Technical Specification Group, Radio Access Network
Meeting #4, Miami, 17-18 June 1999

TSGR#4(99)327

Source:

Title: **Multiplexing and channel coding (TDD)**

Document for: **Endorsement**

Agenda Item: **6.1**

TS 25.222 V1.1.0 1999-06

3rd Generation Partnership Project (3GPP);
Technical Specification Group (TSG)
Radio Access Network (RAN);
Working Group 1 (WG1);
Multiplexing and channel coding (TDD);

Reference

<Workitem>

Keywords

<keyword[, keyword]>

3GPP

Postal address

Office address

Internet

http://www.3gpp.org

Contents

1 Intellectual Property Rights

<Editor’s note: this section will be completed when an official format for the document is agreed>

2 Foreword

This Technical Specification has been produced by the 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 this TS, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version 3.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 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 specification;

3 Scope

This 3GPP Report describes multiplexing, channel coding and interleaving for UTRA Physical Layer TDD mode.

Text without revision marks has been approved in the previous TSG-RAN WG1 meetings, while text with revision marks is subject to approval.

4 References

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

[1] TS 25.202 (V1.0.0): “UE capabilities”

[2] TS 25.211 (V1.0.0): “Transport channels and physical channels (FDD)”

[3] TS 25.212 (V1.0.0): “Multiplexing and channel coding (FDD)”

[4] TS 25.213 (V1.0.0): “Spreading and modulation (FDD)”

[5] TS 25.214 (V1.0.0): “Physical layer procedures (FDD)”

[6] TS 25.221 (V1.0.0): “Transport channels and physical channels (TDD)”

[7] TS 25.222 (V1.0.0): “Multiplexing and channel coding (TDD)”

[8] TS 25.223 (V1.0.0): “Spreading and modulation (TDD)”

[9] TS 25.224 (V1.0.0): “Physical layer procedures (TDD)”

[10] TS 25.231 (V1.0.0): “Measurements”

[11] TS S2.01 (V1.0.0): “Radio Interface Protocol Architecture”

5 Definitions, symbols and abbreviations

5.1 Definitions

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

<defined term>: <definition>.

5.2 Symbols

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

<symbol> <Explanation>

5.3 Abbreviations

ARQ	Automatic Repeat on Request
BCH	Broadcast Channel
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BS	Base Station
BSS	Base Station Subsystem
CA	Capacity Allocation
CAA	Capacity Allocation Acknowledgement
CBR	Constant Bit Rate
CCCH	Common Control Channel
CCTrCH	Coded Composite Transport Channel
CD	Capacity Deallocation
CDA	Capacity Deallocation Acknowledgement
CDMA	Code Division Multiple Access
CTDMA	Code Time Division Multiple Access
CRC	Cyclic Redundancy Check
DCA	Dynamic Channel Allocation
DCCH	Dedicated Control Channel
DL	Downlink
DRX	Discontinuous Reception
DSCH	Downlink Shared Channel
DTX	Discontinuous Transmission
FACH	Forward Access Channel
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
FEC	Forward Error Control
FER	Frame Error Rate

HCS	Hierarchical Cell Structure
JD	Joint Detection
L1	Layer 1
L2	Layer 2
LLC	Logical Link Control
MA	Multiple Access
MAC	Medium Access Control
MAHO	Mobile Assisted Handover
MO	Mobile Originated
MOHO	Mobile Originated Handover
MS	Mobile Station
MT	Mobile Terminated
NRT	Non-Real Time
PC	Power Control
PCH	Paging Channel
QoS	Quality of Service
QPSK	Quaternary Phase Shift Keying
RACH	Random Access Channel
RF	Radio Frequency
RLC	Radio Link Control
RRC	Radio Resource Control
RRM	Radio Resource Management
RT	Real Time
RU	Resource Unit
SCH	Synchronization Channel
SDCCH	Stand-alone Dedicated Control Channel
SP	Switching Point
TCH	Traffic channel
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
UL	Uplink
UMTS	Universal Mobile Telecommunications System
VBR	Variable Bit Rate

6 Multiplexing, channel coding and interleaving

6.1 General

Data stream from/to MAC and higher layers (Transport block / Transport block set) is encoded/decoded to offer transport services over the radio transmission link. Channel coding scheme is a combination of error detection, error correcting (including rate matching), and interleaving and transport channels mapping onto/splitting from physical channels.

In the UTRA-TDD mode, the total number of basic physical channels (a certain time slot one spreading code on a certain carrier frequency) per frame is given by the maximum number of time slots which is 16 and the maximum number of CDMA codes per time slot. This maximum number of codes is 8 in case the different codes within one time slot are allocated to different users in the uplink and is higher than 8 (e.g. 9 or 10) in the downlink or if several codes are allocated to one single user in the uplink.

6.2 Transport channel coding/multiplexing

Figure 6-1 illustrates the overall concept of transport-channel coding and multiplexing. Data arrives to the coding/multiplexing unit in form of transport block sets, once every transmission time interval. The transmission time interval is transport-channel specific from the set {10 ms, 20 ms, 40 ms, 80 ms}.

The following coding/multiplexing steps can be identified:

- Add CRC to each transport block (see section 6.2.1)
- Channel coding (see section 6.2.3)
- Rate matching (see section 6.2.5)
- Interleaving (two steps, see sections 6.2.4 and 6.2.8)
- Radio frame segmentation *<Editor's note: A new section should be added to describe radio frame segmentation.>*
- Multiplexing of transport channels (two steps, see sections 6.2.2 and 6.2.6)
- Physical channel segmentation (see section 6.2.7)
- Mapping to physical channels (see section 6.2.9)

The coding/multiplexing steps for uplink and downlink are shown in Figure 6-1.

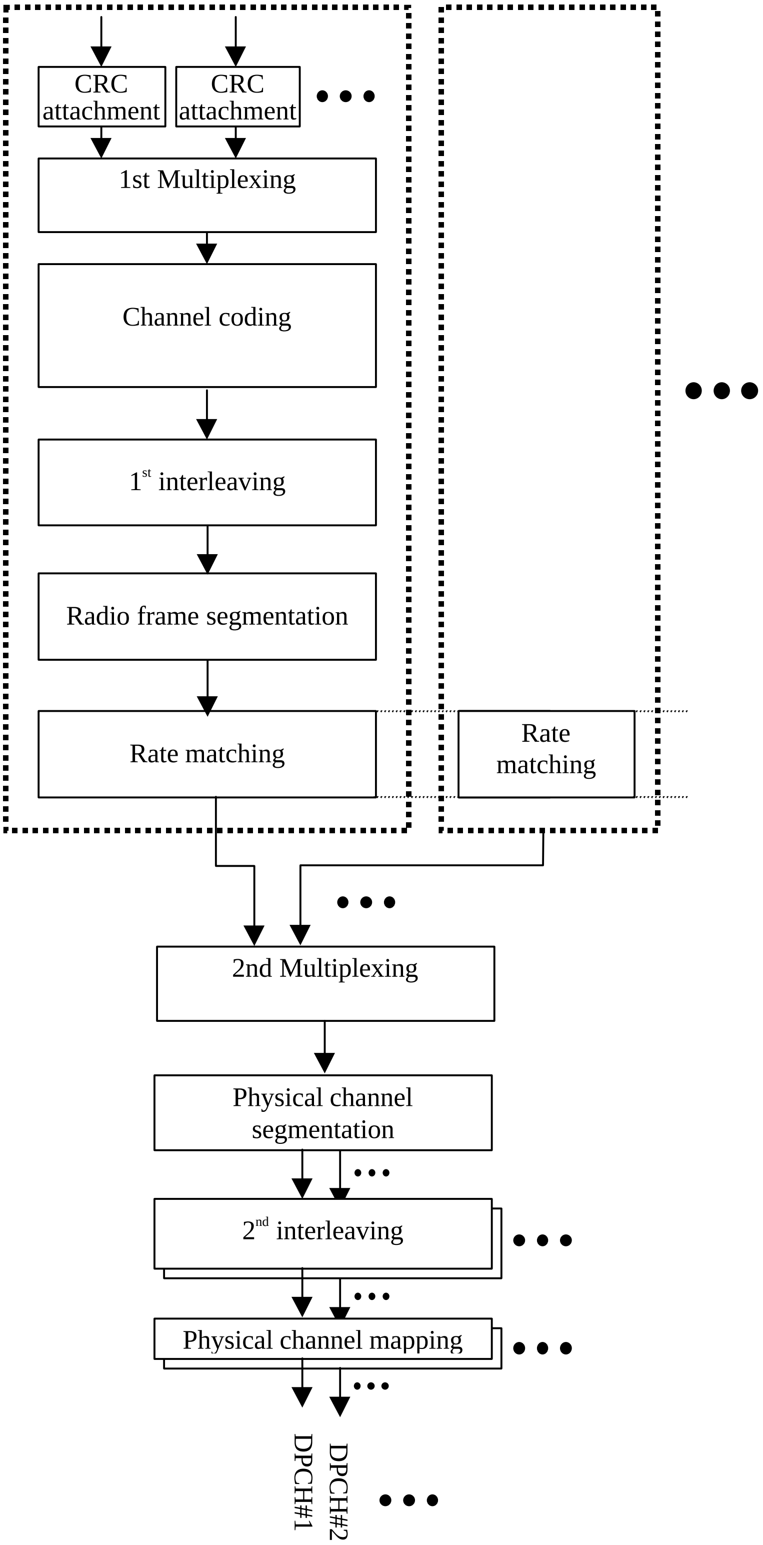


Figure 6–1. Transport channel multiplexing structure for uplink and downlink

Primarily, transport channels are multiplexed as described above, i.e. into one data stream mapped on one or several physical channels. However, an alternative way of multiplexing services is to use multiple CCTrCHs (Coded Composite

Transport Channels), which corresponds to having several parallel multiplexing chains as in Figure 6-1, resulting in several data streams, each mapped to one or several physical channels.

6.2.1 Error detection

Error detection is provided on transport blocks through a Cyclic Redundancy Check. The CRC is 16, 8 or 0 bits and it is signalled from higher layers what CRC length that should be used for each transport channel.

6.2.1.1 CRC calculation

The entire transport block is used to calculate the CRC parity bits for each transport block. The parity bits are generated by one of the following cyclic generator polynomials:

$g_{CRC16}(D) = D^{16} + D^{12} + D^{5} + 1$

$g_{CRC8}(D) = D^{8} + D^{7} + D^{4} + D^{3} + D + 1$

Denote the bits in a transport block delivered to layer 1 by $b_1, b_2, b_3, \dots b_N$, and the parity bits by $p_1, p_2, \dots p_L$. N is the length of the transport block and L is 16, 8, or 0 depending on what is signalled from higher layers.The encoding is performed in a systematic form, which means that in GF(2), the polynomial

$b_1D^{N+15} + b_2D^{N+14} + \dots + b_ND^{16} + p_1D^{15} + p_2D^{14} + \dots + p_{15}D^{1} + p_{16}$

yields a remainder equal to 0 when divided by $g_{CRC16}(D)$. Similarly,

$b_1D^{N+7} + b_2D^{N+6} + \dots + b_ND^{8} + p_1D^{7} + p_2D^{6} + \dots + p_7D^{1} + p_8$

yields a remainder equal to 0 when divided by $g_{CRC8}(D)$.

6.2.1.2 Relation between input and output of the Cyclic Redundancy Check

Bits delivered to layer 1 are denoted $b_1, b_2, b_3, \dots b_N$, where N is the length of the transport block. The bits after CRC attachment are denoted by $w_1, w_2, w_3, \dots w_{N+L}$, where L is 16, 8, or 0. The relation between b and w is:

$w_k = b_k \qquad k = 1, 2, 3, ... N$

$w_k = p_{(L+1-(k-N))} \qquad k = N+1, N+2, N+3, ... N+L$

6.2.2 1st multiplexing

Fix rate transport channels that are characterised by the same transport format attributes (as defined in 25.302) can be multiplexed before coding. When this multiplexing step is present, the transport blocks from different transport channels are serially concatenated. Denote the number of transport channels (TrCHs) by R, the number of transport blocks on each TrCH by P, and the number of bits in each transport block, including CRC bits, by K. The bits before multiplexing can then be described as follows:

Bits from transport block 1 of transport channel 1: $w_{111}, w_{112}, w_{113}, \dots w_{11K}$

Bits from transport block 2 of transport channel 1: $w_{121}, w_{122}, w_{123}, \dots w_{12K}$

…

Bits from transport block P of transport channel 1: $w_{1P1}, w_{1P2}, w_{1P3}, \dots w_{1PK}$

Bits from transport block 1 of transport channel 2: $w_{211}, w_{212}, w_{213}, \dots w_{21K}$

…

Bits from transport block P of transport channel 2: $w_{2P1}, w_{2P2}, w_{2P3}, \dots w_{2PK}$

...

Bits from transport block 1 of transport channel R: w_{R11}, w_{R12}, w_{R13}, … w_{R1K}

…

Bits from transport block P of transport channel R: w_{RP1}, w_{RP2}, w_{RP3}, … w_{RPK}

The bits after first multiplexing are denoted by d_1, d_2, d_3, … d_M, and defined by the following relations:

$$\left.\begin{array}{ll} d_k = w_{11k} & k = 1, 2, \ldots, K \\ d_k = w_{12(k-K)} & k = K + 1, K + 2, \ldots, 2K \\ \ldots & \\ d_k = w_{1P(k-(P-1)K)} & k = (P - 1)K + 1, \ldots, PK \end{array}\right\} \text{TrCH 1}$$

$$\left.\begin{array}{ll} d_k = w_{21(k-PK)} & k = PK + 1, \ldots, (P + 1)K \\ \ldots & \\ d_k = w_{2P(k-(2P-1)K)} & k = (2P - 1)K + 1, \ldots, 2PK \end{array}\right\} \text{TrCH 2}$$

…

$$\left.\begin{array}{ll} d_k = w_{R1(k-(R-1)PK)} & k = (R - 1)PK + 1, \ldots, ((R - 1)P + 1)K \\ \ldots & \\ d_k = w_{RP(k-(RP-1)K)} & k = (RP - 1)K + 1, \ldots, RPK \end{array}\right\} \text{TrCH R}$$

<Note: Above it is assumed that all transport blocks have the same size. There are cases when the total number of bits that are sent during a transmission time interval is not a multiple of the number of transport blocks. A few padding bits are then needed but the exact insertion point (in the multiplexing chain) of these bits is for further study.>

6.2.3 Channel coding

The following channel coding schemes can be applied to transport channels.

- Convolutional coding
- Turbo coding
- Service-specific coding, e.g. unequal error protection for some types of speech codecs.

The maximum encoding segment length for turbo coding is 5120 bits. In Real Time (RT) services a FEC coding is used, instead Non Real Time (NRT) services could be well managed with a proper combination of FEC and ARQ.

Table 6.2.3-1 Error Correction Coding Parameters

Transport channel type	Coding scheme	Coding rate
BCH	Convolutional code	1/2
PCH		
FACH		
RACH		1/2, [2/3, 7/8] *<Editor's note: the values in square brackets have not yet been approved.>*
DCH		1/2 or 1/3
DCH	Turbo code	

Note 1: The exact physical layer encoding/decoding capabilities for different code types are FFS.

Note 2: In the UE the channel coding capability should be linked to the terminal class.

6.2.3.1 Convolutional Coding

- Constraint length K=9. Coding rates 1/2, 1/3 and [2/3, 7/8].
- The configuration of the convolutional coder is presented in Fig. 6-2.
- The output from the convolutional coder shall be done in the order starting from output0, output1 and output2. (When coding rate is 1/2, output is done up to output 1).
- The initial value of the shift register of the coder shall be “all 0”.
- K-1 tail bits (value 0) shall be added to the end of the coding block.

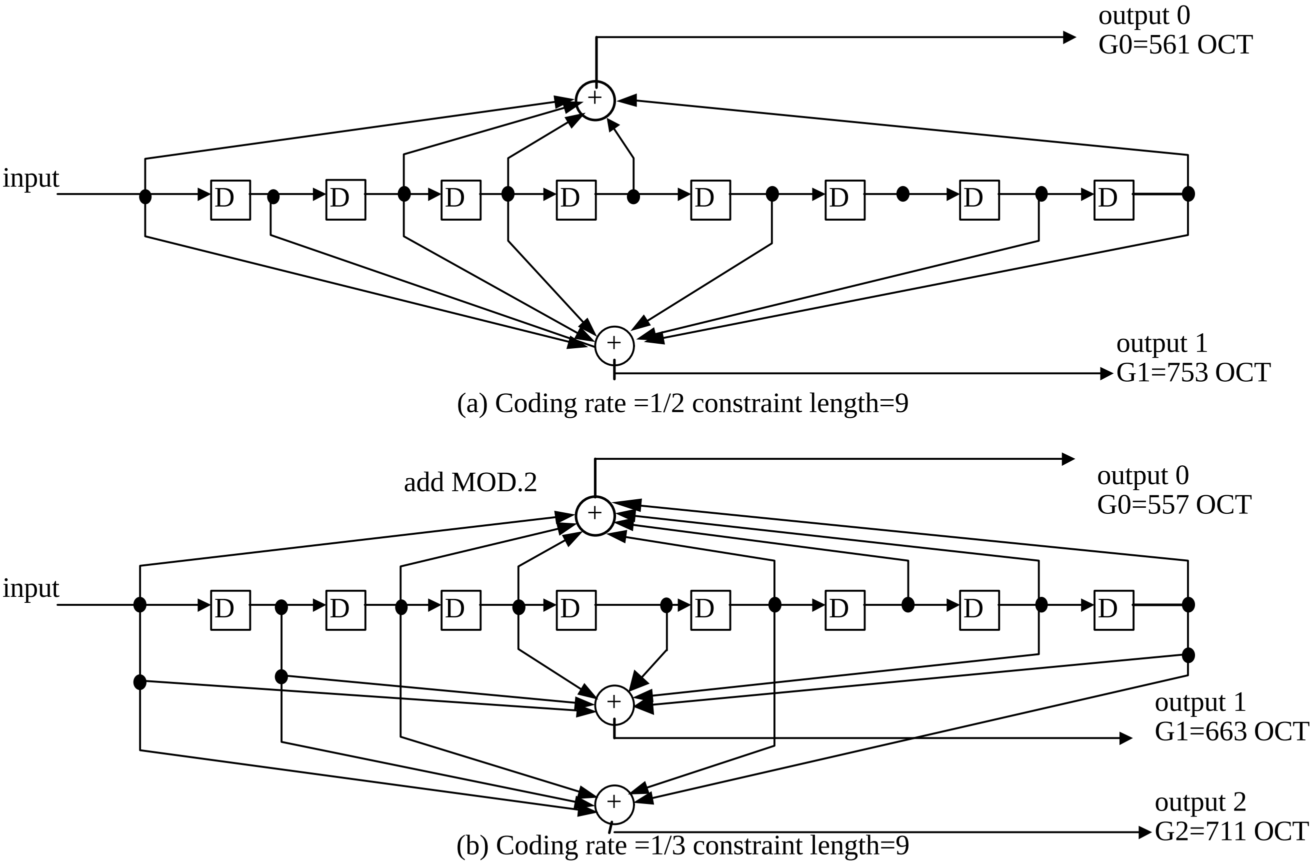


Fig. 6-2 Convolutional Coder

6.2.3.1.1 Segmentation into code blocks for convolutional coding

<Note: It is for further study if the maximum code block size is 504 or shorter.>

If the transport blocks or multiplexed transport blocks are longer than [504] bits (including CRC bits), they are segmented before convolutional encoding. Denote the number of transport blocks before coding by P and the number of bits in each transport block or the sum of the number of bits in the multiplexed blocks by M. Note that if first multiplexing is performed, all transport blocks of a transport channel in the same transmission time interval are multiplexed together, i.e. P=1. The bits before segmentation can then be described as follows:

Bits in transport block 1before segmentation: $d_{1,1}$, $d_{1,2}$, $d_{1,3}$, … $d_{1,M}$

Bits in transport block 2 before segmentation: $d_{2,1}$, $d_{2,2}$, $d_{2,3}$, … $d_{2,M}$

…

Bits in transport block P before segmentation: $d_{P,1}$, $d_{P,2}$, $d_{P,3}$, … $d_{P,M}$

If M ≤ [504], no segmentation is performed. If M > [504] the following parameters are calculated:

Number of code blocks: S = round_up(PM / [504])

Length of coded blocks: C = round_up(PM / S)

Remainder: R = PM – S round_down(PM / S)

Number of filler bits: F = S – R, if R≠0
F = 0, if R=0

round_up(x) means the smallest integer number larger or equal to x.

round_down(x) means the largest integer number smaller or equal to x.

The F filler bits are appended to the end of the last code block before tail insertion and channel encoding. They are denoted f_1, f_2, f_3, ... f_F. The bits after segmentation are denoted by $u_{1,1}$, $u_{1,2}$, $u_{1,3}$, … $u_{1,C}$, $u_{2,1}$, $u_{2,2}$, $u_{2,3}$, ... ,$u_{2,C}$, ... $u_{S,1}$, $u_{S,2}$, $u_{S,3}$, ... ,$u_{S,C}$, and defined by the following relations:

$u_{1,k} = d_{1,k}$	k = 1, 2, 3, ... C
$u_{2,(k-C)} = d_{1,k}$	k = C + 1, C + 2, C + 3, ... 2C
...	
$u_{j,(k-(j-1)C)} = d_{1,k}$	k = (j-1)C + 1, (j-1)C + 2, (j-1)C + 3, ... M
$u_{j,(k-(j-1)C)} = d_{2,(k-M)}$	k = M + 1, M + 2, M + 3, ... jC
$u_{j+1,(k-jC)} = d_{2,(k-M)}$	k = jC + 1, jC + 2, jC + 3, ... (j+1)C
...	
$u_{S,(k-(S-1)C)} = d_{P,(M-C+F+k-(S-1)C)}$	k = (S - 1)C + 1, (S - 1)C + 2, (S - 1)C + 3, ... SC - F
$u_{S,(k-(S-1)C)} = f_{k-SC+F}$	k = SC – F + 1, SC - F+ 2, SC – F +3, ... SC

<Note: Above it is assumed that all transport blocks have the same size. There are cases when the total number of bits that are sent during a transmission time interval is not a multiple of the number of transport blocks. A few padding bits are then needed but the exact insertion point (in the multiplexing chain) of these bits is for further study.>

6.2.3.2 Turbo coding

6.2.3.2.1 Turbo coder

<Note: 4-state SCCC is not included in Release –99.>

<Note: It needs to be clarified from TSG SA what are the service specifications with respect to different qualities of service. The performance below BER of 10^{-6} needs to be studied if there is a requirement for this quality of services over the physical layer.>

For data services requiring quality of service between 10^{-3} and 10^{-6} BER inclusive, parallel concatenated convolutional code (PCCC) with 8-state constituent encoders is used.

The transfer function of the 8-state constituent code for PCCC is

$$G(D)=\left[1,\frac{n(D)}{d(D)}\right]$$

where,

$$d(D)=1+D^2+D^3$$

$$n(D)=1+D+D^3.$$

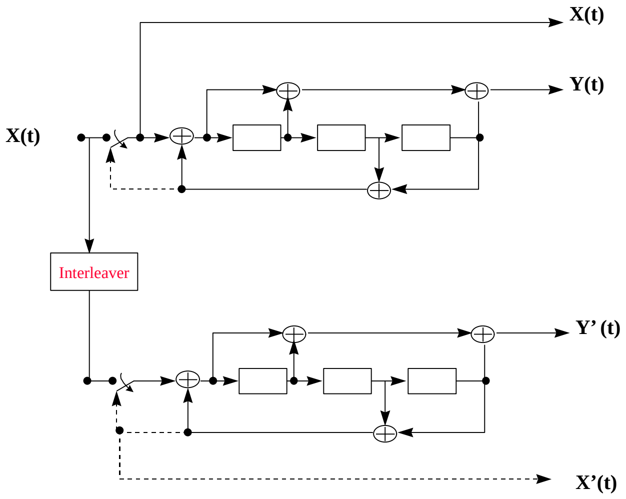


Figure 6-3. Structure of the 8 state PCCC encoder (dotted lines effective for trellis termination only)

The initial value of the shift registers of the PCCC encoder shall be all zeros.

The output of the PCCC encoder is punctured to produce coded bits corresponding to the desired code rate 1/3 or 1/2. For rate 1/3, none of the systematic or parity bits are punctured, and the output sequence is X(0), Y(0), Y'(0), X(1), Y(1), Y'(1), etc. For rate 1/2, the parity bits produced by the constituent encoders are alternately punctured to produce the output sequence X(0), Y(0), X(1), Y'(1), X(2), Y(2), X(3), Y'(3), etc.

The SCCC is a rate 1/3 SCCC, The outer code of the SCCC is a rate 2/3 obtained by puncturing a rate ½ code with generating matrix

$$G^{(o)}(Z) = (1,(1+Z^2)/(1+Z+Z^2))$$

The rate 2/3 is obtained by puncturing every other parity-check bit.
The inner code is a rate ½ systematic recursive convolutional code with the same previous generating matrix

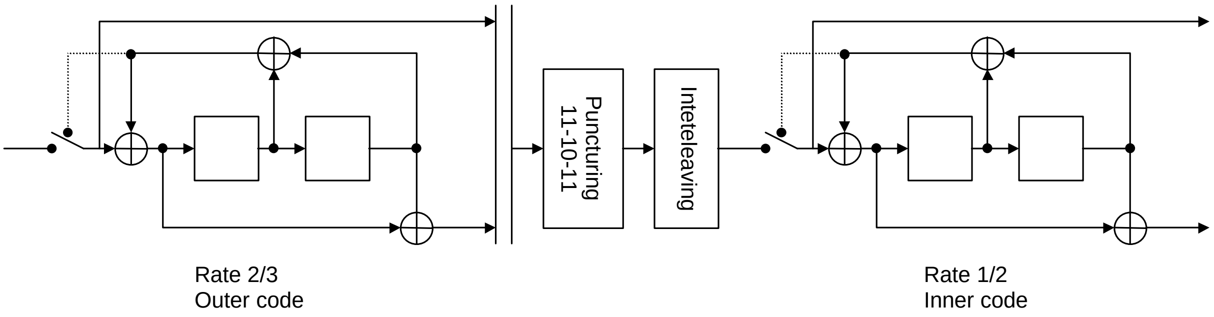


Figure 6-4. Structure of the 4 state SCCC encoder (dotted line effective for trellis termination only).

6.2.3.2.2 Trellis termination in turbo code

Trellis termination is performed by taking the tail bits from the shift register feedback after all information bits are encoded. Tail bits are added after the encoding of information bits.

Trellis termination for PCCC

The first three tail bits shall be used to terminate the first constituent encoder (upper switch of Figure 6-3 in lower position) while the second constituent encoder is disabled. The last three tail bits shall be used to terminate the second constituent encoder (lower switch of Figure 6-3 in lower position) while the first constituent encoder is disabled.

The transmitted bits for trellis termination shall then be

X(t) Y(t) X(t+1) Y(t+1) X(t+2) Y(t+2) X'(t) Y'(t) X'(t+1) Y'(t+1) X'(t+2) Y'(t+2).

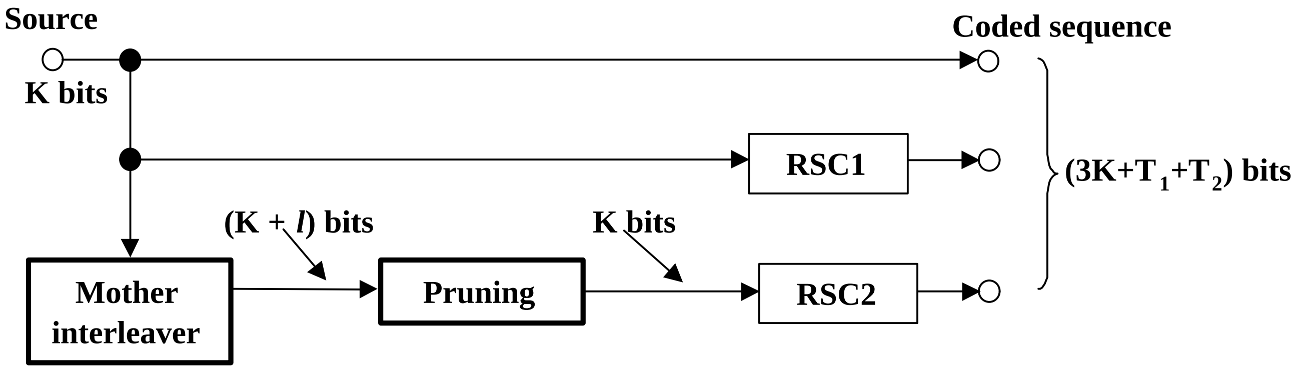
Trellis termination for SCCC

The conventional method of trellis termination is used also for SCCC in which the tail bits are taken from the shift register feedback after all bits are encoded (Figure 6-4). The tailing bits of the outer encoder are included in the interleaver. The outer code is terminated first with two additional input bits taken from the shift register feedback (dotted line), the outer code thus, after puncturing, outputs three additional bit that are feeded into the interleaver. After that all bits have been encoded from the inner encoder (included the interleaved tail bit of the outer encoder), two additional input bits are taken from the shift register feedback of the inner encoder producing four tail bits.

Thus the total overhead due to the tailing bits is 3*2+4 =10 bits.

6.2.3.2.3 Turbo code internal interleaver

Figure 6-5 depicts the overall 8 State PCCC Turbo coding scheme including Turbo code internal interleaver. The Turbo code internal interleaver consists of mother interleaver generation and pruning. For arbitrary given block length K, one mother interleaver is selected from the 207 mother interleavers set. The generation scheme of mother interleaver is described in section 6.2.2.2.3.1. After the mother interleaver generation, *l*-bits are pruned in order to adjust the mother interleaver to the block length K. The definition of *l* is shown in section 6.2.3.2.3.2..



(1) K=481 to 530 bits

K + *l* = 0 (mod 10) ;

p-1 < (K + *l*) / 10 < p+2

K + *l* = 0 (mod 20) ;

p-2 < (K + *l*) / 20 < p+2

where p is a prime number.

Figure 6-5. Overall 8 State PCCC Turbo Coding

6.2.3.2.3.1 Mother interleaver generation

The interleaving consists of three stages. In first stage, the input sequence is written into the rectangular matrix row by row. The second stage is intra-row permutation. The third stage is inter-row permutation. The three-stage permutations are described as follows, the input block length is assumed to be K (320 to 5120 bits).

First Stage:

(1) Determine a row number R such that

R=10 (K = 481 to 530 bits; Case-1)

R=20 (K = any other block length except 481 to 530 bits; Case-2)

(2) Determine a column number C such that

Case-1; C = p = 53

Case-2;

(i) find minimum prime p such that,

0 =< (p+1)-K/R

(ii) if (0 =< p-K/R) then go to (iii)

else C = p+1.

(iii) if (0 =< p-1-K/R) then C=p-1.

else C = p.

(3) The input sequence of the interleaver is written into the RxC rectangular matrix row by row.

<u>Second Stage</u>:

<u>A. If C = *p*</u>

(A-1) Select a primitive root g_0 from Table 6.2.2-2.

(A-2) Construct the base sequence c(i) for intra-row permutation as:

$c(i) = [g_0 \times c(i-1)] \bmod p$, i =1,2,…(p-2)., c(0) = 1.

(A-3) Select the minimum prime integer set {q_j } (j=1,2,…R-1) such that

g.c.d{q_j, p-1} =1

$q_j > 6$

$q_j > q_{(j-1)}$

where g.c.d. is greatest common divider. And q_0 = 1.

(A-4) The set {q_j} is permuted to make a new set {p_j} such that

$p_{P(j)} = q_j$, j = 0, 1, …. R-1,

where P(j) is the inter-row permutation pattern defined in the third stage.

(A-5) Perform the j-th (j = 0,1, 2, …, C-1) intra-row permutation as:

$c_j(i) = c([i \times p_j] \bmod (p-1))$, i =0,1,2,…, (p-2)., and $c_j(p\text{-}1) = 0$,

where c_j(i) is the input bit position of i-th output after the permutation of j-th row.

<u>B. If C = *p*+1</u>(B-1) Same as case A-1.

(B-2) Same as case A-2.

(B-3) Same as case A-3.

(B-4) Same as case A-4.

(B-5) Perform the *j*-th (*j* = 0,1, 2, …, R-1) intra-row permutation as:

$c_j(i) = c([i \times p_j] \mathrm{mod}(p-1))$, i =0,1,2,…, (p-2)., $c_j(p\text{-}1) = 0$, and $c_j(p) = p$,

where $c_j(i)$ is the input bit position of i-th output after the permutation of j-th row.

(B-6) If (K = C x R) then exhange $c_{R-1}(p)$ with $c_{R-1}(0)$.

C. If C = p-1

(C-1) Same as case A-1.

(C-2) Same as case A-2.

(C-3) Same as case A-3.

(C-4) Same as case A-4.

(C-5) Perform the j-th (j = 0,1, 2, …, R-1) intra-row permutation as:

$c_j(i) = c([i \times p_j] \mathrm{mod}(p-1))$ -1, i =0,1,2,…, (p-2).,

where $c_j(i)$ is the input bit position of i-th output after the permutation of j-th row.

Third Stage:

Perform the inter-row permutation based on the following P(j) (j=0,1, ..., R-1) patterns, where P(j) is the original row position of the j-th permuted row.

P_A: {19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 10, 8, 13, 17, 3, 1, 16, 6, 15, 11} for R=20

P_B: {19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 16, 13, 17, 15, 3, 1, 6, 11, 8, 10} for R=20

P_C: {9, 8, 7, 6, 5, 4, 3, 2, 1, 0} for R=10

The usage of these patterns is as follows:

Block length K:	P(j)
320 to 480-bit:	P_A
481 to 530-bit:	P_C
531 to 2280-bit:	P_A
2281 to 2480-bit:	P_B
2481 to 3160-bit:	P_A
3161 to 3210-bit:	P_B
3211 to 5120-bit:	P_A

(2) The output of the mother interleaver is the sequence read out column by column from the permuted R × C matrix.

Table 6.2.3-2. Table of prime *p* and associated primitive root

p	g_o	*p*	g_o	*p*	g_o	*p*	g_o	*p*	g_o	*p*	g_o	*p*	g_o	*p*	g_o
17	3	59	2	103	5	157	5	211	2	269	2	331	3	389	2
19	2	61	2	107	2	163	2	223	3	271	6	337	10	397	5
23	5	67	2	109	6	167	5	227	2	277	5	347	2	401	3
29	2	71	7	113	3	173	2	229	6	281	3	349	2	409	21
31	3	73	5	127	3	179	2	233	3	283	3	353	3		
37	2	79	3	131	2	181	2	239	7	293	2	359	7		
41	6	83	2	137	3	191	19	241	7	307	5	367	6		
43	3	89	3	139	2	193	5	251	6	311	17	373	2		
47	5	97	5	149	2	197	2	257	3	313	10	379	2		
53	2	101	2	151	6	199	3	263	5	317	2	383	5		

6.2.3.2.3.2 Definition of the number of pruning bits

The output of the mother interleaver is pruned by deleting the *l*-bits in order to adjust the mother interleaver to the block length K, where the deleted bits are non-existent bits in the input sequence. The pruning bits number *l* is defined as:

$l = \mathrm{R} \times \mathrm{C} - \mathrm{K}$,

where R is the row number and C is the column number defined in section 6.2.3.2.3.1.

6.2.3.2.4 Encoding blocks for turbo code

Input data blocks for a turbo encoder consist of the user data and possible extra data being appended to the user data before turbo encoding. The encoding segments for a turbo encoder are defined in terms of systematic bits. The segment includes the user data, a possible error detection field (CRC), possible filler bits, and the termination. The Algorithm for combining and segmentation is as follows:

Inputs:

N_{DATA} size of input data block to turbo encoder

N_{TAIL} number of tail bits to be appended to the encoding segments (termination)

Outputs:

N_{S} number of segments

N_{TB} number of bits in the turbo encoder input segments

N_{FILL} number of filler (zero) bits in the last turbo encoder input segment

Do:

1. Let N_{S} = round_up(N_{DATA} / (5120 – N_{TAIL}))

2. Let N_{TB} = round_up (N_{DATA} / N_S) + N_{TAIL};
3. Let N_{REM} = remainder of N_{DATA} / N_S;
4. If N_{REM} not equal to 0 then insert N_{FILL} = ($N_S - N_{REM}$) zero bits to the end of the input data else N_{FILL} = 0.
5. End.

Here *round_up(x)* stands for an smallest interger number being larger or equal to *x*.

All turbo encoder input segments are of equal size and therefore the same turbo interleaver can be used for all turbo segments. A number of systematic bits over an entire channel interleaving block at output of the encoder is

$$N_S * (\text{round\_up}(N_{DATA} / N_S) + N_{TAIL}).$$

The N_{FILL} filler bits are padded to the end of the last encoding segment in order to make the last segment equal size to the precedent ones. The filler bits are encoded.

6.2.3.3 Service specific coding

<Editor's note: The application of service specific coding is still under discussion in AdHoc #4.>
In addition to standard channel coding options a service specific encoder can be used. In the simplest case it can mean that there is no channel coding at all. A more typical example is an unequal-error-protection channel code for a specific speech codec.

6.2.4 1st interleaving

The 1st interleaving of channel interleaving consists of two stage operations. In first stage, the input sequence is written into rectangular matrix row by row. The second stage is inter-column permutation. The two-stage operations are described as follows, the input block length is assumed to be K_1.

First Stage:

(1) Select a column number C_1 from Table 6.2.4-1.

(2) Determine a row number R_1 by finding minimum integer R_1 such that,

$$K_1 <= R_1 \times C_1.$$

(3) The input sequence of the 1st interleaving is written into the $R_1 \times C_1$ rectangular matrix row by row.

Second Stage:

(1) Perform the inter-column permutation based on the pattern $\{P_1 (j)\}$ (j=0,1, ..., C-1) that is shown in Table 4-xx, where $P_1 (j)$ is the original column position of the j-th permuted column.

(2) The output of the 1st interleaving is the sequence read out column by column from the inter-column permuted $R_1 \times C_1$ matrix and the output is pruned by deleting the non-existence bits in the input sequence, where the deleting bits number l_1 is defined as:

$$l_1 = R_1 \times C_1 - K_1.$$

Table 6.2.4-1

Interleaving span	Column number C_1	Inter-column permutation patterns
10 ms	1	{0}
20 ms	2	{0,1}

40 ms	4	{0,2,1,3}
80 ms	8	{0,4,2,6,1,5,3,7}

6.2.5 Rate matching

The rate matching applies repetition and puncturing of the different transport channels.

For each combination of rates of the different transport channels, a puncturing/repetition factor is assigned to each transport channel. The set of puncturing/repetition factors is determined based on following criteria:

- desired transmission quality requirements of each transport channel is fulfilled and not significantly exceeded. This means that required transmission power to meet quality requirements for all transport channels is as low as possible.
- The total bit rate after transport channel multiplexing is identical to the total channel bit rate of the dedicated physical channels allocated
- on uplink and downlink, the total number of allocated resource units should be minimised
- the puncturing factors should not exceed a certain maximum puncturing factor, specific for each transport channel.

<Editor's note: Maximum puncturing factor will require further discussion.>

6.2.5.1 Determination of rate matching parameters

The selection of the relative puncturing/repetition if several coded transport channels are multiplexed together is done in a way to ensure, that the relative SNR requirements to achieve the required QoS are balanced and that the maximum allowed puncturing is not exceeded.

<Editor's note: There exist proposals for the exact determination of the puncturing rate for each individual coded transport channel, but the relevant text is still to be inserted. >

6.2.5.2 Parameters for rate matching after first interleaving

Rate matching is performed for every radio frame.

Let's denote:

F: the number of radio frames corresponding to the transmission time interval.

k: the current radio frame in the transmission time interval (0 ≤ k < F)

$R_F(k)$ is the inverse interleaving function of the 1st MIL interleaver (note that the inverse interleaving function is identical to the interleaving function itself for the 1st MIL interleaver).

N_C number of data bits before rate matching for a particular coded transport channel for the radio frame k.

N_i number of data bits after rate matching for this particular coded transport channel for the radio frame k.

```
-- calculate average puncturing distance
q:= ëN_c/(ôN_i-N_cô)û -- where ë û means round downwards and ô ômeans absolute value.

if q is even  -- avoid hitting the same column twice:
    then q' = q – gcd(q, F)/F  -- where gcd (q, F) means greatest common divisor of q and F
    --  note that q' is now not an integer, but a multiple of 1/8
else
    q' = q

endif
```

– calculate S(k), representing the shift of the puncturing pattern for the radio frame k. *S*(k) is used when preloading *e* in the rate matching formula in section 6.2.5.3 below.

```
for i = 0 to F-1
    S(R_F (éi*q'ù mod F)) = (éi*q'ù div F) -- where é ù  means round upwards.

end for
```

6.2.5.3 Rate matching algorithm

Let’s denote:

$S_0 = \left\{ d_1, d_2, \ldots, d_{N_C} \right\}$ = set of N_C data bits

The rate matching rule is as follows:

```
if puncturing is to be performed

y = $N_C$-$N_i$

e = (2*S(k) * y +  Nc) mod 2$N_C$  -- initial error between current and desired puncturing ratio
m = 1           -- index of current bit
do while m <= $N_C$
e = e – 2 * y           -- update error
if e <= 0 then          -- check if bit number m should be punctured
puncture bit m from set $S_0$
e = e + 2*$N_C$         -- update error
end if
m = m + 1               --  next bit
end do
else
y = $N_i$-$N_C$

e = (2*S(k) * y +  Nc) mod 2$N_C$  -- initial error between current and desired puncturing ratio
m = 1               -- index of current bit
do while m <= $N_C$
e = e – 2 * y           -- update error
do while e <= 0         -- check if bit number m should be repeated
repeat bit m from set $S_0$
e = e +  2*$N_C$   -- update error
enddo
m = m + 1               --  next bit
end do
end if
```

6.2.6 2nd multiplexing

<Editor’s note: This section needs to be updated since it does not take into account that transport channels may have been multiplexed before coding.>

The coded transport channels are serially multiplexed within one radio frame. The output after the multiplexer (before the 2nd interleaver) will thus be according to Fig. 6-6. Several CCTrCHs are supported.

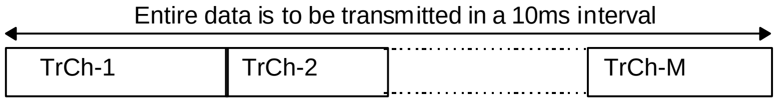


Figure 6-6.   Transport channel multiplexing.

6.2.7 Physical channel segmentation

<Editor’s note: Physical channel segmentation will depend upon the QoS parameters of the different transport channels in each CCTrCH. It will be necessary to specify what are the parameters expected from L2 for it.>

Data after multiplexing of transport channels with different QoS can get segmented into multiple physical channels which are transmitted in parallel during a 10ms interval.

6.2.8 2nd interleaving

The 2nd interleaving of channel interleaving consists of two stage operations. In first stage, the input sequence is written into rectangular matrix row by row. The second stage is inter-column permutation. The two-stage operations are described as follows, the input block length is assumed to be K_2.

First Stage:

(1) Set a column number $C_2 = 32$.

(2) Determine a row number R_2 by finding minimum integer R_2 such that,

$$K_2 <= R_2 \times C_2.$$

(3) The input sequence of the 2^{nd} interleaving is written into the $R_2 \times C_2$ rectangular matrix row by row.

Second Stage:

(1) Perform the inter-column permutation based on the pattern $\{P_2 (j)\}$ (j=0,1, ..., C-1) that is shown in Table 4-yy, where $P_2 (j)$ is the original column position of the j-th permuted column.

(2) The output of the 2nd interleaving is the sequence read out column by column from the inter-column permuted $R_2 \times C_2$ matrix and the output is pruned by deleting the non-existence bits in the input sequence, where the deleting bits number l_2 is defined as:

$$l_2 = R_2 \times C_2 - K_2.$$

Table 6.2.8-1

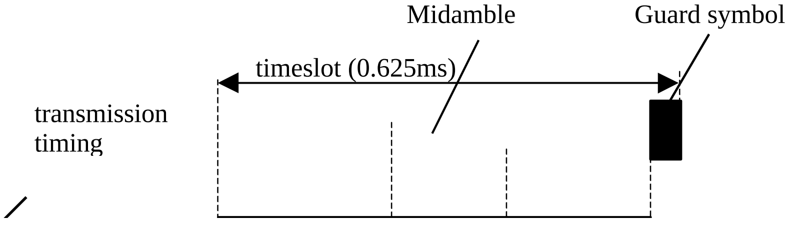
Column number C_2	Inter-column permutation patterns
32	{0, 16, 8, 24, 4, 20, 12, 28, 18, 2, 26, 10, 22, 6, 30, 14, 17, 1, 25, 9, 21, 5, 29, 13, 3, 19, 11, 27, 7, 23, 15, 31}

6.2.9 Physical channel mapping

- If transport data is less than the number of DPCH bits in a radio frame, dynamic rate matching is applied to ensure that all resource units which belong to one user are either completely filled with data or empty.
- The transmission of the DPCH symbols shall be ON, only if there is data to transmit. If there is no data, the transmission shall be OFF.
- For transport channels not relying on TFCI for transport format detection (blind transport format detection), the positions of the transport channels within the frame should be fixed.
- For transport channels relying on TFCI for transport format detection, the positions of the transport channels should be non-fixed.

6.2.10 Multicode transmission

- For multi-code transmission several codes within one or several timeslots can be allocated for each user independently for uplink and downlink. Those resource units are negotiated by means of higher layers at the beginning of a transmission dependent on the individual services.
- For one user within a certain timeslot there is only one midamble code.
- Dependent on the actual amount of data to be transmitted not all resource units which are allocated to a certain link are used continuously. If no data is transmitted within a time slot during a frame, then also the midamble will be omitted.



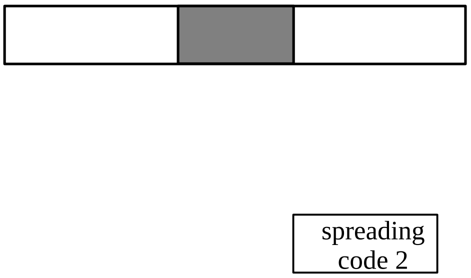


Fig. 6-7 Spreading code and midamble in multi-code transmission

6.2.11 Transport format detection

Transport format detection can be performed both with and without Transport Format Combination Indicator (TFCI). If a TFCI is transmitted, the receiver detects the transport format combination from the TFCI. When no TFCI is transmitted, so called blind transport format detection is used, i.e. the receiver side uses the possible transport format combinations as a priori information.

6.2.11.1 Blind transport format detection

<Editor's note: The status of the support for blind transport format detection with more than one possible transport format in downlink is FFS. BRD may require high implementation complexity in UE.>

Blind transport format detection may be performed in the receiver by trying all possible combinations of the transport format.

6.2.11.2 Transport format detection based on TFCI

6.2.11.2.1 Transport Format Combination Indicator

Transport Format Combination Indicator (TFCI) informs the receiver of the number of bits in each frame of each of the services currently in use. As soon as a certain bit-rate is known, the number of code channels, the spreading factor and the puncturing/repetition rate is immediately known from the rules described in section 6.2.5.

This document therefore only explains the mapping from TFCI bits to TFCI service rate combinations.

A connection may in general include the variable-rate services S_1, S_2, …, S_K. Each service S_i has a set of possible transport format combination indicators $TF_{i,1}$,$TF_{i,2}$,…$TF_{i,Li}$:

S_1: $TF_{1,1}$, … , $TF_{1,L1}$

S_2: $TF_{2,1}$, … , $TF_{2,L2}$

…

S_K: $TF_{K,1}$, … , $TF_{K,LK}$

This gives L=L1xL2x…xLK service rate combinations, and thus it is required that L is less than or equal to 64 with the default TFCI word or 1024 with the extended TFCI word.

These service rate combinations shall be mapped to a certain service rate combination number, *m*, in the following way:

```
For j=K:-1:1,
  SRC[j]= m MOD L[j];
  m = m DIV L[j];
End;
```

From this pseudo-code, given a service rate combination number, i.e. a certain combination of TFCI bits, *m*, SRC contains the rates of each of the K services. The integer values used for *m* shall be consecutive, starting from 0. Note that this code gives the mapping rule from *m* to SRC, i.e. the rule used in the receiving side. The mapping rule from SRC to *m*, i.e. the transmitting side rule, is [TBD].

6.3 Coding for layer 1 control

6.3.1 Coding of transport format combination indicator (TFCI)

<Editor's note:.This text was updated to reflect TDD specific interleaving.>

<WG1 Note: Conforming simulation results should be given.>

Encoding of the TFCI bits depends on the number of them. If there are at most 6 bits of TFCI the channel encoding is done as described in section 6.3.1.1. Correspondingly, if the TFCI word is extended to 7-10 bits the channel encoding is done as explained in the section 6.3.1.2. When decoding, default TFCI words are assumed.

6.3.1.1 Default TFCI word

If the number of TFCI bits is 6 a biorthogonal (32, 6) block code is used. The code words of the biorthogonal (32, 6) code are from two mutually biorthogonal sets, $S_{C_5} = \{C_5(0), C_5(1),..., C_5(31)\}$ and its binary complement, $\overline{S}_{C_5} = \{\overline{C}_5(0), \overline{C}_5(1),..., \overline{C}_5(31)\}$. Words of set S_{C5} are from the level 5 of the code three, which is generated, using the short code generation method defined in chapter 6.2 of 25.213. The mapping of information bits to code words is shown in the Table 6.3.1- 1.

Table 6.3.1- 1 Mapping of information bits to code words for biorthogonal (32, 6) code.

Information bits	Code word
000000	$C_5(0)$
000001	$\overline{C_5(0)}$
000010	$C_5(1)$
...	...
111101	$\overline{C_5(30)}$
111110	$C_5(31)$
111111	$\overline{C_5(31)}$

6.3.1.2 Extended TFCI word

If the number of TFCI bits is 7-10 the TFCI information field is split into two words of length 5 bits as shown in the following formula:

$n := \lfloor \sqrt{TFCI} \rfloor$; n is the largest integer being smaller than or equal to the square root of the transmitted TFCI value.

```
if $TFCI < n^2 + n$
    then $Word1 := n$; $Word2 := TFCI - n^2$
    else $Word2 := n$; $Word1 := n^2 + 2n - TFCI$
```

Both of the words are encoded using biorthogonal (16, 5) block code. The code words of the biorthogonal (16, 5) code are from two mutually biorthogonal sets, $S_{C_4} = \{C_4(0), C_4(1),..., C_4(15)\}$ and its binary complement, $\overline{S}_{C_4} = \{\overline{C}_4(0), \overline{C}_4(1),..., \overline{C}_4(15)\}$. Words of set S_{C_4} are from the level 4 of the code three, which is generated, using the short code generation method defined in chapter 3.2.4.2.2.1.1.1. The mapping of information bits to code words is shown in theTable 6.3.1-2.

Table 6.3.1- 2 Mapping of information bits to code words for biorthogonal (16, 5) code.

Information bits	Code word
00000	$C_4(0)$
00001	$\overline{C_4(0)}$
00010	$C_4(1)$
...	...
11101	$\overline{C_4(14)}$
11110	$C_4(15)$
11111	$\overline{C_4(15)}$

6.3.1.3 Coding of short TFCI lengths

If the number of TFCI bits is 1 or 2, then repetition will be used for coding. In this case each bit is repeated 3 times giving 4-bit transmission for a single TFCI bit and 8-bit transmission for 2 TFCI bits.

If the number of TFCI bits is in the range of 3 to 5, then one word of the biorthogonal (16,5) block code, as described in section 6.3.1.2, will be used.

6.3.1.4 Interleaving of TFCI

In the case of the shortest Transmission Time Interval is 20 ms or above, the redundant TFCI information should be combined in the receiver and may be interleaved over two slots, spaced by a single frame.

The details of the interleaving scheme are FFS.

7 History

Document history		
v0.0.1	1999-02-12	Document created based on the documents Tdoc UMTS (xx.10) V0.5.0 and ARIB Volume 3 Version 1.0-1.0.
V0.0.2	1999-02-18	Document updated based on comments in the TSG-RAN WG1 e-mail reflector.
V0.1.0	1999-02-26	This version was approved at WG1#2 as the initial merge of ARIB and ETSI documents.
V0.1.1	1999-03-03	Document updated based on Adhoc decisions and technical comments made at WG1#2.
V1.0.1	1999-03-23	Document updated based on AdHoc decisions made at WG1#3.
V1.0.2	1999-03-24	Corrections made based on feedback in WG1#3 meeting.
V1.1.0	1999-03-26	This version was approved at WG1#3 meeting.
V1.1.1	1999-04-19	Document updated based on AdHoc decisions at WG1#4.
V1.1.2	1999-04-20	Document updated based on additional text proposals at WG1#4.
V2.0.0	1999-04-20	This version was approved at WG1#4 as an input to TSG RAN meeting.
TS 25.222 V1.0.0	1999-04-22	Noted by TSG-RAN as TS 25.222 V1.0.0
TS 25.222 V1.0.1	1999-06-04	Updated based on AdHoc decisions at WG1#5
TS 25.222 V1.1.0	1999-06-04	This version was approved at WG1#5 meeting.

The editor for TS 25.222 Multiplexing, channel coding and interleaving description (TDD), is:

Jussi Kahtava
Nokia-Japan
Email: jussi.kahtava@nmp.nokia.com

This document is written in Microsoft Word 97.