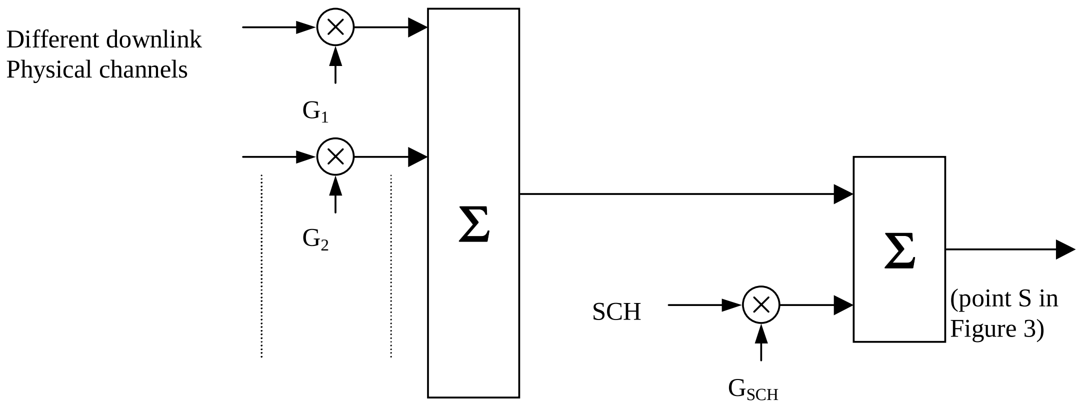


Figure 5: Combination of different physical channels in downlink in case of SCH timeslot

6.6.3 Combination of signature sequences for E-HICH

Multiple HARQ acknowledgement indicator signature sequences may be mapped onto the same channelisation code. Each signature sequence (described in [8]) is first subjected to QPSK modulation as described in subclause 5.2.1.1 to form the output sequence $d_{n,h}^{(k,i)}$ for the h^{th} indicator sequence, where $n=1,2,\ldots,N_k$ and $i=1,2$. Code k is the same value for all signature sequences mapped to the same channelisation code.

When multiple signature sequences are to be transmitted on the same channelisation code, the following procedure shall be applied prior to spreading.

Each QPSK-modulated stream $d_{n,h}^{(k,i)}$ is amplitude-weighted by a factor g_h according to the desired signature sequence power. A summation is then performed across all H signature sequences mapped to the same channelisation code as shown in figure 5a. The output of the summation block is the sequence:

$$e_n^{(k,i)} = \sum_{h=1}^{H} g_h d_{n,h}^{(k,i)} \quad (n = 1,2,\ldots,N_k) \text{ and } (i=1,2) \tag{8}$$

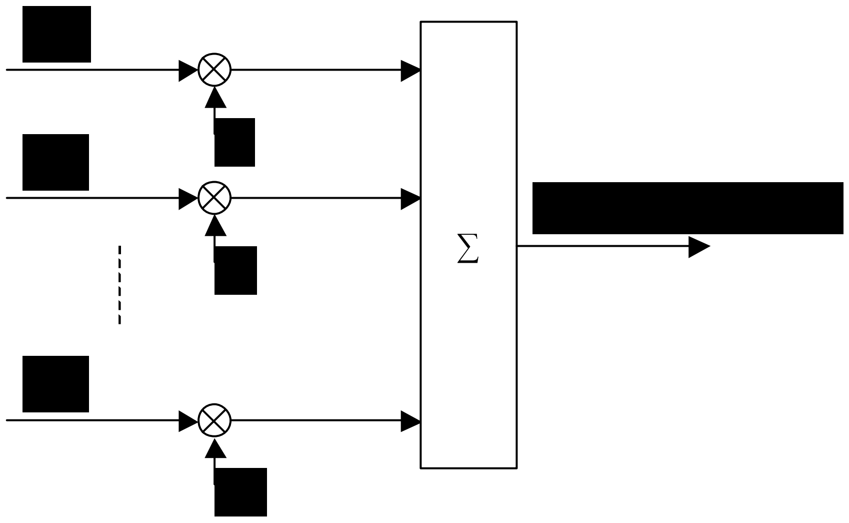


Figure 5a: Combination of HARQ acknowledgement indicator sequences prior to spreading

The sequence $e_n^{(k,i)}$ is mapped to a single channelisation code and subject to spreading at SF=16 (for 3.84Mcps) and at SF=32 (for 7.68Mcps) in accordance with the general method of subclause 6.

6.7 Modulation for the 1.28 Mcps option

The complex-valued chip sequence is modulated as shown in figure 6.

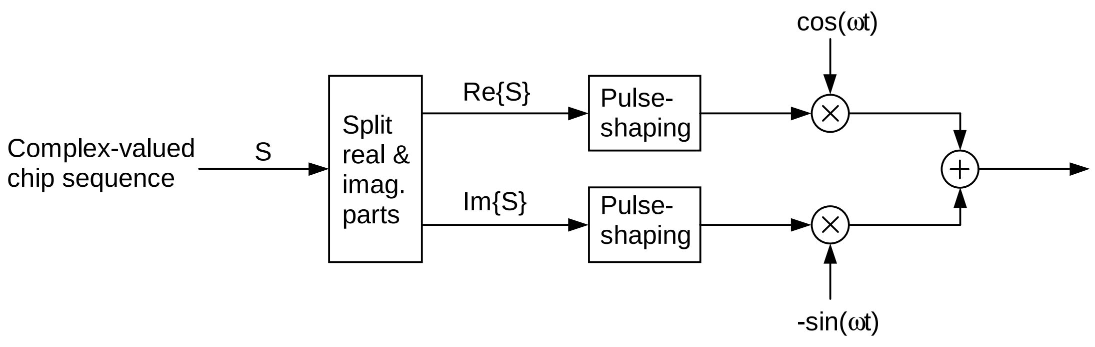


Figure 6: Modulation of complex valued chip sequences

The pulse-shaping characteristics are described in [9] and [10].

6.7.1 Combination of physical channels in uplink

The combination of physical channels in uplink is the same as in the 3.84 Mcps TDD cf. [6.6.1 Combination of physical channels in uplink] In the case of E-PUCH, only a single uplink physical channel is transmitted per timeslot and the procedures of subclause 6.7.1a shall instead apply).

6.7.1a Physical channel transmission for E-PUCH

The physical channel transmission for E-PUCH is the same as in the 3.84 Mcps TDD cf. [6.6.1a Physical channel transmission for E-PUCH]

6.7.2 Combination of physical channels in downlink

Figure 7 illustrates how different physical downlink channels are combined within one timeslot. Each spread channel is separately weighted by a weight factor G_i.. All downlink physical channels are then combined using complex addition.

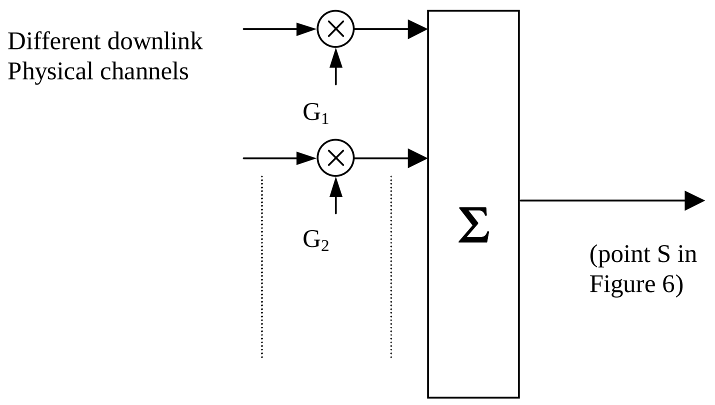


Figure 7: Combination of different physical channels in downlink

6.7.3 Combination of signature sequences for Scheduled E-HICH

For Scheduled E-HICH, every scheduled user is assigned one signature sequence which is related to the E-DCH resources allocated by Node-B to indicate ACK/NACK. Multiple users" HARQ acknowledgement indicator signature sequences may be mapped onto the same channelisation code. Each signature sequence (described in [8]) is first subjected to QPSK modulation as described in subclause 5.2.1.1 to form the output sequence $d_{n,h}^{(k,i)}$ for the h^{th} indicator sequence, where $n=1,2,\ldots,N_k$ and $i=1,2$. Code k is the same value for all signature sequences mapped to the same channelisation code.

When multiple signature sequences are to be transmitted on the same channelisation code, the following procedure shall be applied prior to spreading.

Each QPSK-modulated stream $d_{n,h}^{(k,i)}$ is amplitude-weighted by a factor g_h according to the desired signature sequence power. Each E-HICH physical channel may carry ACK/NACK signature sequence(s) for one UE or multiple UEs decided by Node-B. A summation is then performed across M signature sequences mapped to the same channelisation code as shown in figure 8. The output of the summation block is the sequence:

$$e_n^{(k,i)} = \sum_{h=1}^{M} g_h d_{n,h}^{(k,i)} \quad (n = 1,2,\ldots,N_k) \text{ and } (i=1,2) \qquad (9)$$

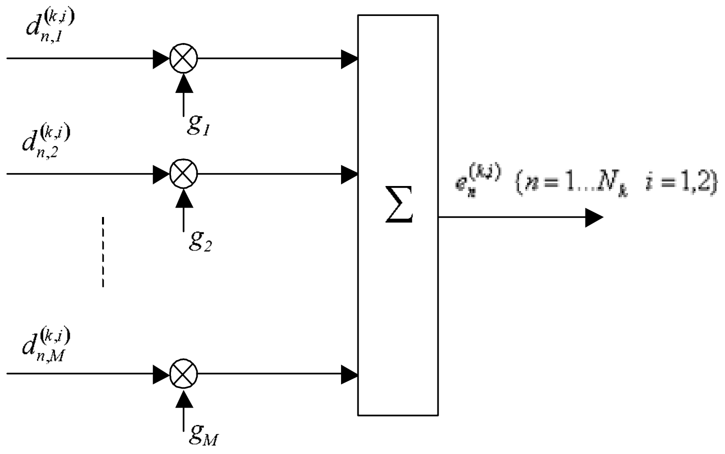


Figure 8: Combination of HARQ acknowledgement indicator sequences prior to spreading for Scheduled E-HICH

The sequence $e_n^{(k,i)}$ is mapped to a single channelisation code and subject to spreading at SF=16 in accordance with the general method of subclause 6.

6.7.3a Combination of signature sequences for Non-Scheduled E-HICH

For Non-Scheduled E-HICH, the 80 signature sequences are divided into 20 groups while each group includes 4 sequences. Every non-scheduled user is assigned one group by higher layer, from that two sequences are selected to indicate ACK/NACK and TPC/SS command. Multiple users" signature sequences may be mapped onto the same channelisation code. Each user"s two signature sequences (described in [8]) are first subjected to QPSK modulation as described in subclause 5.2.1.1 to form the two output sequences $d_{n,h_1}^{(k,i)}$ and $d_{n,h_2}^{(k,i)}$ for the h^{th} user, where $n=1,2,\ldots,N_k$ and $i=1,2$. Code k is the same value for all signature sequences mapped to the same channelisation code.

When multiple users" signature sequences are to be transmitted on the same channelisation code, the following procedure shall be applied prior to spreading.

Firstly, each user"s QPSK-modulated stream $d_{n,h_2}^{(k,i)}$ corresponding to TPC/SS signature sequence is amplitude-weighted by a factor f_h and added to the QPSK-modulated stream $d_{n,h_1}^{(k,i)}$ corresponding to ACK/NACK signature sequence; Secondly, each user"s combined stream $d_{n,h}^{(k,i)}$ is amplitude-weighted by a factor g_h according to the desired user power. A summation is then performed across M users" signature sequences mapped to the same channelisation code as shown in figure 8a. The output of the summation block is the sequence:

$$e_n^{(k,i)} = \sum_{h=1}^{M} g_h d_{n,h}^{(k,i)} \quad (n = 1,2,\ldots,N_k) \text{ and } (i=1,2) \tag{9a}$$

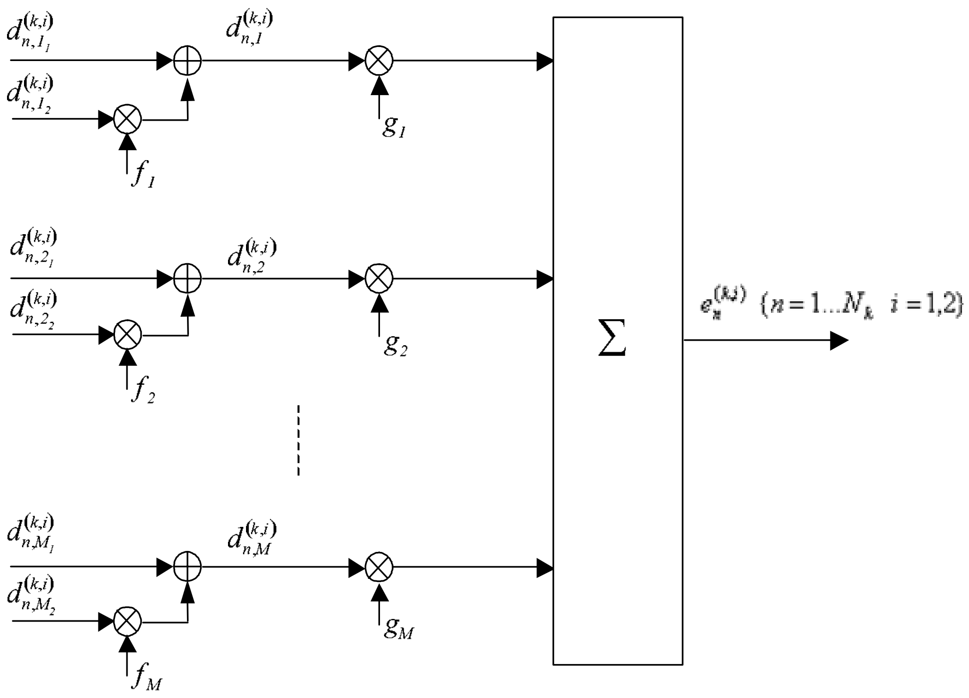


Figure 8a: Combination of ACK/NACK and TPC/SS sequences prior to spreading for Non-Scheduled E-HICH

The sequence $e_n^{(k,i)}$ is mapped to a single channelisation code and subject to spreading at SF=16 in accordance with the general method of subclause 6.

7 Synchronisation codes for the 3.84 Mcps option

7.1 Code Generation

The primary synchronisation code (PSC), C_p , is constructed as a so-called generalised hierarchical Golay sequence. The PSC is furthermore chosen to have good aperiodic auto correlation properties.

Define a = < x_1, x_2, x_3, …, x_{16} > = < 1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, 1, -1, -1, 1 >

The PSC is generated by repeating the sequence 'a' modulated by a Golay complementary sequence and creating a complex-valued sequence with identical real and imaginary components.

The PSC, C_p , is defined as C_p = < y(0),y(1),y(2),...,y(255) >

where $y = (1+\mathrm{j})\times < a, a, a, -a, -a, a, -a, -a, a, a, a, -a, a, -a, a, a >$

and the left most index corresponds to the chip transmitted first in time.

The 12 secondary synchronization codes, {C_0, C_1, C_3, C_4, C_5, C_6, C_8, C_{10}, C_{12}, C_{13}, C_{14},C_{15} } are complex valued with identical real and imaginary components, and are constructed from the position wise multiplication of a Hadamard sequence and a sequence z, defined as

z = $< b, b, b, -b, b, b, -b, -b, b, -b, b, -b, -b, -b, -b, -b >$, where

b = $< x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, -x_9, -x_{10}, -x_{11}, -x_{12}, -x_{13}, -x_{14}, -x_{15}, -x_{16} >$

and x_1, x_2, x_3, …, x_{16} are the same as in the definition of the sequence 'a' above.

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

$$H_0 = (1)$$
$$H_k = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix}, \quad k \geq 1$$

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

Denote the *n*:th Hadamard sequence h_n as a row of H_8 numbered from the top, n = 0, 1, 2, …, 255, in the sequel.

Furthermore, let $h_m(l)$ and $z(l)$ denote the *l*th symbol of the sequence h_m and *z*, respectively where l = 0, 1, 2, …, 255 and l = 0 corresponds to the leftmost symbol.

The i:th secondary SCH code word, C_i, i = 0, 1, 3, 4, 5, 6, 8, 10, 12, 13, 14, 15 is then defined as

$C_i = (1 + j) \times <h_m(0) \times z(0), h_m(1) \times z(1), h_m(2) \times z(2), \ldots, h_m(255) \times z(255)>$,

where $m = (16\times i)$ and the leftmost chip in the sequence corresponds to the chip transmitted first in time.

7.2 Code Allocation

Three secondary SCH codes are QPSK modulated and transmitted in parallel with the primary synchronization code. The QPSK modulation carries the following information:

- the code group that the base station belongs to (32 code groups:5 bits; Cases 1, 2);
- the position of the frame within an interleaving period of 20 msec (2 frames:1 bit, Cases 1, 2);
- the position of the SCH slot(s) within the frame (2 SCH slots:1 bit, Case 2).

The modulated secondary SCH codes are also constructed such that their cyclic-shifts are unique, i.e. a non-zero cyclic shift less than 2 (Case 1) and 4 (Case 2) of any of the sequences is not equivalent to some cyclic shift of any other of the sequences. Also, a non-zero cyclic shift less than 2 (Case 1) and 4 (Case 2) of any of the sequences is not equivalent to itself with any other cyclic shift less than 8. The secondary synchronization codes are partitioned into two code sets for Case 1 and four code sets for Case 2. The set is used to provide the following information:

Case 1:

Table 2: Code Set Allocation for Case 1

Code Set	Code Group
1	0-15
2	16-31

The code group and frame position information is provided by modulating the secondary codes in the code set.

Case 2:

Table 3: Code Set Allocation for Case 2

Code Set	Code Group
1	0-7
2	8-15
3	16-23
4	24-31

The slot timing and frame position information is provided by the comma free property of the code word and the Code group is provided by modulating some of the secondary codes in the code set.

The following SCH codes are allocated for each code set:

Case 1

Code set 1: C_1, C_3, $C_{5.}$

Code set 2: C_{10}, C_{13}, $C_{14.}$

Case 2

Code set 1: C_1, C_3, $C_{5.}$

Code set 2: C_{10}, C_{13}, $C_{14.}$

Code set 3: C_0, C_6, $C_{12.}$

Code set 4: C_4, C_8, $C_{15.}$

The following subclauses 7.2.1 to 7.2.2 refer to the two cases of SCH/P-CCPCH usage as described in [7].

Note that in the tables 4 and 5 corresponding to Cases 1 and 2, respectively, Frame 1 implies the frame with an odd SFN and Frame 2 implies the frame with an even SFN.

7.2.1 Code allocation for Case 1

Table 4: Code Allocation for Case 1

Code Group	Code Set	Frame 1			Frame 2			Associated t_{offset}
0	1	C_1	C_3	C_5	C_1	C_3	$-C_5$	t_0
1	1	C_1	$-C_3$	C_5	C_1	$-C_3$	$-C_5$	t_1
2	1	$-C_1$	C_3	C_5	$-C_1$	C_3	$-C_5$	t_2
3	1	$-C_1$	$-C_3$	C_5	$-C_1$	$-C_3$	$-C_5$	t_3
4	1	jC_1	jC_3	C_5	jC_1	jC_3	$-C_5$	t_4
5	1	jC_1	$-jC_3$	C_5	jC_1	$-jC_3$	$-C_5$	t_5
6	1	$-jC_1$	jC_3	C_5	$-jC_1$	jC_3	$-C_5$	t_6
7	1	$-jC_1$	$-jC_3$	C_5	$-jC_1$	$-jC_3$	$-C_5$	t_7
8	1	jC_1	jC_5	C_3	jC_1	jC_5	$-C_3$	t_8
9	1	jC_1	$-jC_5$	C_3	jC_1	$-jC_5$	$-C_3$	t_9
10	1	$-jC_1$	jC_5	C_3	$-jC_1$	jC_5	$-C_3$	t_{10}
11	1	$-jC_1$	$-jC_5$	C_3	$-jC_1$	$-jC_5$	$-C_3$	t_{11}
12	1	jC_3	jC_5	C_1	jC_3	jC_5	$-C_1$	t_{12}
13	1	jC_3	$-jC_5$	C_1	jC_3	$-jC_5$	$-C_1$	t_{13}
14	1	$-jC_3$	jC_5	C_1	$-jC_3$	jC_5	$-C_1$	t_{14}
15	1	$-jC_3$	$-jC_5$	C_1	$-jC_3$	$-jC_5$	$-C_1$	t_{15}
16	2	C_{10}	C_{13}	C_{14}	C_{10}	C_{13}	$-C_{14}$	t_{16}
17	2	C_{10}	$-C_{13}$	C_{14}	C_{10}	$-C_{13}$	$-C_{14}$	t_{17}
…	…	…	…	…	…	…	…	…
20	2	jC_{10}	jC_{13}	C_{14}	jC_{10}	jC_{13}	$-C_{14}$	t_{20}
…	…	…	…	…	…	…	…	…
24	2	jC_{10}	jC_{14}	C_{13}	jC_{10}	jC_{14}	$-C_{13}$	t_{24}
…	…	…	…	…	…	…	…	…
31	2	$-jC_{13}$	$-jC_{14}$	C_{10}	$-jC_{13}$	$-jC_{14}$	$-C_{10}$	t_{31}

NOTE: The code construction for code groups 0 to 15 using only the SCH codes from code set 1 is shown. The construction for code groups 16 to 31 using the SCH codes from code set 2 is done in the same way.

7.2.2 Code allocation for Case 2

Table 5: Code Allocation for Case 2

Code Group	Code Set	Frame 1						Frame 2						Associated t_{offset}
		Slot k			Slot k+8			Slot k			Slot k+8			
0	1	C_1	C_3	C_5	C_1	C_3	$-C_5$	$-C_1$	$-C_3$	C_5	$-C_1$	$-C_3$	$-C_5$	t_0
1	1	C_1	$-C_3$	C_5	C_1	$-C_3$	$-C_5$	$-C_1$	C_3	C_5	$-C_1$	C_3	$-C_5$	t_1
2	1	jC_1	jC_3	C_5	jC_1	jC_3	$-C_5$	$-jC_1$	$-jC_3$	C_5	$-jC_1$	$-jC_3$	$-C_5$	t_2
3	1	jC_1	$-jC_3$	C_5	jC_1	$-jC_3$	$-C_5$	$-jC_1$	jC_3	C_5	$-jC_1$	jC_3	$-C_5$	t_3
4	1	jC_1	jC_5	C_3	jC_1	jC_5	$-C_3$	$-jC_1$	$-jC_5$	C_3	$-jC_1$	$-jC_5$	$-C_3$	t_4
5	1	jC_1	$-jC_5$	C_3	jC_1	$-jC_5$	$-C_3$	$-jC_1$	jC_5	C_3	$-jC_1$	jC_5	$-C_3$	t_5
6	1	jC_3	jC_5	C_1	jC_3	jC_5	$-C_1$	$-jC_3$	$-jC_5$	C_1	$-jC_3$	$-jC_5$	$-C_1$	t_6
7	1	jC_3	$-jC_5$	C_1	jC_3	$-jC_5$	$-C_1$	$-jC_3$	jC_5	C_1	$-jC_3$	jC_5	$-C_1$	t_7
8	2	C_{10}	C_{13}	C_{14}	C_{10}	C_{13}	$-C_{14}$	$-C_{10}$	$-C_{13}$	C_{14}	$-C_{10}$	$-C_{13}$	$-C_{14}$	t_8
9	2	C_{10}	$-C_{13}$	C_{14}	C_{10}	$-C_{13}$	$-C_{14}$	$-C_{10}$	C_{13}	C_{14}	$-C_{10}$	C_{13}	$-C_{14}$	t_9
10	2	jC_{10}	jC_{13}	C_{14}	jC_{10}	jC_{13}	$-C_{14}$	$-jC_{10}$	$-jC_{13}$	C_{14}	$-jC_{10}$	$-jC_{13}$	$-C_{14}$	t_{10}
11	2	jC_{10}	$-jC_{13}$	C_{14}	jC_{10}	$-jC_{13}$	$-C_{14}$	$-jC_{10}$	jC_{13}	C_{14}	$-jC_{10}$	jC_{13}	$-C_{14}$	t_{11}
12	2	jC_{10}	jC_{14}	C_{13}	jC_{10}	jC_{14}	$-C_{13}$	$-jC_{10}$	$-jC_{14}$	C_{13}	$-jC_{10}$	$-jC_{14}$	$-C_{13}$	t_{12}
13	2	jC_{10}	$-jC_{14}$	C_{13}	jC_{10}	$-jC_{14}$	$-C_{13}$	$-jC_{10}$	jC_{14}	C_{13}	$-jC_{10}$	jC_{14}	$-C_{13}$	t_{13}
14	2	jC_{13}	jC_{14}	C_{10}	jC_{13}	jC_{14}	$-C_{10}$	$-jC_{13}$	$-jC_{14}$	C_{10}	$-jC_{13}$	$-jC_{14}$	$-C_{10}$	t_{14}
15	2	jC_{13}	$-jC_{14}$	C_{10}	jC_{13}	$-jC_{14}$	$-C_{10}$	$-jC_{13}$	jC_{14}	C_{10}	$-jC_{13}$	jC_{14}	$-C_{10}$	t_{15}
16	3	C_0	C_6	C_{12}	C_0	C_6	$-C_{12}$	$-C_0$	$-C_6$	C_{12}	$-C_0$	$-C_6$	$-C_{12}$	t_{16}
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
23	3	jC_6	$-jC_{12}$	C_0	jC_6	$-jC_{12}$	$-C_0$	$-jC_6$	jC_{12}	C_0	$-jC_6$	jC_{12}	$-C_0$	t_{20}
24	4	C_4	C_8	C_{15}	C_4	C_8	$-C_{15}$	$-C_4$	$-C_8$	C_{15}	$-C_4$	$-C_8$	$-C_{15}$	t_{24}
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
31	4	jC_8	$-jC_{15}$	C_4	jC_8	$-jC_{15}$	$-C_4$	$-jC_8$	jC_{15}	C_4	$-jC_8$	jC_{15}	$-C_4$	t_{31}

NOTE: The code construction for code groups 0 to 15 using the SCH codes from code sets 1 and 2 is shown. The construction for code groups 16 to 31 using the SCH codes from code sets 3 and 4 is done in the same way.

7.3 Evaluation of synchronisation codes

The evaluation of information transmitted in SCH on code group and frame timing is shown in table 6, where the 32 code groups are listed. Each code group is containing 4 specific scrambling codes (cf. subclause 6.4), each scrambling code associated with a specific short and long basic midamble code.

Each code group is additionally linked to a specific t_{Offset}, thus to a specific frame timing. By using this scheme, the UE can derive the position of the frame border due to the position of the SCH sequence and the knowledge of t_{Offset}. The complete mapping of Code Group to Scrambling Code, Midamble Codes and t_{Offset} is depicted in table 6.

Table 6: Mapping scheme for Cell Parameters, Code Groups, Scrambling Codes, Midambles and t_{Offset}

CELL PARA-METER	Code Group	Associated Codes			Associated t_{Offset}
		Scrambling Code	Long Basic Midamble Code	Short Basic Midamble Code	
0	Group 0	Code 0	m_{PL0}	m_{SL0}	t_0
1		Code 1	m_{PL1}	m_{SL1}	
2		Code 2	m_{PL2}	m_{SL2}	
3		Code 3	m_{PL3}	m_{SL3}	
4	Group 1	Code 4	m_{PL4}	m_{SL4}	t_1
5		Code 5	m_{PL5}	m_{SL5}	
6		Code 6	m_{PL6}	m_{SL6}	
7		Code 7	m_{PL7}	m_{SL7}	
. . . .					
124	Group 31	Code 124	m_{PL124}	m_{SL124}	t_{31}
125		Code 125	m_{PL125}	m_{SL125}	
126		Code 126	m_{PL126}	m_{SL126}	
127		Code 127	m_{PL127}	m_{SL127}	

For basic midamble codes m_P cf. [7], annex A 'Basic Midamble Codes'.

Each cell shall cycle through two sets of cell parameters in a code group with the cell parameters changing each frame. Table 7 shows how the cell parameters are cycled according to the SFN.

Table 7: Alignment of cell parameter cycling and SFN

Initial Cell Parameter Assignment	Code Group	Cell Parameter used when SFN mod 2 = 0	Cell Parameter used when SFN mod 2 = 1
0	Group 0	0	1
1		1	0
2		2	3
3		3	2
4	Group 1	4	5
5		5	4
6		6	7
7		7	6
. . . .			
124	Group 31	124	125
125		125	124
126		126	127
127		127	126

7A Synchronisation codes for the 7.68 Mcps option

7A.1 Code Generation

The primary synchronisation code (PSC), C_p , is constructed as a so-called generalised hierarchical Golay sequence. The PSC is furthermore chosen to have good aperiodic auto correlation properties.

Define a = < x_1, x_2, x_3, …, x_{16} > = < 1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, 1, -1, -1, 1 >

The PSC of length 512 chips is generated by repetition coding and repeating the sequence 'a' modulated by a Golay complementary sequence and creating a complex-valued sequence with identical real and imaginary components.

The PSC, C_p , is defined as C_p = < y(0),y(0),y(1),y(1),y(2),y(2)...,y(255),y(255) >

where $y = (1+\mathrm{j}) \times < a, a, a, -a, -a, a, -a, -a, a, a, a, -a, a, -a, a, a >$

and the left most index corresponds to the chip transmitted first in time.

The 12 secondary synchronization codes, {C_0, C_1, C_3, C_4, C_5, C_6, C_8, C_{10}, C_{12}, C_{13}, C_{14}, C_{15} } are complex valued with identical real and imaginary components, and are constructed from repetition coding of the position wise multiplication of a Hadamard sequence and a sequence z, defined as

z = $< b, b, b, -b, b, b, -b, -b, b, -b, b, -b, -b, -b, -b, -b >$, where

b = $< x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, -x_9, -x_{10}, -x_{11}, -x_{12}, -x_{13}, -x_{14}, -x_{15}, -x_{16} >$

and x_1, x_2, x_3, …, x_{16} are the same as in the definition of the sequence 'a' above.

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

$$H_0 = (1)$$
$$H_k = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix}, \quad k \geq 1$$

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

Denote the *n*:th Hadamard sequence h_n as a row of H_8 numbered from the top, n = 0, 1, 2, …, 255, in the sequel.

Furthermore, let $h_m(l)$ and $z(l)$ denote the *l*th symbol of the sequence h_m and *z*, respectively where l = 0, 1, 2, …, 255 and l = 0 corresponds to the leftmost symbol.

The i:th secondary SCH code word, C_i, i = 0, 1, 3, 4, 5, 6, 8, 10, 12, 13, 14, 15 is of length 512 chips and is then defined as

$C_i = (1 + j) \times <h_m(0) \times z(0), h_m(0) \times z(0), h_m(1) \times z(1), h_m(1) \times z(1), \ldots, h_m(255) \times z(255) , h_m(255) \times z(255)>$,

where $m = (16 \times \mathrm{i})$ and the leftmost chip in the sequence corresponds to the chip transmitted first in time.

7A.2 Code Allocation

Three secondary SCH codes are QPSK modulated and transmitted in parallel with the primary synchronization code. The QPSK modulation carries the following information:

- the code group that the base station belongs to (32 code groups:5 bits; Cases 1, 2);
- the position of the frame within an interleaving period of 20 msec (2 frames:1 bit, Cases 1, 2);
- the position of the SCH slot(s) within the frame (2 SCH slots:1 bit, Case 2).

The QPSK modulation sequences for the 7.68Mcps TDD option are unique to the modulation sequences for the 3.84Mcps TDD option.

The modulated secondary SCH codes are also constructed such that their cyclic-shifts are unique, i.e. a non-zero cyclic shift less than 2 (Case 1) and 4 (Case 2) of any of the sequences is not equivalent to some cyclic shift of any other of the sequences. Also, a non-zero cyclic shift less than 2 (Case 1) and 4 (Case 2) of any of the sequences is not equivalent to itself with any other cyclic shift less than 8. The secondary synchronization codes are partitioned into two code sets for Case 1 and four code sets for Case 2. The set is used to provide the following information:

Case 1:

Table 7A: Code Set Allocation for Case 1

Code Set	Code Group
1	0-15
2	16-31

The code group and frame position information is provided by modulating the secondary codes in the code set.

Case 2:

Table 7B: Code Set Allocation for Case 2

Code Set	Code Group
1	0-7
2	8-15
3	16-23
4	24-31

The slot timing and frame position information is provided by the comma free property of the code word and the Code group is provided by modulating some of the secondary codes in the code set.

The following SCH codes are allocated for each code set:

Case 1

Code set 1: C_1, C_3, $C_{5.}$

Code set 2: C_{10}, C_{13}, $C_{14.}$

Case 2

Code set 1: C_1, C_3, $C_{5.}$

Code set 2: C_{10}, C_{13}, $C_{14.}$

Code set 3: C_0, C_6, $C_{12.}$

Code set 4: C_4, C_8, $C_{15.}$

The following subclauses 7A.2.1 to 7A.2.2 refer to the two cases of SCH/P-CCPCH usage as described in [7].

Note that in the tables 7C and 7D corresponding to Cases 1 and 2, respectively, Frame 1 implies the frame with an odd SFN and Frame 2 implies the frame with an even SFN.

7A.2.1 Code allocation for Case 1

Table 7D: Code Allocation for Case 1

Code Group	Code Set	Frame 1			Frame 2			Associated t_{offset}
0	1	C_1	C_3	jC_5	C_1	C_3	$-jC_5$	t_0
1	1	C_1	$-C_3$	jC_5	C_1	$-C_3$	$-jC_5$	t_1
2	1	$-C_1$	C_3	jC_5	$-C_1$	C_3	$-jC_5$	t_2
3	1	$-C_1$	$-C_3$	jC_5	$-C_1$	$-C_3$	$-jC_5$	t_3
4	1	jC_1	jC_3	jC_5	jC_1	jC_3	$-jC_5$	t_4
5	1	jC_1	$-jC_3$	jC_5	jC_1	$-jC_3$	$-jC_5$	t_5
6	1	$-jC_1$	jC_3	jC_5	$-jC_1$	jC_3	$-jC_5$	t_6
7	1	$-jC_1$	$-jC_3$	jC_5	$-jC_1$	$-jC_3$	$-jC_5$	t_7
8	1	jC_1	C_5	C_3	jC_1	C_5	$-C_3$	t_8
9	1	jC_1	$-C_5$	C_3	jC_1	$-C_5$	$-C_3$	t_9
10	1	$-jC_1$	C_5	C_3	$-jC_1$	C_5	$-C_3$	t_{10}
11	1	$-jC_1$	$-C_5$	C_3	$-jC_1$	$-C_5$	$-C_3$	t_{11}
12	1	jC_3	C_5	C_1	jC_3	C_5	$-C_1$	t_{12}
13	1	jC_3	$-C_5$	C_1	jC_3	$-C_5$	$-C_1$	t_{13}
14	1	$-jC_3$	C_5	C_1	$-jC_3$	C_5	$-C_1$	t_{14}
15	1	$-jC_3$	$-C_5$	C_1	$-jC_3$	$-C_5$	$-C_1$	t_{15}
16	2	C_{10}	C_{13}	jC_{14}	C_{10}	C_{13}	$-jC_{14}$	t_{16}
17	2	C_{10}	$-C_{13}$	jC_{14}	C_{10}	$-C_{13}$	$-jC_{14}$	t_{17}
…	…	…	…	…	…	…	…	…
20	2	jC_{10}	jC_{13}	jC_{14}	jC_{10}	jC_{13}	$-jC_{14}$	t_{20}
…	…	…	…	…	…	…	…	…
24	2	jC_{10}	C_{14}	C_{13}	jC_{10}	C_{14}	$-C_{13}$	t_{24}
…	…	…	…	…	…	…	…	…
31	2	$-jC_{13}$	$-C_{14}$	C_{10}	$-jC_{13}$	$-C_{14}$	$-C_{10}$	t_{31}

NOTE: The code construction for code groups 0 to 15 using only the SCH codes from code set 1 is shown. The construction for code groups 16 to 31 using the SCH codes from code set 2 is done in the same way.

7A.2.2 Code allocation for Case 2

Table 7C: Code Allocation for Case 2

Code Group	Code Set	Frame 1						Frame 2						Associated t_{offset}
		Slot k			Slot k+8			Slot k			Slot k+8			
0	1	C_1	C_3	jC_5	C_1	C_3	$-jC_5$	$-C_1$	$-C_3$	jC_5	$-C_1$	$-C_3$	$-jC_5$	t_0
1	1	C_1	$-C_3$	jC_5	C_1	$-C_3$	$-jC_5$	$-C_1$	C_3	jC_5	$-C_1$	C_3	$-jC_5$	t_1
2	1	jC_1	jC_3	jC_5	jC_1	jC_3	$-jC_5$	$-jC_1$	$-jC_3$	jC_5	$-jC_1$	$-jC_3$	$-jC_5$	t_2
3	1	jC_1	$-jC_3$	jC_5	jC_1	$-jC_3$	$-jC_5$	$-jC_1$	jC_3	jC_5	$-jC_1$	jC_3	$-jC_5$	t_3
4	1	jC_1	C_5	C_3	jC_1	C_5	$-C_3$	$-jC_1$	$-C_5$	C_3	$-jC_1$	$-C_5$	$-C_3$	t_4
5	1	jC_1	$-C_5$	C_3	jC_1	$-C_5$	$-C_3$	$-jC_1$	C_5	C_3	$-jC_1$	C_5	$-C_3$	t_5
6	1	jC_3	C_5	C_1	jC_3	C_5	$-C_1$	$-jC_3$	$-C_5$	C_1	$-jC_3$	$-C_5$	$-C_1$	t_6
7	1	jC_3	$-C_5$	C_1	jC_3	$-C_5$	$-C_1$	$-jC_3$	C_5	C_1	$-jC_3$	C_5	$-C_1$	t_7
8	2	C_{10}	C_{13}	jC_{14}	C_{10}	C_{13}	$-jC_{14}$	$-C_{10}$	$-C_{13}$	jC_{14}	$-C_{10}$	$-C_{13}$	$-jC_{14}$	t_8
9	2	C_{10}	$-C_{13}$	jC_{14}	C_{10}	$-C_{13}$	$-jC_{14}$	$-C_{10}$	C_{13}	jC_{14}	$-C_{10}$	C_{13}	$-jC_{14}$	t_9
10	2	jC_{10}	jC_{13}	jC_{14}	jC_{10}	jC_{13}	$-jC_{14}$	$-jC_{10}$	$-jC_{13}$	jC_{14}	$-jC_{10}$	$-jC_{13}$	$-jC_{14}$	t_{10}
11	2	jC_{10}	$-jC_{13}$	jC_{14}	jC_{10}	$-jC_{13}$	$-jC_{14}$	$-jC_{10}$	jC_{13}	jC_{14}	$-jC_{10}$	jC_{13}	$-jC_{14}$	t_{11}
12	2	jC_{10}	C_{14}	C_{13}	jC_{10}	C_{14}	$-C_{13}$	$-jC_{10}$	$-C_{14}$	C_{13}	$-jC_{10}$	$-C_{14}$	$-C_{13}$	t_{12}
13	2	jC_{10}	$-C_{14}$	C_{13}	jC_{10}	$-C_{14}$	$-C_{13}$	$-jC_{10}$	C_{14}	C_{13}	$-jC_{10}$	C_{14}	$-C_{13}$	t_{13}
14	2	jC_{13}	C_{14}	C_{10}	jC_{13}	C_{14}	$-C_{10}$	$-jC_{13}$	$-C_{14}$	C_{10}	$-jC_{13}$	$-C_{14}$	$-C_{10}$	t_{14}
15	2	jC_{13}	$-C_{14}$	C_{10}	jC_{13}	$-C_{14}$	$-C_{10}$	$-jC_{13}$	C_{14}	C_{10}	$-jC_{13}$	C_{14}	$-C_{10}$	t_{15}
16	3	C_0	C_6	jC_{12}	C_0	C_6	$-jC_{12}$	$-C_0$	$-C_6$	jC_{12}	$-C_0$	$-C_6$	$-jC_{12}$	t_{16}
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
23	3	jC_6	$-C_{12}$	C_0	jC_6	$-C_{12}$	$-C_0$	$-jC_6$	C_{12}	C_0	$-jC_6$	C_{12}	$-C_0$	t_{20}
24	4	C_4	C_8	jC_{15}	C_4	C_8	$-jC_{15}$	$-C_4$	$-C_8$	jC_{15}	$-C_4$	$-C_8$	$-jC_{15}$	t_{24}
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
31	4	jC_8	$-C_{15}$	C_4	jC_8	$-C_{15}$	$-C_4$	$-jC_8$	C_{15}	C_4	$-jC_8$	C_{15}	$-C_4$	t_{31}

NOTE: The code construction for code groups 0 to 15 using the SCH codes from code sets 1 and 2 is shown. The construction for code groups 16 to 31 using the SCH codes from code sets 3 and 4 is done in the same way.

7A.3 Evaluation of synchronisation codes

The evaluation of information transmitted in SCH on code group and frame timing is shown in table 7E, where the 32 code groups are listed. Each code group contains 4 specific scrambling codes, each scrambling code associated with a specific short and long basic midamble code.

Each code group is additionally linked to a specific t_{Offset}, thus to a specific frame timing. By using this scheme, the UE can derive the position of the frame border due to the position of the SCH sequence and the knowledge of t_{Offset}. The complete mapping of Code Group to Scrambling Code, Midamble Codes and t_{Offset} is depicted in table 7E.

Table 7E: Mapping scheme for Cell Parameters, Code Groups, Scrambling Codes, Midambles and t_{Offset}

CELL PARA-METER	Code Group	Associated Codes			Associated t_{Offset}
		Scrambling Code	Long Basic Midamble Code	Short Basic Midamble Code	
0	Group 0	Code 0	m_{PL0}	m_{SL0}	t_0
1		Code 1	m_{PL1}	m_{SL1}	
2		Code 2	m_{PL2}	m_{SL2}	
3		Code 3	m_{PL3}	m_{SL3}	
4	Group 1	Code 4	m_{PL4}	m_{SL4}	t_1
5		Code 5	m_{PL5}	m_{SL5}	
6		Code 6	m_{PL6}	m_{SL6}	
7		Code 7	m_{PL7}	m_{SL7}	
. . . .					
124	Group 31	Code 124	m_{PL124}	m_{SL124}	t_{31}
125		Code 125	m_{PL125}	m_{SL125}	
126		Code 126	m_{PL126}	m_{SL126}	
127		Code 127	m_{PL127}	m_{SL127}	

Each cell shall cycle through two sets of cell parameters in a code group with the cell parameters changing each frame. Table 7F shows how the cell parameters are cycled according to the SFN.

Table 7F: Alignment of cell parameter cycling and SFN

Initial Cell Parameter Assignment	Code Group	Cell Parameter used when SFN mod 2 = 0	Cell Parameter used when SFN mod 2 = 1
0	Group 0	0	1
1		1	0
2		2	3
3		3	2
4	Group 1	4	5
5		5	4
6		6	7
7		7	6
. . . .			
124	Group 31	124	125
125		125	124
126		126	127
127		127	126

8 Synchronisation codes for the 1.28 Mcps option

8.1 The downlink pilot timeslot (DwPTS)

The contents of DwPTS is composed of 64 chips of a SYNC-DL sequence, cf.[AA.1 Basic SYNC-DL sequence] and 32 chips of guard period (GP). The SYNC-DL code is not scrambled

There should be 32 different basic SYNC-DL codes for the whole system.

For the generation of the complex valued SYNC-DL codes of length 64 , the basic binary SYNC-DL codes $=(s_1, s_2, \ldots, s_{64})$ of length 64 shown in Table AA.1 are used. The relation between the elements $\underline{\mathbf{s}}$ and $\mathbf{s}$ is given by:

$$\underline{s}_i = (j)^i \cdot s_i \qquad s_i \in \{1,-1\} \; i = 1,\ldots,64 \tag{1}$$

Hence, the elements $\underline{\mathbf{s}}_i$ of the complex SYNC-DL code $\underline{\mathbf{s}}$ are alternating real and imaginary.

The SYNC-DL is QPSK modulated and the phase of the SYNC-DL is used to signal the presence of the P-CCPCH in the multi-frame of the resource units of code $c_{Q=16}^{(k=1)}$ and $c_{Q=16}^{(k=2)}$ in time slot #0.

8.1.1 Modulation of the SYNC-DL

The SYNC-DL sequences are modulated with respect to the midamble ($m^{(1)}$) in time slot #0.

Four consecutive phases (phase quadruple) of the SYNC-DL are used to indicate the presence of the P-CCPCH in the following 4 sub-frames. In case the presence of a P-CCPCH is indicated, the next following sub-frame is the first sub-frame of the interleaving period. As QPSK is used for the modulation of the SYNC-DL, the phases 45, 135, 225, and 315° are used.

The total number of different phase quadruples is 2 (S1 and S2). A quadruple always starts with an even system frame number ((SFN mod 2) =0). Table 8 is showing the quadruples and their meaning.

Table 8: Sequences for the phase modulation for the SYNC-DL

Name	Phase quadruple	Meaning
S1	135, 45, 225, 135	There is a P-CCPCH in the next 4 sub-frames
S2	315, 225, 315, 45	There is no P-CCPCH in the next 4 sub-frames

8.2 The uplink pilot timeslot (UpPTS)

The contents in UpPTS is composed of 128 chips of a SYNC-UL sequence, cf. [AA.2 Basic SYNC-UL sequence] and 32chips of guard period (GP) .The SYNC-UL code is not scrambled.

There should be 256 different basic SYNC-UL codes (see Table AA.2) for the whole system.

For the generation of the complex valued SYNC-UL codes of length 128, the basic binary SYNC-UL codes $=(s_1, s_2, \ldots, s_{128})$ of length 128 shown in Table AA.2 are used. The relation between the elements $\underline{\mathbf{s}}$ and $\mathbf{s}$ is given by:

$$\underline{s}_i = (j)^i \cdot s_i \qquad s_i \in \{1,-1\} \; i = 1,\ldots,128 \tag{2}$$

Hence, the elements $\underline{\mathbf{s}}_i$ of the complex SYNC-UL code $\underline{\mathbf{s}}$ are alternating real and imaginary.

8.3 Code Allocation

Relationship between the SYNC-DL and SYNC-UL sequences, the scrambling codes and the midamble codes

Code Group	**Associated Codes**			
	SYNC-DL ID	SYNC-UL ID	Scrambling Code ID	Basic Midamble Code ID
Group 1	0	0...7	0	0
			1	1
			2	2
			3	3
Group 2	1	8...15	4	4
			5	5
			6	6
			7	7
. . .				
Group 32	31	248...255	124	124
			125	125
			126	126
			127	127

9 Cell synchronisation codes

The cell synchronisation codes (CSCs) are constructed as so-called CEC sequences, i.e. concatenated and periodically extended complementary sequences. They are complex-valued sequences that are derived as cyclically offset versions from a set of possible constituent Golay complementary pairs.

The CSCs are chosen to have good aperiodic auto correlation properties. The aperiodic auto correlations of the applicable constituent Golay complementary pairs and every pair of their derived cyclically offset versions are complementary. Furthermore, orthogonality is preserved for all CSCs which are derived from the same constituent Golay complementary pair due to this complementary property.

The delay and weight matrices for the set of M = 8 possible constituent Golay complementary pairs are listed in the table below:

Code ID *m*	**Delay matrices D_m and weight matrices W_m of constituent Golay complementary pairs**
0	D_0 = <512, 64, 128, 1, 16, 4, 256, 32, 8, 2>, W_0 = <1, 1, 1, 1, -1, -1, 1, 1, 1, 1>
1	D_1 = <2, 16, 32, 256, 1, 8, 128, 4, 512, 64>, W_1 = <1, -1, 1, -1, 1, -1, -1, 1, -1, -1>
2	D_2 = <16, 512, 32, 256, 4, 1, 64, 8, 2, 128>, W_2 = <-1, 1, 1, -1, -1, 1, -1, 1, -1, -1>
3	D_3 = <512, 16, 8, 4, 2, 256, 128, 64, 32, 1>, W_3 = <-1, -1, -1, -1, -1, 1, -1, 1, 1, 1>
4	D_4 = <512, 128, 256, 32, 2, 4, 64, 1, 16, 8>, W_4 = <1, -1, 1, -1, -1, -1, -1, -1, -1, 1>
5	D_5 = <1, 2, 4, 64, 512, 16, 32, 256, 128, 8>, W_5 = <-1, 1, 1, 1, 1, -1, -1, 1, -1, 1>
6	D_6 = <8, 16, 128, 2, 32, 1, 256, 512, 4, 64>, W_6 = <-1, -1, 1, 1, 1, 1, -1, -1, -1, 1>
7	D_7 = <1, 2, 128, 16, 256, 32, 8, 512, 64, 4>, W_7 = <1, 1, -1, -1, -1, -1, 1, -1, -1, -1>

A constituent Golay complementary pair of length N = 1024, defined as:

$s_m = \langle s_m(0), s_m(1), s_m(2), \ldots, s_m(1023)\rangle$ and $g_m = \langle g_m(0), g_m(1), g_m(2), \ldots, g_m(1023)\rangle$

shall be derived from the selected delay and weight matrices:

$D_m = \langle D_m(0), D_m(1), D_m(2), \ldots, D_m(9)\rangle$ and $W_m = \langle W_m(0), W_m(1), W_m(2), \ldots, W_m(9)\rangle$

as follows.

Define:

$a^{(0)} = \langle a^{(0)}(0), a^{(0)}(1), a^{(0)}(2), \ldots, a^{(0)}(1023)\rangle = \langle 1, 0, 0, \ldots, 0\rangle$ and

$b^{(0)} = \langle b^{(0)}(0), b^{(0)}(1), b^{(0)}(2), \ldots, b^{(0)}(1023)\rangle = \langle 1, 0, 0, \ldots, 0\rangle$.

Then, the elements of the set of auxiliary sequences:

$a^{(n)} = \langle a^{(n)}(0), a^{(n)}(1), a^{(n)}(2), \ldots, a^{(n)}(1023)\rangle$ and $b^{(n)} = \langle b^{(n)}(0), b^{(n)}(1), b^{(n)}(2), \ldots, b^{(n)}(1023)\rangle$

are given by the recursive relations:

$a^{(n+1)}(i) = a^{(n)}(i) + W_m(n) \times b^{(n)}(i - D_m(n))$ and

$b^{(n+1)}(i) = a^{(n)}(i) - W_m(n) \times b^{(n)}(i - D_m(n))$

with element index $i = 0, 1, 2, \ldots, 1023$ and iteration index $n = 0, 1, 2, \ldots, 9$. Operations on the element index shall be performed modulo 1024.

The elements of the constituent Golay complementary pairs s_m and g_m are then obtained from the output of the last iteration step using:

$s_m(i) = a^{(10)}(i)$ and $g_m(i) = b^{(10)}(i)$ for $i = 0, 1, 2, ..., 1023$

From each applicable constituent Golay complementary pair s_m and g_m, up to K = 8 different cyclically offset pairs $s_m^{(k)}$ and $g_m^{(k)}$, with offset index $k = 0, 1, 2, \ldots, K-1$, of length 1152 chips can be derived. The complementary property of the respective aperiodic auto correlation is preserved for each particular pair of sequences $s_m^{(k)}$ and $g_m^{(k)}$. The generation of the K cyclically offset pairs from s_m and g_m is done in a similar way as the generation of the user midambles from a periodic basic midamble sequence as described in [7].

With N = 1024, K = 8, W = 128, the elements of a cyclically offset pair:

$s_m^{(k)} = \langle s_m^{(k)}(0), s_m^{(k)}(1), s_m^{(k)}(2), \ldots, s_m^{(k)}(1151)\rangle$ and $g_m^{(k)} = \langle g_m^{(k)}(0), g_m^{(k)}(1), g_m^{(k)}(2), \ldots, g_m^{(k)}(1151)\rangle$

for a particular offset k, with $k = 0, 1, 2, \ldots, K-1$, shall be derived from the elements of the constituent Golay complementary pairs s_m and g_m using:

$s_m^{(k)}(i) = (j)^i \times s_m(i + k \times W)$ and $g_m^{(k)}(i) = (j)^i \times g_m(i + k \times W)$ for $i = 0, 1, 2, ..., N - k \times W - 1$,

$s_m^{(k)}(i) = (j)^i \times s_m(i - N + k \times W)$ and $g_m^{(k)}(i) = (j)^i \times g_m(i - N + k \times W)$ for $i = N - k \times W, N - k \times W + 1, ..., 1151$.

Hence, the elements of $s_m^{(k)}$ and $g_m^{(k)}$ are alternating real and imaginary.

Note that both $s_m^{(0)}$ and $g_m^{(0)}$ simply correspond to s_m and g_m respectively, followed by its first W elements as post extension and that both $s_m^{(7)}$ and $g_m^{(7)}$ simply correspond to the last W elements of s_m and g_m in form of a pre extension, followed by s_m and g_m respectively.

Finally, the CSC $C_{CSC,m}^{(k)}$ derived from the m:th applicable constituent Golay complementary pair s_m and g_m, and for the k:th offset is then defined as a concatenation of $s_m^{(k)}$ and $g_m^{(k)}$ by:

$C_{CSC,m}^{(k)} = \langle s_m^{(k)}(0), s_m^{(k)}(1), s_m^{(k)}(2), \ldots, s_m^{(k)}(1151), g_m^{(k)}(0), g_m^{(k)}(1), g_m^{(k)}(2), \ldots, g_m^{(k)}(1151)\rangle$

where the leftmost element $s_m^{(k)}(0)$ in the sequence corresponds to the chip to be first transmitted in time. An CSC has therefore length 2304 chips.

Note that due to this construction method, the auto correlations for all CSCs derived from one particular constituent Golay complementary pair s_m and g_m can be obtained simultaneously and in sequential order from the sum of partial correlations with s_m and g_m, these CSCs remaining orthogonal.

CSCs derived according to above have complex values and shall not be subject to the channelisation or scrambling process, i.e. its elements represent complex chips for usage in the pulse shaping process at modulation.

Annex A (normative): Scrambling Codes

The applicable scrambling codes are listed below. Code numbers are referring to table 6 'Mapping scheme for Cell Parameters, Code Groups, Scrambling Codes, Midambles and t_{offset}' in subclause 6.3 'Evaluation of synchronisation codes'.

Scrambling Code	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}
Code 0	-1	1	-1	-1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1
Code 1	1	1	1	1	1	-1	1	-1	1	-1	-1	1	1	1	-1	-1
Code 2	1	-1	1	1	1	-1	1	1	-1	1	1	1	1	-1	-1	-1
Code 3	1	1	1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	1	-1
Code 4	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	1	1	1	-1
Code 5	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1	1	-1
Code 6	-1	1	-1	-1	-1	1	-1	-1	-1	1	1	1	1	-1	-1	-1
Code 7	1	-1	1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	1	1	-1
Code 8	1	1	1	-1	-1	-1	1	-1	1	1	-1	1	1	1	1	-1
Code 9	1	1	-1	1	1	1	1	-1	1	1	1	-1	-1	-1	1	-1
Code 10	1	-1	1	-1	1	1	1	1	-1	-1	1	1	-1	1	1	-1
Code 11	-1	1	1	1	1	-1	-1	-1	-1	1	-1	-1	-1	1	-1	-1
Code 12	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1	1	1	1	1	-1
Code 13	1	-1	1	1	1	-1	-1	-1	1	-1	-1	-1	-1	1	-1	-1
Code 14	1	-1	-1	-1	-1	1	-1	-1	1	-1	1	1	1	-1	-1	-1
Code 15	1	1	-1	-1	-1	1	1	-1	1	-1	1	-1	-1	-1	-1	-1
Code 16	1	-1	-1	1	-1	1	-1	1	-1	-1	-1	-1	1	1	-1	-1
Code 17	1	1	1	-1	1	1	1	-1	1	1	-1	1	-1	-1	1	-1
Code 18	-1	1	1	1	-1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-1
Code 19	-1	1	-1	-1	1	-1	-1	-1	-1	1	1	1	-1	1	-1	-1
Code 20	-1	-1	-1	-1	1	-1	1	-1	-1	1	1	-1	1	1	-1	-1
Code 21	1	1	1	1	-1	-1	1	1	-1	1	1	-1	1	-1	1	-1
Code 22	1	-1	-1	-1	-1	1	1	1	-1	1	-1	-1	-1	1	-1	-1
Code 23	-1	1	1	1	-1	1	1	1	1	-1	1	1	-1	1	-1	-1
Code 24	-1	-1	1	-1	1	1	1	-1	-1	-1	-1	1	-1	-1	1	-1
Code 25	1	-1	1	1	1	-1	1	1	1	-1	1	1	-1	1	-1	-1
Code 26	1	-1	-1	-1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1
Code 27	-1	1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	-1	-1
Code 28	-1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	1	1	1	1	-1
Code 29	1	-1	1	1	-1	1	-1	-1	-1	1	-1	-1	-1	1	-1	-1
Code 30	-1	-1	-1	-1	-1	-1	1	1	1	-1	-1	1	1	-1	1	-1
Code 31	1	1	-1	-1	1	1	1	1	-1	1	-1	1	-1	1	1	-1
Code 32	1	-1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1
Code 33	-1	-1	-1	1	1	1	1	-1	1	1	1	-1	1	1	1	-1
Code 34	1	-1	-1	-1	1	-1	-1	-1	1	-1	1	1	-1	1	-1	-1
Code 35	1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1
Code 36	1	1	-1	1	1	1	-1	1	-1	-1	-1	1	1	1	1	-1
Code 37	-1	-1	-1	1	-1	-1	1	-1	-1	-1	1	-1	1	1	1	-1
Code 38	-1	1	-1	-1	1	-1	1	1	1	-1	-1	-1	1	-1	-1	-1
Code 39	-1	1	1	1	1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	-1
Code 40	-1	1	-1	1	-1	-1	-1	-1	-1	-1	1	1	-1	1	1	-1
Code 41	1	1	-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	1	-1
Code 42	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	1	-1	-1	-1
Code 43	-1	-1	1	1	-1	-1	-1	-1	-1	1	-1	1	-1	1	1	-1

Scrambling Code	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}
Code 44	-1	-1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	1	-1
Code 45	-1	-1	1	-1	1	1	-1	1	1	1	1	-1	1	1	1	-1
Code 46	-1	1	1	-1	1	1	-1	-1	-1	-1	-1	-1	1	-1	1	-1
Code 47	1	-1	-1	1	1	1	-1	-1	1	1	1	1	1	-1	1	-1
Code 48	1	1	-1	1	1	1	-1	1	1	1	-1	1	-1	-1	1	-1
Code 49	-1	-1	1	1	-1	1	1	-1	-1	1	-1	1	-1	-1	-1	-1
Code 50	1	1	-1	1	-1	-1	1	-1	1	1	1	-1	1	1	1	-1
Code 51	1	-1	-1	1	1	1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
Code 52	1	1	1	-1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1
Code 53	-1	1	1	1	-1	-1	-1	1	-1	1	1	1	1	1	1	-1
Code 54	-1	-1	1	-1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1
Code 55	-1	1	1	-1	-1	-1	-1	-1	1	-1	1	-1	1	1	-1	-1
Code 56	-1	1	1	1	-1	1	1	1	-1	1	1	1	1	-1	-1	-1
Code 57	-1	1	1	-1	-1	-1	1	1	-1	1	-1	1	-1	-1	-1	-1
Code 58	-1	1	-1	1	-1	-1	-1	-1	-1	1	1	-1	1	1	-1	-1
Code 59	1	1	-1	-1	-1	-1	-1	-1	1	-1	1	-1	-1	1	1	-1
Code 60	-1	1	1	-1	1	1	1	1	-1	1	-1	1	1	1	-1	-1
Code 61	-1	-1	1	1	1	-1	-1	1	1	-1	1	-1	-1	-1	-1	-1
Code 62	-1	1	-1	-1	1	1	1	-1	1	-1	-1	-1	-1	-1	1	-1
Code 63	-1	1	-1	1	-1	-1	1	1	1	-1	-1	1	-1	-1	-1	-1
Code 64	1	-1	-1	1	-1	-1	1	1	-1	-1	-1	-1	1	-1	1	-1
Code 65	-1	-1	-1	1	1	1	1	-1	-1	-1	1	-1	-1	-1	1	-1
Code 66	-1	-1	-1	-1	1	-1	-1	1	1	1	-1	-1	1	-1	1	-1
Code 67	-1	-1	-1	1	1	1	-1	1	1	1	-1	1	1	1	1	-1
Code 68	1	-1	1	1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	1	-1
Code 69	-1	-1	1	-1	1	-1	-1	-1	1	1	1	-1	-1	1	-1	-1
Code 70	1	1	-1	1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	1	-1
Code 71	1	-1	-1	1	-1	-1	-1	-1	-1	1	-1	1	1	1	-1	-1
Code 72	1	1	1	1	-1	1	1	-1	1	1	-1	-1	1	-1	1	-1
Code 73	-1	1	1	1	-1	-1	-1	1	-1	1	-1	-1	-1	-1	1	-1
Code 74	1	1	-1	1	-1	1	-1	-1	-1	-1	-1	1	1	-1	-1	-1
Code 75	1	1	-1	-1	1	-1	-1	1	-1	1	-1	1	-1	-1	-1	-1
Code 76	-1	1	-1	-1	-1	-1	-1	1	-1	1	1	1	-1	-1	1	-1
Code 77	-1	1	-1	1	1	1	1	1	-1	1	1	-1	1	1	-1	-1
Code 78	-1	1	-1	1	-1	1	1	-1	-1	-1	1	1	-1	-1	-1	-1
Code 79	-1	1	-1	1	1	1	-1	-1	-1	1	1	-1	-1	-1	-1	-1
Code 80	1	1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	1	-1	-1
Code 81	1	1	1	1	1	-1	1	-1	-1	-1	1	1	-1	1	1	-1
Code 82	-1	1	-1	1	1	1	1	1	1	1	-1	-1	-1	1	1	-1
Code 83	1	1	-1	-1	1	-1	1	-1	-1	-1	-1	-1	-1	1	1	-1
Code 84	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1	1	-1
Code 85	-1	1	1	-1	-1	1	-1	1	1	1	1	1	1	1	-1	-1
Code 86	-1	-1	-1	1	-1	-1	-1	1	1	1	-1	1	-1	-1	1	-1
Code 87	1	1	-1	-1	-1	1	-1	1	1	1	1	1	-1	1	1	-1
Code 88	-1	1	1	-1	1	1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
Code 89	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	-1	1	-1	-1
Code 90	1	-1	-1	-1	-1	-1	-1	1	1	-1	1	1	-1	-1	1	-1
Code 91	-1	1	-1	-1	-1	-1	1	-1	1	-1	1	1	-1	-1	1	-1
Code 92	-1	1	1	-1	1	-1	1	-1	-1	-1	-1	-1	1	1	-1	-1
Code 93	-1	-1	-1	-1	-1	1	1	-1	-1	-1	1	1	1	-1	1	-1
Code 94	1	-1	1	-1	-1	1	1	-1	1	1	-1	-1	-1	-1	-1	-1
Code 95	1	1	1	1	1	-1	-1	1	-1	-1	1	1	1	-1	1	-1
Code 96	1	1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	1	-1	1	-1

Scrambling Code	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}
Code 97	1	1	-1	-1	1	-1	-1	1	1	1	1	1	1	-1	1	-1
Code 98	1	1	-1	1	1	-1	1	1	1	1	1	-1	1	-1	-1	-1
Code 99	1	-1	1	-1	1	-1	-1	1	-1	-1	1	1	-1	-1	-1	-1
Code 100	1	-1	1	1	-1	-1	1	-1	-1	1	-1	-1	-1	-1	1	-1
Code 101	1	1	1	1	-1	1	-1	1	1	1	-1	-1	-1	1	1	-1
Code 102	1	-1	1	-1	1	1	1	1	-1	1	1	-1	1	1	-1	-1
Code 103	-1	-1	1	-1	-1	1	-1	-1	1	1	1	-1	1	-1	-1	-1
Code 104	1	-1	1	1	-1	1	1	1	-1	1	1	1	-1	1	-1	-1
Code 105	1	1	1	1	1	1	-1	-1	1	-1	-1	1	1	-1	1	-1
Code 106	1	1	-1	-1	-1	1	-1	1	-1	-1	-1	-1	-1	1	1	-1
Code 107	-1	-1	-1	-1	1	1	-1	-1	-1	1	1	-1	1	-1	1	-1
Code 108	-1	-1	-1	1	-1	1	-1	-1	1	1	-1	1	1	-1	-1	-1
Code 109	-1	1	-1	1	1	-1	-1	1	1	1	-1	-1	-1	-1	-1	-1
Code 110	-1	-1	1	1	-1	1	-1	1	1	1	1	1	-1	1	1	-1
Code 111	1	1	1	-1	-1	1	1	1	1	1	-1	1	-1	1	-1	-1
Code 112	-1	-1	1	1	1	-1	1	-1	1	1	1	1	-1	1	1	-1
Code 113	1	1	-1	-1	1	-1	1	-1	1	1	1	1	-1	1	1	-1
Code 114	-1	-1	-1	1	1	-1	-1	-1	1	1	-1	1	-1	1	-1	-1
Code 115	1	-1	-1	1	1	1	1	1	1	-1	1	-1	1	1	-1	-1
Code 116	-1	1	1	1	1	-1	1	1	1	-1	1	1	1	-1	-1	-1
Code 117	1	1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	1	-1
Code 118	-1	-1	-1	-1	-1	1	-1	1	1	-1	-1	1	1	1	-1	-1
Code 119	-1	-1	-1	1	-1	1	1	1	-1	-1	1	-1	-1	1	-1	-1
Code 120	-1	-1	1	-1	1	-1	1	1	-1	-1	1	-1	-1	1	-1	-1
Code 121	-1	1	1	1	1	1	1	-1	1	-1	1	1	-1	-1	1	-1
Code 122	-1	-1	-1	1	1	-1	1	1	-1	-1	1	-1	1	-1	-1	-1
Code 123	1	-1	1	-1	1	1	-1	-1	1	-1	-1	1	-1	-1	-1	-1
Code 124	-1	-1	1	1	1	1	1	1	1	-1	1	-1	-1	1	1	-1
Code 125	1	-1	-1	1	1	-1	1	-1	1	1	1	1	1	1	-1	-1
Code 126	1	1	1	1	-1	1	-1	1	-1	1	1	-1	1	1	-1	-1
Code 127	1	-1	1	-1	-1	-1	-1	-1	1	-1	-1	1	1	1	-1	-1

Annex AA (normative): Synchronisation sequence

AA.1 Basic SYNC-DL sequence

Table AA.1: Basic SYNC-DL Codes

Code ID	SYNC-DL Codes of length 64
0	B3A7CC05A98688E4
1	9D559BD290606791
2	2CE7BA12A017C3A2
3	34511D20672F4712
4	9A772841474603F2
5	9109B1A5CE01F228
6	8FD429B3594501C0
7	25251354AA3F8C19
8	C9A3B8E0C043EA56
9	BA04B888E5BC1802
10	A735354299370207
11	74C3C8DA4415AE51
12	F4FD0458A0124663
13	A011D4E16C3D6064
14	BDA0661B0CAA8C68
15	8E31123F28928698
16	F095C1632E2906AB
17	B60B4A8A664071CF
18	AA094DCCE91E041A
19	C0C31CDA8A256807
20	D516964FB18C1890
21	30DE01834F4AACCE
22	8F700323BA5CAD34
23	1B50F4DEE0C1380C
24	443382164F56F2D1
25	E1E4005D49B846B4
26	040A97165330BFAA
27	C48E26881693AD78
28	D4354B2FE02361CC
29	5383AB6C8A10CE84
30	D417A730F2F12244
31	ABF0A0D905A939C4

AA.2 Basic SYNC-UL Codes

Table AA.2: Basic SYNC-UL Codes

Code ID	SYNC-UL Codes of length 128
0	C11C20F0D1807DB8859175B798EC094A
1	91278068081EC8E74543DBC1C9AD4235
2	38F5AEE2E513DB12A663BA04160103E5
3	7AA8A0A210F12A1E4332F2EDD33011FC
4	C180EA3B9BA1774EB9611BD249C4A508
5	B072A2C839489D496B98CE9D0132FBC9
6	B2723EAC6EB01667F2B33961C8074234
7	C4144AD060F0EC095E227B92CF7C8280
8	653036A10D3054146FCF815986C63A14
9	F899CA61435D64DC07FDF04C4A0C053A
10	B56F2D6893A8051407F4C341D88DC7DC
11	DC0BE838242142EDE6413A72C88D74AA
12	22A2FD86E4086C70A4860B13C76E579F
13	A3CBC21322C97D2A02728E7875F39588
14	D4EC4F694A082CB38E3B1558A0FCC89F
15	CC891141C4E216D235C15CF5D3F9B002
16	A1993114C50B77CB0C0725D1E22FD016
17	24F73A979DE52F82E8800CCB93842A59
18	8F878FA04659842E294D8DEAB20BA2FD
19	AC90B0442D70662B028CF76A6BECDF09
20	D94A284DF64D7B0102F0E084C29C88C8
21	8603200C7596F24E865FD3815693358D
22	B466B12CF433642BD8B08F1F452E0550
23	86A3A1772C1C99FCA7DBBA0C312E34A0
24	622A1889F72A9A2C042D46F08EFEE1AC
25	BF220A362BC0D3B0D7CE400954C6CFAE
26	D28D73C52E89CF57905C502244F63616
27	AD4E1C2103697D64D8B9D4C035D90548
28	8F081A9BA12B6C6BD024531AA984D21C
29	E4092429BE82988E1E3585BF6A6AE550
30	08BD36E0A9C061782CB38B35B335CA56
31	1CDFF3CC2685D1C44F4A1059AB03F40A
32	506ED4E88FB1CECE3243F2A27A0221A4
33	846CF58A7AB613C83A24130B5778C0E2
34	A2711A99E26A0C75AC026F4CFAECE893
35	D846EEEBA2432AC05A01043C62579DCF
36	6B16B4E851CAF2121FC4CF88820C89E7
37	AA4889A78207674A74E10C6F2BE11D48
38	8534CF8145BC991052814ED5C72709EE
39	01AEF15D2290A84A607425746D9963C7

40	999188F758245D5164FE16D852942C71
41	CF71C008599287E446E30745BD56E2D2
42	248414BA0DF8CDC4711FE7C8707ED0AD
43	EB2E263EC016191C81AB714BFE4D2B30
44	862082A7482FAC1C499793A0D8CED670
45	DE2C22B2783AB75A7342608DE413840A
46	E31AA60B727F2CA2A78DAAC10665011D
47	CEF6CD06509870AC9E0177ACD550921D
48	E52C84D499FFCDC287581691471540F2
49	B33BF6551A4322504BEE0930BCA1EC68
50	555BE6886D0FC43D72315E6C6D384148
51	8444F67451EE23CE1240C90F0B52A492
52	5C290D28E84060E69D09788A261B10FF
53	337E0C35E83CD38CCC5D45804241F952
54	A7879F0D31A8982A01EE6AC4952984DC
55	A37F506508928C70A83D69A2373781B9
56	42F55208EE12909803A7CBEB19B5419E
57	57E5E268A328FCC9ED04B9E5420AC702
58	EB033AD1222F84D8642C4E3FAAD28206
59	98EE1415F026AC0E862C520451697DD0
60	6A0528AEA4B7CD6702660D81F8821E19
61	763D626A87C603BCB09E1A4C800A378F
62	EEA61897879289340C23F669D6A03762
63	A6571B3CC2D0E04F017ACC808B92DCE7
64	DDF88B52EA1831D293A803CF23C8C471
65	6CA4D333A2684140475DAB491F61C17A
66	A7D2AD23043989A13289F7C3E135580A
67	B1C752FA66B41C81904EDE27EA000E2E
68	8694BE3CC1CB36BE2A095F89CC619080
69	9C20334E1BBC596B25E151180BF99940
70	484256214F81070DD9C49A2B05A43DCE
71	401A20BCBE29B7438A7AEE44635A9E23
72	8858585C3239CBF628033FA0DF189378
73	EFA36404C1BA5118CC5F9052FD28D9C3
74	155609873D8A042D496E6477B747C4F8
75	8446077883A6D7D2549CC9742E3FD023
76	E630142B189AA209371A6F0FFDBC30A7
77	C46060535AC6DBB2095F1D7826D0CD5C
78	E00D19E48797148B28DEDA9D429362E2
79	645DE447E938485489416CAFCC1C571F
80	DA10AFBF2AE61C593A1D88584DE30598
81	BB248AEA5FD3FE210CD48FC401E1A686
82	A89F146BD9191F445301C081CB6F5625
83	15BBF04F247C59150208949EB6B9CC58

84	08F48BFA7804B5B2CC2E96510232E062
85	9AA2BE74005A3679C626B209580B8D03
86	9D40664A2C808F2F293E255398B37E6A
87	6869C98A8AAD81CAE41A23C83FF9EEA0
88	576E8948E61BD0927C4140C3C04C4CF3
89	0F942C67A1137B6EAA058C2A74872C73
90	9D058E27ED546C10632684BBC84E5BC1
91	79D4B840E20148B134F90B51164BCBD0
92	0E35E1D8D1214C05FAC790B69B239150
93	FFA1BB0232CD71480BE5CA1C2A269F89
94	B2956F5F4E270446F9211584792628DB
95	F56CCA23421C8EC8F8A41F7DA4A41EA2
96	0B5ECA04F1789A7148C80C39D57D05F6
97	A10B538E8A8CFC8F8925C485F2A88660
98	9925C2C715001D9FC78ACCC51DA1AF34
99	0DAC9CFDEA40429A8B12C7D320D60F70
100	377FC9A097017958440914E83118E39D
101	8421096FA8B47E4E943B6473671955CC
102	574086183477C4F68540CB7E858263B1
103	895B6A8980C6703C779F49F40C5CFC19
104	D0D253E157BC19262150CEA668679E71
105	B8889C60EBA812BD7F0B6498823296D2
106	A13FB9F3A08528E44B13C12CF0D461AA
107	8D4DCFBE43D6E2024B1F8470224AA330
108	536D159E119E0893838657B12A074E64
109	DCFD49C504AD3A2F049A0CB70238EC8A
110	D363DB4C46C11757FA8FB18139789102
111	424A1E8A1D4DA256E4CA3BC8C2201BE3
112	417B619ED30FEB0A847CC3A191A20398
113	843FBBC95453C61786D1332612B45B4D
114	F26CACC0732CF8ED0C5BC1462B1620B4
115	88E0FE440C70E9249A92A7AF94638880
116	99A52B7D8C950308057E0661D7459960
117	A5C28218BF5D16E63E42698A0A6B0896
118	B2763BEEC784A12E8C50778536921806
119	987B2B6A3A77A059B30A082457AB84E0
120	820DB500F1B206358D7A7F210AB85AA8
121	97760A5CFC5E03EB439C914590045938
122	896A720E8857C8708A59F8C94DE0841E
123	2D101F0CF95263843412577340DEBB11
124	E8E5214B4DCF5D11A245B0149D49C87C
125	51224EAA10099ACDE384834A5ADF03D8
126	64E51253554A230C186FDE4E8781BC09
127	A499E391E69ED08890AC1A82A6115BEC

128	EE54C6E1834210D3EC1B07A456B92AA8
129	949DB5CA82420B54C1E0BCC111E704D9
130	9439EE9A9E4C447D1AA350926495047F
131	AD095CC0E7438AECE38D60980B3F2D00
132	83089C254C5EE9788072BC3D9282F798
133	A27DC1A457BC5A56563D8A9B11203615
134	713053A9C0B1B08B14705FF5A7244DB4
135	D36D4B9F4007354E0EC1B0CA8C8C7124
136	82E7C990612114F1CCE1BD9509FD4386
137	C8D83FF0B48B14830D2015D53F8C0672
138	08AF223C869A36B169148FDDABB7D120
139	B6C284C600AD0A99F86C449F8F4C53A6
140	DC741B320C07682AF92AC4DBDE0C28C2
141	89B8D84FA902265850C0FA6FF0EB2C4F
142	A69445B3A52201DB984BC03D1956D7F3
143	0FE0F7224B7AD72E4D4530D0223F590C
144	1B8C06F051434048EB925133AD3BD3F9
145	E133D4C3C942726A351300C37E55D0DF
146	9E09481D1881A66F562D8B453BC83AB2
147	2397B04B60A3C5700907BDBBA4E818C8
148	8F81F7A08CC6C8DA3D692AD34F50C012
149	9AB325352981BCCFA072F8FDE3009221
150	4FA88B7F1F8A620C31B0D486C52AC2F6
151	097AF0ADD16D7D39851049F0130EE444
152	A5027732DACFF11C388D5820A4A9BA49
153	1CD981EA2EDB46218A407C7E20D4BE84
154	D0FD94279FA67EC61A3904C0AD8ACA04
155	EA73A9415EC2004D49E9D0F645961C75
156	005AF0614A7552041194DEECBF8DD016
157	B514481533DA0A731705B93CF634E40D
158	983054521841A6E4FF34B2C07B5684FE
159	C46D927D0FD2B2F509550025677C6871
160	2AD85C08127487C87ECE014D65169102
161	0F617852FA3930AA7EE74B400B2CC831
162	AE9D395004C6E27540C378625D36E0D6
163	DC4FA55750F10B0636248F12C212FFE4
164	D3602B8D6CBF1809C88B827185631ECF
165	A94825850708E7723EA8F22C44BF78B2
166	A62D231C16AEEFE0B0026B306662945A
167	9C7BE810A86465A50551F89125D93B12
168	9712D9338B9CC60485C10172F50F121F
169	A3902CE0E0B9912591FF28C695728257
170	4167057891AB29473A9E0F67F3658921
171	B3368B91EC12A284BC414C8F0D7F8D20

172	EE21888101ABF06C1175828CB58B598D
173	E43923A00ECC32CCC2D162A4A44BD7F4
174	CC9E30B8538AD51703EEB6F70801AB22
175	B908AD2F1501DA1C156811736CD798CD
176	2B46302ACCC2F808797FC648A614326D
177	8A54494F1BE27235B8764023AA0FBCFA
178	BC1041E6F636421E89277DC154439103
179	275B39A63029B974E3561AE0A8FC8032
180	9283F6FE819B80492A22B85CE5CE5DC4
181	4CCB52C0CE058A78022C22DF5788CBCC
182	B0DF9608DE549A6F6C581516919A81E6
183	2CA185163CC36060D1E85BB0A7FBB988
184	66101D2846155CAC986FC790D2124EFC
185	8016E3904644D2093579B83BD7AB5071
186	531CAB7085BEC14257439658023647CF
187	DF2910165AA5051E41F6EB198E4D491C
188	BA32052042B0FB2188DE7857DA1B6788
189	9E6D075AFF0EA4153615E140BF380666
190	9ACC5A037902534642A3BE391AA40F9B
191	4D741A3B4499843010D7E5FA8988DC80
192	FA1421C96EDC6092726154560B1C2FC8
193	882946076223CAE0B0BFE3EDA59826D5
194	CEBB288C28B7472A0D3917012276C034
195	BD35A6E00C9528DB38289CF823C34F30
196	E2C93618B6B2800D51171A5F85746A55
197	B43EF39A1A64F0E220AF740F9494291B
198	AC537817C2612744A58132A8AFBC44A3
199	98A321249A821DDBF81C38235A371A14
200	AE1D46069090D81BB6B08FED9E687285
201	7EAE2415DC2CD60AE083249A33B56E05
202	3D942AAA9BC9F27289421CE0B301FB98
203	1548BA6D08530727AC6D059C005C6C42
204	FF47C21142C65B502DA70647BAE831D1
205	C83AA7FEAC5E51A08091E10DB0C233D9
206	E86EDD2EC2DAA3104229EDC43471A16A
207	22FAFB9C184B78B56EE91B6602C03244
208	E45631DC509B1290C08D2C1A1F15DBFE
209	D203C51207092B56568FDAD9E2D44473
210	2AA87F31A7D1AB1C90024F936006C4A5
211	913136153593DEABC7305BF0C5A62180
212	D8DA5FE401F2758642A082C53A6A5CB8
213	23C2295213147F324DE8EC1C103BAE88
214	883AF097FCDE82B366A1844245E0D727
215	79E5E9F8C933159ACADC22A06F900A70

216	FE40502B44A9E44B2C336250D47538CC
217	670452E19172C843176F1278FE41D584
218	B7EAA436078E6886A3024F593AD57580
219	1044D4CDD7230E7B1953AD1232DF07E2
220	4D821ECAC3D845A2E1011695624576FF
221	96622ED2FBD44D1B859D70601999F438
222	CCC31C3D6D5B41B8D82FF4522A4C0146
223	4A84F7CD62E0C712980E6A0C89BF394F
224	10E56751F000927284DBE174E68ECC4C
225	A3DE70921356F026E084CFE302A210A9
226	B12DA0621B343A8C3FE941A32EA5D571
227	D653135DE825A74B743E275C19020C71
228	5CAD301BF846B2EE921D33A3D4BB1220
229	1292445ACBB548C668FC3853578474E6
230	B94B4B89C0654688C9E007D9061DF5FE
231	75A2C91E76061A8680884E8BFD14A64A
232	83726F3070B47ECE21504A5065D74A36
233	964A471444A270840919F7FE07382D14
234	A582701EBFCA899B8497088C3560F300
235	64FCB63E21CAC63002D1E09FD1543274
236	B1E1C83F689ADF422C865F98D288838A
237	A06A0D822165D3F3416B47419ECCB547
238	1D2068039A32B7EF728914ECE07CB416
239	64C0CF81F78E8823ECC8661A5295422A
240	902A7243F593F2180E5A306A8438E6A9
241	A4CCED356D56BF1B41C28E1504301FE8
242	82AE90E2F76B3055A2E3A966025CC01A
243	8B90D5A62364E18574145C5895CEFF60
244	43F7EA1AB0D19032551AD9DE21307353
245	DD5D8424AC60360B1C14E65815C9B15E
246	C632A67382ECB2681DFB8525140E2878
247	3A6ACF212B6F8B9C53FF224C2E00C16C
248	86A90C267B1171093F362FE5CB14E3A0
249	EA262EC36E6589C3BB005426AF2590F4
250	200F03126C5B0D7B901128E7757C5F70
251	68FC090C2221AA98BF0D24E85066EFC2
252	9E26CEC67832FC42A87E92FA1015212E
253	ACD889634F79506F2582EA03240F2A07
254	AA65407E1F4A33BF9A62860A3D6A4CC0
255	B1B950AC76A608AA32D04B03C7FF24D3

Annex B (informative): Generalised Hierarchical Golay Sequences

B.1 Alternative generation

The generalised hierarchical Golay sequences for the PSC described in 7.1 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:

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

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

$a_0(k) = \delta(k)$ 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 $b_4(k) = a_4(k)$, $b_6(k) = a_6(k)$.

Annex C (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
14/01/00	RAN_05	RP-99593	-		Approved at TSG RAN #5 and placed under Change Control	-	3.0.0
14/01/00	RAN_06	RP-99696	001	01	Primary and Secondary CCPCH in TDD	3.0.0	3.1.0
14/01/00	RAN_06	RP-99695	003	1	Alignment of Terminology Regarding Spreading for TDD Mode	3.0.0	3.1.0
14/01/00	RAN_06	RP-99696	004	-	Code allocation for Case 3	3.0.0	3.1.0
14/01/00	-	-	-		Change history was added by the editor	3.1.0	3.1.1
31/03/00	RAN_07	RP-000069	002	3	Cycling of cell parameters	3.1.1	3.2.0
31/03/00	RAN_07	RP-000069	005	-	Removal of Synchronisation Case 3 in TDD	3.1.1	3.2.0
31/03/00	RAN_07	RP-000069	006	1	Signal Point Constellation	3.1.1	3.2.0
03/05/00	-	-	-	-	Revision marks accepted to create clean version	3.2.0	3.2.1
26/06/00	RAN_08	RP-000273	008	-	Editorial Modifications for 25.223	3.2.1	3.3.0
26/06/00	RAN_08	RP-000273	009	-	Editorial modification of 25.223	3.2.1	3.3.0
26/06/00	RAN_08	RP-000273	010	-	Editorial modification of 25.223	3.2.1	3.3.0
26/06/00	RAN_08	RP-000273	011	2	Editorial modification of 25.223	3.2.1	3.3.0
26/06/00	RAN_08	RP-000273	012	2	Modified code sets on SCH for cell search in UTRA TDD	3.2.1	3.3.0
26/06/00	RAN_08	RP-000273	013	1	Editorial update of TS25.223	3.2.1	3.3.0
23/09/00	RAN_09	RP-000346	007	1	Gain Factors for TDD Mode	3.3.0	3.4.0
23/09/00	RAN_09	RP-000346	014	-	Synchronisation codes	3.3.0	3.4.0
16/03/01	RAN_11	-	-	-	Approved as Release 4 specification (v4.0.0) at TSG RAN #11	3.4.0	4.0.0
16/03/01	RAN_11	RP-010064	015	1	Code specific phase offsets for TDD	3.4.0	4.0..0
16/03/01	RAN_11	RP-010073	016	-	Cell synchronisation codes for R'4 Node B sync over air interface in UTRA TDD	3.4.0	4.0.0
16/03/01	RAN_11	RP-010071	017	1	Inclusion of 1.28Mcps TDD in TS 25.223	3.4.0	4.0.0
15/06/01	RAN_12	RP-010337	019	-	Addition to the abbreviation list and definition of a constant	4.0.0	4.1.0
21/09/01	RAN_13	RP-010524	021	1	Clarification of notations in TS25.221 and TS25.223	4.1.0	4.2.0
21/09/01	RAN_13	RP-010530	022	1	Clarification of notations in TS25.221 and TS25.223	4.1.0	4.2.0
14/12/01	RAN_14	RP-010748	023	-	A correction of Figure 7 in subclause 7.7.2 of TS 25.223	4.2.0	4.3.0
08/03/03	RAN_15	RP-020051	025	1	Removal of quantisation of bj gain factor when calculated from a reference TFC	4.3.0	4.4.0
08/03/03	RAN_15	RP-020051	028	-	Channelisation code-specific multiplier operation under autonomous SF change	4.3.0	4.4.0
08/03/03	RAN_15	RP-020051	030	-	Alignment of gamma(i) gains of 25.223 with SIR target of WG2 25.331	4.3.0	4.4.0
08/03/03	RAN_15	RP-020058	026	1	CR to include HSDPA in TS25.223	4.4.0	5.0.0
07/06/02	RAN_16	RP-020317	031	-	Correction of SPC for 16QAM in TDD	5.0.0	5.1.0
22/12/02	RAN_18	RP-020852	033	-	Editorial modification to the section numberings	5.1.0	5.2.0
25/03/03	RAN_19	RP-030140	034	3	Miscellaneous Corrections	5.2.0	5.3.0
13/01/04	RAN_22	-	-	-	Created for M.1457 update	5.3.0	6.0.0
12/12/05	RAN_30	RP-050728	0037	-	Correction to 16QAM modulation function	6.0.0	6.1.0
20/03/06	RAN_31	RP-060079	0038	-	Introduction of 7.68Mcps TDD option	6.1.0	7.0.0
12/06/06	RAN_32	RP-060295	0040	-	Correction of the values of weight factors	7.0.0	7.1.0
29/09/06	RAN_33	RP-060492	0041	-	Introduction of E-DCH for 3.84Mcps and 7.68Mcps TDD	7.1.0	7.2.0
13/03/07	RAN_35	RP-070118	0042	1	Introduction of E-DCH for 1.28Mcps TDD	7.2.0	7.3.0

History

Document history		
V7.0.0	March 2006	Publication
V7.1.0	June 2006	Publication
V7.2.0	September 2006	Publication
V7.3.0	March 2007	Publication