ETSI TS 101 376-5-3 V1.2.1 (2002-04)

Technical Specification

GEO-Mobile Radio Interface Specifications;
Part 5: Radio interface physical layer specifications;
Sub-part 3: Channel Coding;
GMR-1 05.003

Reference

RTS/SES-001-05003R1

Keywords

GMR, MSS, MES, satellite, GSO, S-PCN, GSM, interface, coding, mobile, radio

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

The attention of ETSI has been drawn to the Intellectual Property Rights (IPRs) listed below which are, or may be, or may become, Essential to the present document. The IPR owner has undertaken to grant irrevocable licences, on fair, reasonable and non-discriminatory terms and conditions under these IPRs pursuant to the ETSI IPR Policy. Further details pertaining to these IPRs can be obtained directly from the IPR owner.

The present IPR information has been submitted to ETSI and pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

IPRs:

Project	Company	Title	Country of Origin	Patent n°	Countries Applicable
TS 101 376 V1.1.1	Digital Voice Systems Inc		US	US 5,226,084	US
TS 101 376 V1.1.1	Digital Voice Systems Inc		US	US 5,715,365	US
TS 101 376 V1.1.1	Digital Voice Systems Inc		US	US 5,826,222	US
TS 101 376 V1.1.1	Digital Voice Systems Inc		US	US 5,754,974	US
TS 101 376 V1.1.1	Digital Voice Systems Inc		US	US 5,701,390	US

IPR Owner: Digital Voice Systems Inc
One Van de Graaff Drive Burlington,
MA 01803
USA

Contact: John C. Hardwick
Tel.: +1 781 270 1030
Fax: +1 781 270 0166

Project	Company	Title	Country of Origin	Patent n°	Countries Applicable
TS 101 376 V1.1.1	Ericsson Mobile Communication	Improvements in, or in relation to, equalizers	GB	GB 2 215 567	GB
TS 101 376 V1.1.1	Ericsson Mobile Communication	Power Booster	GB	GB 2 251 768	GB
TS 101 376 V1.1.1	Ericsson Mobile Communication	Receiver Gain	GB	GB 2 233 846	GB
TS 101 376 V1.1.1	Ericsson Mobile Communication	Transmitter Power Control for Radio Telephone System	GB	GB 2 233 517	GB

IPR Owner: Ericsson Mobile Communications (UK) Limited
The Keytech Centre, Ashwood Way
Basingstoke
Hampshire RG23 8BG
United Kingdom

Contact: John Watson
Tel.: +44 1256 864 821

Project	Company	Title	Country of Origin	Patent n°	Countries Applicable
TS 101 376 V1.1.1	Hughes Network Systems		US	Pending	US

IPR Owner: Hughes Network Systems
11717 Exploration Lane
Germantown, Maryland 20876
USA

Contact: John T. Whelan
Tel: +1 301 428 7172
Fax: +1 301 428 2802

Project	Company	Title	Country of Origin	Patent n°	Countries Applicable
TS 101 376 V1.1.1	Lockheed Martin Global Telecommunic. Inc	2.4-to-3 KBPS Rate Adaptation Apparatus for Use in Narrowband Data and Facsimile Communication Systems	US	US 6,108,348	US
TS 101 376 V1.1.1	Lockheed Martin Global Telecommunic. Inc	Cellular Spacecraft TDMA Communications System with Call Interrupt Coding System for Maximizing Traffic ThroughputCellular Spacecraft TDMA Communications System with Call Interrupt Coding System for Maximizing Traffic Throughput	US	US 5,717,686	US
TS 101 376 V1.1.1	Lockheed Martin Global Telecommunic. Inc	Enhanced Access Burst for Random Access Channels in TDMA Mobile Satellite System	US	US 5,875,182	
TS 101 376 V1.1.1	Lockheed Martin Global Telecommunic. Inc	Spacecraft Cellular Communication System	US	US 5,974,314	US
TS 101 376 V1.1.1	Lockheed Martin Global Telecommunic. Inc	Spacecraft Cellular Communication System	US	US 5,974,315	US
TS 101 376 V1.1.1	Lockheed Martin Global Telecommunic. Inc	Spacecraft Cellular Communication System with Mutual Offset High-Margin Forward Control Signals	US	US 6,072,985	US
TS 101 376 V1.1.1	Lockheed Martin Global Telecommunic. Inc	Spacecraft Cellular Communication System with Spot Beam Pairing for Reduced Updates	US	US 6,118,998	US

IPR Owner: Lockheed Martin Global Telecommunications, Inc.
900 Forge Road
Norristown, PA. 19403
USA

Contact: R.F. Franciose
Tel.: +1 610 354 2535
Fax: +1 610 354 7244

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Satellite Earth Stations and Systems (SES).

The contents of the present document are subject to continuing work within TC-SES and may change following formal TC-SES approval. Should TC-SES modify the contents of the present document, it shall then be republished by ETSI with an identifying change of release date and an increase in version number as follows:

Version 1.m.n

where:

- the third digit (n) is incremented when editorial only changes have been incorporated in the specification;
- the second digit (m) is incremented for all other types of changes, i.e. technical enhancements, corrections, updates, etc.

The present document is part 5, sub-part 3 of a multi-part deliverable covering the GEO-Mobile Radio Interface Specifications, as identified below:

Part 1: "General specifications";

Part 2: "Service specifications";

Part 3: "Network specifications";

Part 4: "Radio interface protocol specifications";

Part 5: "Radio interface physical layer specifications";

Sub-part 1: "Physical Layer on the Radio Path: General Description; GMR-1 05.001";

Sub-part 2: "Multiplexing and Multiple Access; Stage 2 Service Description; GMR-1 05.002";

Sub-part 3: "Channel Coding; GMR-1 05.003";

Sub-part 4: "Modulation; GMR-1 05.004";

Sub-part 5: "Radio Transmission and Reception; GMR-1 05.005";

Sub-part 6: "Radio Subsystem Link Control; GMR-1 05.008";

Sub-part 7: "Radio Subsystem Synchronization; GMR-1 05.010";

Part 6: "Speech coding specifications";

Part 7: "Terminal adaptor specifications".

Introduction

GMR stands for GEO (Geostationary Earth Orbit) Mobile Radio interface, which is used for mobile satellite services (MSS) utilizing geostationary satellite(s). GMR is derived from the terrestrial digital cellular standard GSM and supports access to GSM core networks.

Due to the differences between terrestrial and satellite channels, some modifications to the GSM standard are necessary. Some GSM specifications are directly applicable, whereas others are applicable with modifications. Similarly, some GSM specifications do not apply, while some GMR specifications have no corresponding GSM specification.

Since GMR is derived from GSM, the organization of the GMR specifications closely follows that of GSM. The GMR numbers have been designed to correspond to the GSM numbering system. All GMR specifications are allocated a unique GMR number as follows:

GMR-n xx.zyy

where:

- xx.0yy (z = 0) is used for GMR specifications that have a corresponding GSM specification. In this case, the numbers xx and yy correspond to the GSM numbering scheme.
- xx.2yy (z = 2) is used for GMR specifications that do not correspond to a GSM specification. In this case, only the number xx corresponds to the GSM numbering scheme and the number yy is allocated by GMR.
- n denotes the first (n = 1) or second (n = 2) family of GMR specifications.

A GMR system is defined by the combination of a family of GMR specifications and GSM specifications as follows:

- If a GMR specification exists it takes precedence over the corresponding GSM specification (if any). This precedence rule applies to any references in the corresponding GSM specifications.

NOTE: Any references to GSM specifications within the GMR specifications are not subject to this precedence rule. For example, a GMR specification may contain specific references to the corresponding GSM specification.

- If a GMR specification does not exist, the corresponding GSM specification may or may not apply. The applicability of the GSM specifications is defined in GMR-1 01.201 [7].

1 Scope

The present document specifies the data blocks given to the encryption unit and the mapping onto the free bits of a burst. It includes the specifications for encoding, reordering, interleaving and detailed mapping onto the burst. It does not specify the channel decoding method. The definition is given for each kind of logical channel, starting with the data provided to the channel encoder by the speech coder, the data terminal equipment, or the controller of the Mobile Earth Station (MES).

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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] GMR-1 01.004 (ETSI TS 101 376-1-1): "GEO-Mobile Radio Interface Specifications; Part 1: General specifications; Sub-part 1: Abbreviations and acronyms; GMR-1 01.004".

[2] GMR-1 05.002 (ETSI TS 101 376-5-2): "GEO-Mobile Radio Interface Specifications; Part 5: Radio interface physical layer specifications; Sub-part 2: Multiplexing and Multiple Access; Stage 2 Service Description; GMR-1 05.002".

[3] GMR-1 05.004 (ETSI TS 101 376-5-4): "GEO-Mobile Radio Interface Specifications; Part 5: Radio interface physical layer specifications; Sub-part 4: Modulation; GMR-1 05.004".

[4] GMR-1 05.008 (ETSI TS 101 376-5-6): "GEO-Mobile Radio Interface Specifications; Part 5: Radio interface physical layer specifications; Sub-part 6: Radio Subsystem Link Control; GMR-1 05.008".

[5] GMR-1 05.010 (ETSI TS 101 376-5-7): "GEO-Mobile Radio Interface Specifications; Part 5: Radio interface physical layer specifications; Sub-part 7: Radio Subsystem Synchronization; GMR-1 05.010".

[6] GMR-1 03.020 (ETSI TS 101 376-3-9): "GEO-Mobile Radio Interface Specifications; Part 3: Network specifications; Sub-part 9: Security related Network Functions; GMR-1 03.020".

[7] GMR-1 01.201 (ETSI TS 101 376-1-2): "GEO-Mobile Radio Interface Specifications; Part 1: General specifications; Sub-part 2: Introduction to the GMR-1 Family; GMR-1 01.201".

[8] GMR-1 04.021 (ETSI TS 101 376-4-10): "GEO-Mobile Radio Interface Specifications; Part 4: Radio interface protocol specifications; Sub-part 10: Rate Adaptation on the Access Terminal-Gateway Station Subsystem (MES-GSS) Interface; GMR-1 04.021".

[9] GMR-1 04.008 (ETSI TS 101 376-4-8): "GEO-Mobile Radio Interface Specifications; Part 4: Radio interface protocol specifications; Sub-part 8: Mobile Radio Interface Layer 3 Specifications; GMR-1 04.008".

3 Definitions and abbreviations

For the purposes of the present document, the terms, definitions and abbreviations given in GMR-1 01.004 [1] apply.

4 General

4.1 General organization

Each channel has its own coding and interleaving scheme. However, the channel coding and interleaving are organized in such a way as to allow, as much as possible, a unified decoder structure. The channel coding and interleaving organization is shown in figure 4.1.

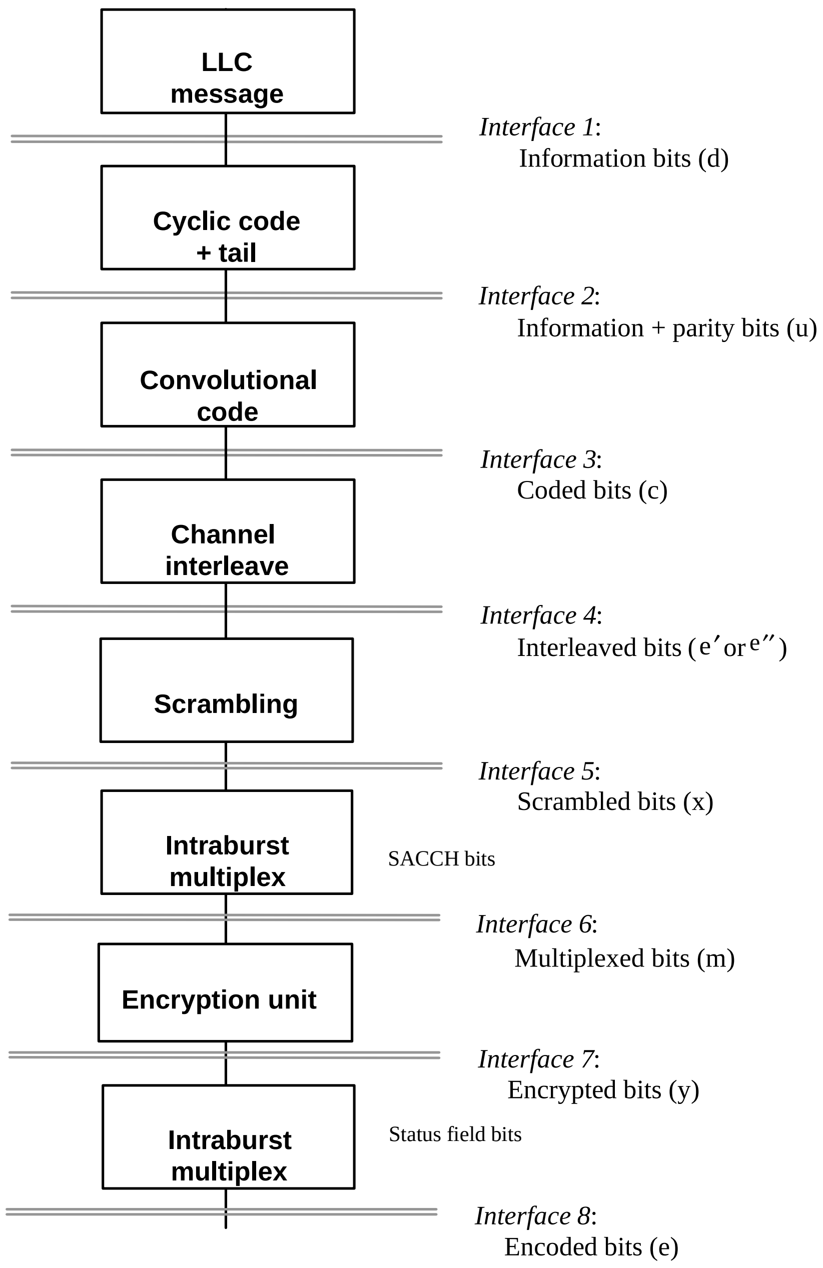

LLC message
Interface 1: Information bits (d)
Cyclic code + tail
Interface 2: Information + parity bits (u)
Convolutional code
Interface 3: Coded bits (c)
Channel interleave
Interface 4: Interleaved bits (e′ or e″)
Scrambling
Interface 5: Scrambled bits (x)
Intraburst multiplex
SACCH bits
Interface 6: Multiplexed bits (m)
Encryption unit
Interface 7: Encrypted bits (y)
Intraburst multiplex
Status field bits
Interface 8: Encoded bits (e)

Figure 4.1: Channel coding and interleaving organization

Each channel uses the following sequence and order of operations:

- the information bits are coded with a systematic block code cyclic redundancy check (CRC), building words of information + parity bits;
- these information + parity bits are encoded with a convolutional code, building the coded bits;
- the coded bits are reordered and potentially interleaved over multiple bursts;
- the interleaved bits are scrambled and, in some cases, multiplexed with other bits (before or after encryption);
- the multiplexed bits are mapped to the physical burst as described in GMR-1 05.002 [2].

The encryption process is described in GMR-1 03.020 [6].

4.2 Naming convention

For ease of understanding, a naming convention for bits is given for use throughout the technical specification.

The naming conventions associated with the interfaces of the reference model in figure 4.1 are summarized in table 4.1. Note that, when the specific data block or burst are clear from context, the corresponding index can be dropped.

Table 4.1: Notation used to specify channel coding operations

Notation	Meaning
d(k)	Information bits provided by the speech coder, data terminal equipment, or controllers
u(k)	Bits delivered by the CRC encoder unit and presented to the forward error correction (FEC) encoder unit
c(k)	Bits delivered by the FEC encoder unit to the channel interleaving unit
e'(j), e''(j)	Bits delivered by the channel interleaving unit to the scrambler
x(k), x(Bj)	Bits delivered by the scrambler to the first intraburst multiplexing unit
m(k), m(Bj)	Bits delivered by the first intraburst multiplexing unit
s_p(B',j),s_n(B'',j)	Status field bits input to the second intraburst multiplexing unit
y(k), y(Bj)	Encrypted bits delivered by the encryption unit
e(k), e(Bj)	Encoded bits delivered by the coding structure
i, j, k	Indices used to number the bits in a burst or data block
n, B, B', B'	Index used to number the bursts allocated to the channel under consideration

4.3 Parity checking

Let K be the size of a data block {d(0), ..., d(K-1)}. For certain channels, a n-bit cyclic redundancy check (CRC) (where n is 8, 12, or 16 depending on the channel) is applied to the data block for error detection at the receiver. The following CRC generator polynomials are used:

$$g_8(D) = D^8 + D^7 + D^4 + D^3 + D + 1;$$

$$g_{12}(D) = D^{12} + D^{11} + D^3 + D^2 + D + 1;$$

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

Table 4.2 indicates the CRC polynomials used in GMR-1 channels.

Table 4.2: CRC polynomials used in GMR-1

Channel	$g_8(D)$	$g_{12}(D)$	$g_{16}(D)$
BCCH			X
PCH			X
AGCH			X
RACH	X	X	
CBCH			X
SDCCH			X
SACCH			X
FACCH3			X
FACCH6			X
FACCH9			X
TACCH			X
GBCH			X

The parity bits are computed by the customary systematic encoding method for cyclic codes. Define the information word polynomial d(D) as follows:

$$d(D) = d(K\text{-}1)D^{K-1} + d(K\text{-}2)D^{K-2} + \ldots + d(1)D + d(0).$$

The parity polynomial $p(D) = p(0) + p(1)D + p(2)D^2 + \ldots + p(n\text{-}1)D^{n-1}$ is the remainder resulting from the division of $g_n(D)$ into the product $d(D) \times D^n$:

$$p(D) = Re\,m\left\{\frac{d(D)*D^n}{g_n(D)}\right\}.$$

The code word polynomial is:

$$u(D) = d(D)\,.\,D^n + p(D);$$

$$= d(K\text{-}1)D^{K+(n-1)} + d(K\text{-}2)D^{K+(n-2)} + \ldots + d(0)\,D^n;$$

$$+ p(n\text{-}1)D^{(n-1)} + \ldots + p(1)D + p(0).$$

The coefficients of u(D) are passed to the next stage of the encoding process, from the highest power of D to the lowest power of D.

4.4 Convolutional coding

4.4.1 Convolutional encoding (all channels except TCH3)

Convolutional encoding is performed in a block mode using zero-valued tail bits to flush the encoder.

4.4.1.1 Rate 1/2 convolutional code

The Rate 1/2 convolutional code of constraint length 5 is defined by the following generator polynomials:

$$g_0(D) = 1 + D^3 + D^4;$$

$$g_1(D) = 1 + D + D^2 + D^4.$$

The input data block {u(0), u(1), ..., u(K-1)} to be encoded is first extended with tail bits so that u(k) = 0 for. The coded bits are then defined by the following set of linear equations:

For k = 0, ... , K+3

$c(2k) = u(k) \oplus u(k-3) \oplus u(k-4)$, where $\oplus$ denotes modulo-2 addition;

$c(2k+1) = u(k) \oplus u(k-1) \oplus u(k-2) \oplus u(k-4)$.

This results in a block of coded bits *{c(0), c(1), c(2), ..., c(2K+7)}*.

4.4.1.2 Rate 1/4 convolutional code

The Rate 1/4 convolutional code is of constraint length 5 defined by the following generator polynomials:

$$g_0(D) = 1 + D^3 + D^4;$$

$$g_1(D) = 1 + D + D^2 + D^4;$$

$$g_2(D) = 1 + D^2 + D^4;$$

$$g_3(D) = 1 + D + D^2 + D^3 + D^4.$$

The input data block {u(0), u(1), ..., u(K-1)} to be encoded is first extended with tail bits so that u(k) = 0 for. The coded bits are then defined by the following set of linear equations:

For k = 0, ... , K+3

$c(4k) = u(k) \oplus u(k-3) \oplus u(k-4)$), where $\oplus$ denotes modulo-2 addition;

$c(4k+1) = u(k) \oplus u(k-1) \oplus u(k-2) + \oplus u(k-4)$;

$c(4k+2) = u(k) \oplus u(k-2) \oplus u(k-4)$;

$c(4k+3) = u(k) \oplus u(k-1) \oplus u(k-2) \oplus u(k-3) \oplus u(k-4)$.

This results in a block of coded bits *{c(0), c(1), c(2), ..., c(4K+15)}*.

4.4.1.3 Rate 1/3 convolutional code

The Rate 1/3 convolutional code is of constraint length 5 defined by the following generator polynomials:

$$g_0(D) = 1 + D^2 + D^4;$$

$$g_1(D) = 1 + D + D^3 + D^4;$$

$$g_2(D) = 1 + D + D^2 + D^3 + D^4.$$

The input data block {u(0), u(1), ..., u(K-1)} to be encoded is first extended with tail bits so that u(k) = 0 for. The coded bits are then defined by the following set of linear equations:

For k = 0, ... , K+3

$c(3k) = u(k) \oplus u(k-2) \oplus u(k-4)$), where $\oplus$ denotes modulo-2 addition;

$c(3k+1) = u(k) \oplus u(k-1) \oplus u(k-3) + \oplus u(k-4)$;

$c(3k+2) = u(k) \oplus u(k-1) \oplus u(k-2) \oplus u(k-3) \oplus u(k-4)$.

This results in a block of coded bits *{c(0), c(1), c(2), ..., c(3K+11)}*.

4.4.1.4 Rate 1/5 convolutional code

The Rate 1/5 convolutional code is of constraint length 5 defined by the following generator polynomials:

$$g_0(D) = 1 + D^2 + D^4;$$

$$g_1(D) = 1 + D + D^3 + D^4;$$

$$g_2(D) = 1 + D + D^2 + D^3 + D^4;$$

$$g_3(D) = 1 + D^2 + D^3 + D^4;$$

$$g_4(D) = 1 + D + D^2 + D^4.$$

The input data block {u(0), u(1), ..., u(K-1)} to be encoded is first extended with tail bits so that u(k) = 0 for $K \leq k \leq K + 3$. The coded bits are then defined by the following set of linear equations:

For k = 0, ... , K+3

$c(5k) = u(k) \oplus u(k-2) \oplus u(k-4))$, where $\oplus$ denotes modulo-2 addition;

$c(5k+1) = u(k) \oplus u(k-1) \oplus u(k-3) + \oplus u(k-4)$;

$c(5k+2) = u(k) \oplus u(k-1) \oplus u(k-2) \oplus u(k-3) \oplus u(k-4)$;

$c(5k+3) = u(k) \oplus u(k-2) \oplus u(k-3) + \oplus u(k-4)$;

$c(5k+2) = u(k) \oplus u(k-1) \oplus u(k-2) \oplus u(k-4)$.

This results in a block of coded bits *{c(0), c(1), c(2), ..., c(5K+19)}.*

4.4.2 Convolutional encoding for TCH3

Unlike the convolutional coding scheme employed for the other channels, the convolutional coding for TCH3 uses a circular encoding method (tail-biting) to avoid the overhead introduced by the tail bits otherwise.

The Rate 1/2 convolutional code of constraint length 7 is used. This code is defined by the following generator polynomials:

$$g_0(D) = 1 + D^2 + D^3 + D^5 + D^6;$$

$$g_1(D) = 1 + D + D^2 + D^3 + D^6.$$

The encoder is initialized with bits {u(K-1), u(K-2), …, u(K-6)} from the input data block {u(0), u(1), …, u(K-1)} to be encoded; (bit u(K-1) is placed in the register D^1 and bit u(K-6) is placed in the register D^6). The coded bits are then defined by the following set of linear equations:

For k = 0, …, K-1

$c(2k) = u(k) \oplus u(k-2) \oplus u(k-3) \oplus u(k-5) \oplus u(k-6)$, where $\oplus$ denotes modulo-2 addition;

$c(2k+1) = u(k) \oplus u(k-1) \oplus u(k-2) \oplus u(k-3) \oplus u(k-6)$.

This results in a block of coded bits {c(0), c(1), …, c(2K-1)}.

4.4.3 Viterbi decoder for TCH3

The convolutional encoder for TCH3 uses a circular encoding method. Therefore, the final state, the state at the last symbol interval (48th in case of TCH3), is not fixed. This requires that the Viterbi decoder perform trellis expansion over a total of 90 symbol intervals, even though there are only 48 input symbols. The voice model parameters representing the fundamental frequency, gain and voicing decisions typically change slowly. At three points in the trellis, the state is dominated by these model parameters and the metrics are adjusted at these points in order to decrease the likelihood of a path being chosen that represents a large change in the voice model parameters.

4.5 Puncturing

The number of available free bits on a burst may not be sufficient for the transmission of all coded bits output by the convolutional encoder. In this case, selected coded bits are punctured and not processed for transmission. The coded bits to be punctured are specified by channel-dependent puncturing masks. These masks take the form of an $n \times L$ binary array, in which the ith row applies to the coded bits produced by the $g_i(\mathrm{D})$ generator polynomial, $i = 0, ..., n$-1. A 1-entry in the mask denotes that the corresponding coded bit is transmitted, while a 0-entry denotes a punctured bit. The parameter L denotes the period of the puncturing pattern. If the period is less than the total number of information plus tail bits input to the encoder, the puncturing mask is reapplied on a periodic basis. In some instances, prefix and suffix masks are applied at the beginning and end of the burst, respectively, to facilitate matching the puncturing to the available free bits.

The puncturing masks used in GMR-1 for the Rate 1/2, Rate 1/3 and Rate 1/5 convolutional code are listed in tables 4.3, 4.4 and 4.5 respectively. The identifier P(r; L) denotes the preferred puncturing mask in which r coded bits are punctured every L input bits to the convolutional encoder. The time-reversed version of this mask is denoted by P*(r; L).

Table 4.3: GMR-1 puncturing masks for the rate 1/2 convolutional code

Identifier	Mask	Remark
P(2;3)	$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 3/4.
P(2;5)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 5/8.
P*(2;5)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$	Time-reversal of the puncturing mask P(2;5).
P(3;11)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 11/19.
P(4;12)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 12/20.
P*(4;12)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$	Time-reversal of the puncturing mask P(4,12).
P(1;2)	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 2/3.

Table 4.4: GMR-1 puncturing masks for the rate 1/3 convolutional code

Identifier	Mask	Remark
P(1;6)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	Puncturing mask that, if applied repetitively for NT6 burst punctures 24 bits giving an effective code rate = 0,3523.
P(2;5)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 5/13.
P(1;5)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 5/14.
P^{*}(1;5)	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	Time-reversal of the puncturing mask P(1,5).

Table 4.5: GMR-1 puncturing masks for the rate 1/5 convolutional code

Identifier	Mask	Remark
P(2;3)	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 3/13.
P(5;3)	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$	Puncturing mask that, if applied repetitively, produces effective code rate = 3/10.
P*(5;3)	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$	Time-reversal of the puncturing mask P(1,5).

4.6 Golay encoding

The (24,12) Golay code is used to encode blocks of 12 information bits into codewords of length 24 bits. The systematic encoding of 12 information bits {u(0), u(1), ..., u(11)} is accomplished via the generator matrix ***G*** of figure 4.2.

If $\underline{u}$ = [u(0), u(1), ..., u(11)] denotes the row vector of information bits to be encoded, the output codeword is given by the matrix product $\underline{c} = \underline{u}\,\boldsymbol{G}$. This results in a block of coded bits {c(0), c(1), ..., c(23)}.

	Binary representation	Hexadecimal
	1000 0000 0000 1111 1010 0100	(0x800fa4)
	0100 0000 0000 1100 1100 1110	(0x400cce)
	0010 0000 0000 1010 1001 1101	(0x200a9d)
	0001 0000 0000 1001 1111 1000	(0x1009f8)
	0000 1000 0000 0110 1011 1010	(0x0806ba)
	0000 0100 0000 0101 1110 0011	(0x0405e3)
G =	0000 0010 0000 0011 1101 0110	(0x0203d6)
	0000 0001 0000 1110 0110 1001	(0x010e69)
	0000 0000 1000 1101 0101 0101	(0x008d55)
	0000 0000 0100 0000 0111 1111	(0x00407f)
	0000 0000 0010 1011 0011 0011	(0x002b33)
	0000 0000 0001 0111 0000 1111	(0x00170f)

Figure 4.2: Systematic generator matrix for (24,12) Golay code

4.7 Reed-Solomon encoding

The systematic (15,9) Reed-Solomon (RS) code generated over the Galois field GF(2^4) is used to encode blocks of 9 information symbols of 4 bits each into a block of 15 coded symbols. The Galois field GF(2^4) is defined with α as it primitive element under a irreducible polynomial:

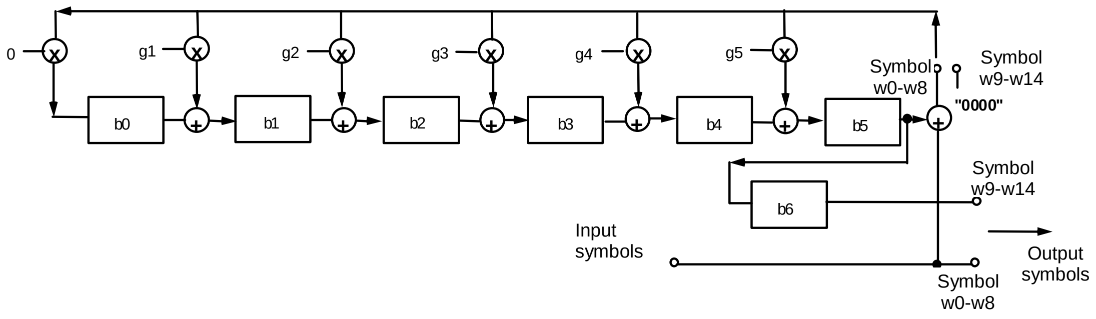
$$p(X) = 1 + X + X^4.$$

The generator polynomial G(X) is defined as:

$$G(X) = g_0 + g_1X + g_2X^2 + g_3X^3 + g_4X^4 + g_5X^5 + X^6.$$

4.7.1 Encoder

The (15, 9) Reed-Solomon encoder computes and adds 6 error correction symbols (parity symbols) to a block of 9 information symbols. The encoding algorithm can be represented by the feedback shift register shown in figure 4.3.



NOTE: All symbols, path, registers and coef. are 4 bits.

Figure 4.3: Reed-Solomon encoder block diagram

A pair of switches direct the input message symbols through to the output and also into the parity generator for the first 9 symbols. On the 10th through the 15th symbols, the switch directs 0s into the feedback path and shifts the error correction symbols out of the register to the output. The addition operation "+" and the multiplication operation "x" indicate the special GF(2^4) arithmetic described in clause 4.7.2. The tap coefficients $g_0, \ldots, g_5$ are the coefficients of the generator polynomial G(X) and are given in table 4.8.

4.7.2 Galois field arithmetics

The field elements of GF(2^4) are expressed as powers of a primitive element α. Each element also corresponds to a polynomial with binary coefficients. It is the four tuple binary coefficient that is used in the shift register. Table 4.6 shows GF(2^4) elements as a power of α, as a polynomial and as the decimal representation of the polynomial.

Table 4.6: GF(2^4) field elements

GF(2^4) field element	Binary polynomial representation	Decimal representation of field element
0	0000	0
$\alpha^0 = \alpha^{15} = 1$	1000	8
α^1	0100	4
α^2	0010	2
α^3	0001	1
α^4	1100	12
α^5	0110	6
α^6	0011	3
α^7	1101	13
α^8	1010	10
α^9	0101	5
α^{10}	1110	14
α^{11}	0111	7
α^{12}	1111	15
α^{13}	1011	11
α^{14}	1001	9

Addition is performed by bit-by-bit exclusive OR between symbols.

EXAMPLE:

$$\begin{array}{l} \alpha^{6}\ 0011 \\ \underline{+\alpha^{5}\ 0110} \\ \alpha^{9}\ 0101 \end{array}$$

In implementing the adder, it is not necessary to know the exponential form of the field element. The multiplier, however, makes use of the exponential form and the rule,

$$\alpha^{i} \text{ x } \alpha^{j} = \alpha^{(i+j) mod 15}.$$

Using the field element table above, a lookup table of exponents for each field element may be constructed as table 4.7.

Table 4.7: Power of α lookup table

Field Element	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Power of alpha	-	3	2	6	1	9	5	11	15	14	8	13	4	7	10	12

To multiply two field elements:

If either element is zero, the result is zero.

Otherwise:

- look up the exponents in table 4.7;
- add the exponents mod 15;
- look up the field element from table 4.6.

Multiplication is required between the feedback and each tap coefficient. The tap coefficients g0, g1, etc. are the coefficients of the generator polynomial G(X), which for this code is given in table 4.8.

Table 4.8: Generator coefficients

Coefficient	**Field element**	**Field value (n tuple)**	**Decimal**
g_0	α^{6}	0011	3
g_1	α^{9}	0101	5
g_2	α^{6}	0011	3
g_3	α^{4}	1100	12
g_4	α^{14}	1001	9
g_5	α^{10}	1110	14

4.7.3 Encoder feedback register operation

The register (b_0-b_6) is initially loaded with zeros. When the message symbol (w0-w8) arrives, it is delivered to the output and at the same time "added" to b_5 to become the common feedback symbol. At each tap, the feedback symbol is multiplied by the tap coefficient (g_0-g_5). The product is then available at each tap for the shift operation. The shift begins at the output end, using the previous register values. Symbol $_5$ is shifted to b_6, b_4 is added to the feedback result and stored in b_5. Continue until finally b_0 is loaded with its new value.

For output symbols w9-w15, a zero is asserted on the feedback and the register contents are shifted to the output. The shift occurs before the output is taken. An alternate implementation would be to simply read out the register in order b_5-b_0 for the w_9-w_{14} symbol outputs.

4.8 Interleaving

Intraburst and interburst interleaving schemes are based on block interleaving methods with pseudorandom permutations and are channel dependent.

4.8.1 Intraburst interleaving

Intraburst interleaving is performed by mapping the block of the coded bits into a N × 8 matrix, interchanging the columns using the pseudorandom permutation factor of 5 and reading out blocks of data by columns. Matrix dimension N is channel dependent and depends on the number of coded bits.

When columns are interchanged the index of the matrix element (i, j) changes to (i, j_p), where $j_p = (j \times 5) \bmod 8$.

4.8.2 Interburst interleaving

The following description illustrates the interburst interleaving function as specified for TCH6 and TCH9.

When burst N, the output of the intraburst interleaver, arrives to the interburst interleaver, it is stored in the M × K array together with the M-1 previously arrived bursts. For M = 3 (depth 3 interleaver), the output of the inter-burst interleaver is a sequence E:

$$E = \{E_0, E_1, \ldots, E_k, \ldots, E_K\} = \{N_0, (N-1)_1, (N-2)_2, N_3, (N-1)_4, (N-2)_5, \ldots, (N-k')_k, \ldots\};$$

where k = 0, …, K refers to the intra-burst bit number and k' = k mod 3.

The array for a depth 3 interleaver when burst N arrives is shown below. The first few elements of the output sequence E are shown in bold.

N_0	N_1	N_2	**N_3**	N_4	N_5	..	..	N_k	..	..	..	N_K
$(N-1)_0$	**$(N-1)_1$**	$(N-1)_2$	$(N-1)_3$	**$(N-1)_4$**	$(N-1)_5$	..	..	$(N-1)_k$	..	..	..	$(N-1)_K$
$(N-2)_0$	$(N-2)_1$	**$(N-2)_2$**	$(N-2)_3$	$(N-2)_4$	**$(N-2)_5$**	..	..	$(N-2)_k$	..	..	..	$(N-2)_K$

When burst N+1 arrives to the interburst interleaver, the interleaver array looks like the one shown below. The first few elements of the output sequence (E+1) are shown in bold.

$(N+1)_0$	$(N+1)_1$	$(N+1)_2$	**$(N+1)_3$**	$(N+1)_4$	$(N+1)_5$	..	..	$(N+1)_k$	..	..	..	$(N+1)_K$
N_0	**N_1**	N_2	N_3	**N_4**	N_5	..	..	N_k	..	..	..	N_K
$(N-1)_0$	$(N-1)_1$	**$(N-1)_2$**	$(N-1)_3$	$(N-1)_4$	**$(N-1)_5$**	..	..	$(N-1)_k$	..	..	..	$(N-1)_K$

When burst N+2 arrives to the interburst interleaver, the interleaver array looks like the one shown below. The first few elements of the output sequence (E+2) are shown in bold.

$(N+2)_0$	$(N+2)_1$	$(N+2)_2$	**$(N+2)_3$**	$(N+2)_4$	$(N+2)_5$	..	..	$(N+2)_k$	..	..	..	$(N+2)_K$
$(N+1)_0$	**$(N+1)_1$**	$(N+1)_2$	$(N+1)_3$	**$(N+1)_4$**	$(N+1)_5$	..	..	$(N+1)_k$	..	..	..	$(N+1)_K$
N_0	N_1	**N_2**	N_3	N_4	**N_5**	..	..	N_k	..	..	..	N_K

The transmission of the burst N is completed at the time when N+2 burst arrives to the interburst interleaver, i.e. two bursts later.

When burst E arrives to the interburst deinterleaver, it is stored in the M × K array together with the (M -1) previously arrived sequences.

For M = 3, the output of interburst deinterleaver is a sequence D:

$$D = \{D_0, D_1, \ldots, D_k, \ldots, D_K\} = \{(E-2)_0, (E-1)_1, E_2, (E-2)_3, (E-1)_4, E_5, \ldots, (E-k'')_k, \ldots\};$$

where k = 0, …, K and k" = (2-k) mod 3.

The array for a depth 3 de-interleaver when burst E arrives is shown below. The first few elements of the output sequence D are shown in bold.

E_0	E_1	**E_2**	E_3	E_4	**E_5**	..	..	E_k	..	..	..	E_K
$(E-1)_0$	**$(E-1)_1$**	$(E-1)_2$	$(E-1)_3$	**$(E-1)_4$**	$(E-1)_5$	..	..	$(E-1)_k$	..	..	..	$(E-1)_K$
$(E-2)_0$	$(E-2)_1$	$(E-2)_2$	**$(E-2)_3$**	$(E-2)_4$	$(E-2)_5$	..	..	$(E-2)_k$	..	..	..	$(E-2)_K$

The deinterleaver output is a sequence D with the elements D_k, where:

$$D_k = (E-k'')_k = ((N-k')-k'')_k = (N - k \bmod 3 - (2-k) \bmod 3)_k = (N-2)_k.$$

When burst (E+1) arrives to the interburst de-interleaver, the deinterleaver array looks like the one shown below. The first few elements of the output sequence (D+1) are shown in bold.

$(E+1)_0$	$(E+1)_1$	**$(E+1)_2$**	$(E+1)_3$	$(E+1)_4$	**$(E+1)_5$**	..	..	$(E+1)_k$	..	..	..	$(E+1)_K$
E_0	**E_1**	E_2	E_3	**E_4**	E_5	..	..	E_k	..	..	..	E_K
$(E-1)_0$	$(E-1)_1$	$(E-1)_2$	**$(E-1)_3$**	$(E-1)_4$	$(E-1)_5$	..	..	$(E-1)_k$	..	..	..	$(E-1)_K$

The de-interleaver output is a sequence (D+1) with the elements $(D+1)_k$, where:

$$(D+1)_k = ((E+1)-k'')_k = (((N+1)-k')-k'')_k = ((N+1) -k \bmod 3 - (2-k) \bmod 3)_k = (N-1)_k.$$

When burst (E+2) arrives to the interburst deinterleaver, the deinterleaver array looks like the one shown below. The first few elements of the output sequence (D+2) are shown in bold.

$(E+2)_0$	$(E+2)_1$	**$(E+2)_2$**	$(E+2)_3$	$(E+2)_4$	**$(E+2)_5$**	..	..	$(E+2)_k$	..	..	..	$(E+2)_K$
$(E+1)_0$	**$(E+1)_1$**	$(E+1)_2$	$(E+1)_3$	**$(E+1)_4$**	$(E+1)_5$	..	..	$(E+1)_k$	..	..	..	$(E+1)_K$
E_0	E_1	E_2	**E_3**	E_4	E_5	..	..	E_k	..	..	..	E_K

The deinterleaver is a sequence (D+2) with the elements $(D+2)_k$, where:

$$(D+2)_k = ((E+2)-k'')_k = (((N+2)-k')-k'')_k = ((N+2) -k \bmod 3 - (2-k) \bmod 3)_k = N_k.$$

In other words, the sequence (D+2) is the burst N recovered with a two-burst delay.

4.9 Scrambling

The scrambler adds a binary pseudo-noise sequence (masking sequence) to the input bit stream in order to randomize the number of 0s and 1s in the output bit stream. Addition is performed modulo 2. The scrambler is also a descrambler since the original bit stream is restored when the scrambling process is repeated.

The masking sequence is generated by a linear feedback shift register with connection polynomial $h(D) = 1 + D + D^{15}$. The block diagram is shown in figure 4.4. Initial contents of the shift register are specified by the polynomial $i(D) = 1 + D + D^3 + D^6 + D^8 + D^{10} + D^{11} + D^{14}$. At the start of each scrambling/descrambling operation, the shift register is always reset to this prescribed initial state.

The first value of the masking sequence is $\mu(0) = 0$, being the modulo-2 sum of the first and last elements in the shift register after initialization. For each input bit, the shift register is clocked once to generate the next element of the masking sequence.

Let $\mu(k)$ denote the value of the masking sequence after the kth clocking of the shift register, where $\mu(0)$ is the value corresponding to the initial shift register state. When the scrambler is presented with an input block $\{\varepsilon(0), ..., \varepsilon(K-1)\}$ of length K, it produces the output block $\{x(0), ..., x(K-1)\}$, according to the rule:

$$x(k) = [\varepsilon(k) + \mu(k)] \bmod 2.$$

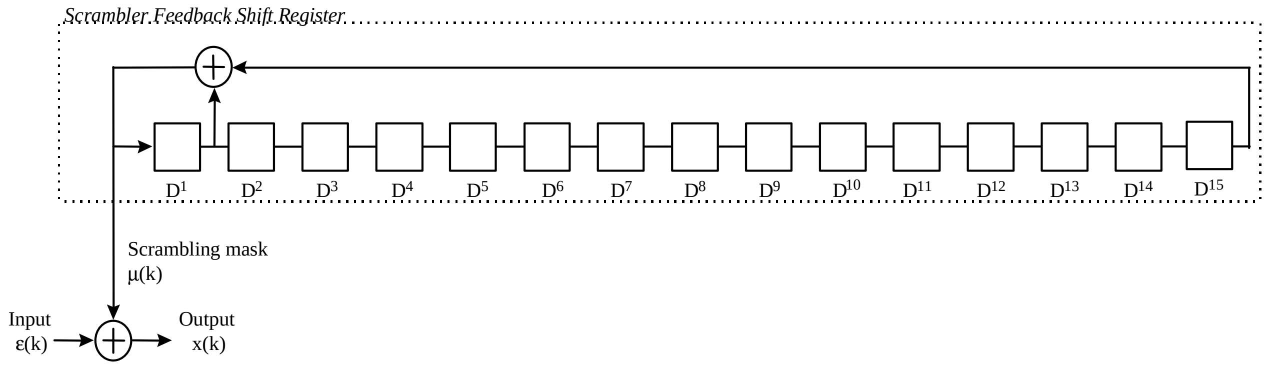

Scrambler Feedback Shift Register
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
Scrambling mask μ(k)
Input ε(k)
Output x(k)

Figure 4.4: Scrambler/descrambler block diagram

5 Traffic channels

5.1 Traffic channel-3 (TCH3)

The user unit delivers to the encoder a bit stream organized into blocks of 80 information bits {d(0), …, d(79)}.

5.1.1 Channel coding

The block of first 48 information bits {d(0), …, d(47)} is encoded via the Rate 1/2 convolutional code described in clause 4.4.2. Convolutional encoding produces a block of 96 bits {b(0), …, b(95)}.

Puncturing is performed using mask P(1;2) shown in table 4.3. The puncturing mask is applied repetitively 24 times to produce a block of 72 punctured bits {b'(0), …, b'(71)}.

The encoder unit outputs a block of 104 coded bits {c(0), …, c(103)} to the interleaving unit, where:

c(k) = b' (k),	for k = 0, …, 71;
c(k+24) = d(k),	for k = 48, …, 79.

5.1.2 Interleaving

The block of 104 coded bits {c(0), …, c(103)} is interleaved as following:

$c'(i,j) = c(k)$, where $i = k \bmod 24$, $j = \mathrm{INT}(k/24)$ as $k = 0, \ldots, 103$;

$e'(k) = c'(i,j)$, where $k = j + 5 \times i$, $j = 0, \ldots, 4$ for $i = 0, \ldots, 7$;

and $k = j + 4 \times i + 8$, $j = 0, \ldots, 3$ for $i = 8, \ldots, 23$.

Each two blocks of interleaved bits e'(k), produced by encoding of a pair of information blocks d(k), are mapped to the output block of 208 bits {e''(0), …, e''(207)}. Let $e'_0(k)$ be the first block and $e'_1(k)$ be the second block of interleaved bits, then depending on the predefined parameter m (m = 0 or 1):

for m = 0

$e''(k) = e'_0(k)$, where $k = 0, \ldots, 103$;

$e''(k+104) = e'_1(k)$, where $k = 0, \ldots, 103$.

for $m = 1$

$e''(2k) = e'_0(k)$, where $k = 0, \ldots, 103$;

$e''(2k + 1) = e'_1(k)$, where $k = 0, \ldots, 103$.

5.1.3 Scrambling, multiplexing and encryption

The block {e''(0), …, e''(207)} is scrambled, as described in clause 4.9, to produce the output block {x(0), …, x(207)}. The 208 scrambled bits x(k) become multiplexed bits:

$m(k) = x(k)$, for $k = 0, \ldots, 207$.

The 208 multiplexed bits m(k) are encrypted, as described in clause 8, to produce a block of 208 encrypted bits y(k), that is, then, multiplexed with status field bits as described in clause 7.3.2.1. The resultant block of encoded bits {e(0), …, e(211)} is mapped to the NT3 burst for speech as described in GMR-1 05.002 [2].

5.2 Traffic channel-6 (TCH6)

5.2.1 Channel coding

5.2.1.1 Coding for 2,4 kbit/s fax

The definition of a 3,6 kbit/s radio interface rate data flow for transparent fax service at 2,4 kbit/s is given in GMR-1 04.021 [8].

The user unit delivers to the encoder a bit stream organized in blocks of 36 information bits (data frames) every 10 ms. Four such blocks are dealt with together in the coding process, {d(0), …, d(143)}. The 144 user bits are encoded via the Rate 1/3 convolutional code specified in clause 4.4.1.3. This action results in a block of 444 coded bits {b(0), …, b(443)}.

Puncturing is performed using mask P(1;6), shown in table 4.4. Puncturing mask P(1;6) is applied repetitively 24 times. The result is a block of 420 coded bits {c(0), …, c(419)}.

5.2.1.2 Coding for 2,4 kbit/s data and 4,8 kbit/s fax/data

The definition of a 6 kbit/s radio interface rate data flow for 4,8 kbit/s transparent fax service and 2,4 and 4,8 kbit/s non-transparent data services are given in GMR-1 04.021 [8].

The user unit delivers to the encoder a bit stream organized in blocks of 60 information bits (data frames) every 10 ms. Four such blocks are dealt with together in the coding process {d(0),..., d(239)}. For non-transparent services those four blocks shall align with one 240-bit RLP frame.

The 240 user bits are encoded via the Rate 1/2 convolutional code specified in clause 4.4.1.1. This action results in a block of 488 coded bits {b(0), …, b(487)}.

Puncturing is performed using masks P(4;12), $P^*(4;12)$ and P(3;11), shown in table 4.3. The puncturing mask P(4;12) is applied once at the beginning of the burst, the puncturing mask $P^*(4;12)$ is applied once at the end of the burst and the puncturing mask P(3;11) is applied repetitively (20 times) during the middle portion of the burst. The result is a block of 420 coded bits {c(0), …, c(419)}.

5.2.2 Interleaving

The first 416 punctured coded bit c(k) are then intraburst-interleaved by mapping into a 52 × 8 matrix by rows and then permuting the columns and finally reading out block of data by columns.

$c'(i, j) = c(k),$ where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, \ldots, 415$;

$e'(k) = c'(i, j),$ where $k = i + 52 \times j$, $i = 0, \ldots, 51$ and $j = 0, \ldots, 7$.

The last 4 c(i) bits are then appended to complete the e′(k) sequence:

$e'(k+416) = c(k+416)$ for $k = 0, \ldots, 3$.

The sequence e′(k) of interleaved bits is then interburst-interleaved using a three-burst interleaver. This e′(k) sequence is first mapped into a 3 × 420 matrix where each row corresponds to a burst and the columns contain the bits from the sequence e′(k). Let n (n = 0, ...) be the current TCH6 burst number.

$c''(i, j) = e'(k),$ where $i = n \bmod 3$ and $j = k$, for $k = 0, .. 419$;

$e''(k) = c''(i, j),$ where as $k = 0, .. 419$, $j = 0, \ldots, 419$;

and $i = ((n \bmod 3) - (j \bmod 3) + 3) \bmod 3$.

The TCH6 burst number is incremented with the transmission of each TCH6 burst. When TCH6 burst is stolen by the FACCH6, the TCH6 burst number is not incremented.

5.2.3 Scrambling, multiplexing and encryption

The block {e"(0), ..., e"(419)} is scrambled, as described in clause 4.9, to produce the output block {x(0), ..., x(419)}. The 420 scrambled bits x(k) are multiplexed with the SACCH, as described in clause 7.1, to produce a block of multiplexed bits m(k). The 430 multiplexed bits m(k) are encrypted, as described in clause 8, to produce a block of 430 encrypted bits y(k) that is then, multiplexed with the status field bits as described in clause 7.3.1. The resultant block of encoded bits {e(0), …, e(433)} is mapped to the NT6 burst as described in GMR-1 05.002 [2].

5.3 Traffic channel-9 (TCH9)

5.3.1 Channel coding

5.3.1.1 Coding for 2,4 kbit/s fax

The definition of a 3,6 kbit/s radio interface rate data flow for transparent fax service at 2,4 kbit/s is given in GMR-1 04.021 [8].

The user unit delivers to the encoder a bit stream organized in blocks of 36 information bits (data frames) every 10 ms. Four such blocks are dealt with together in the coding process, {d(0), d(1),..., d(143)}.

The 144 user bits are encoded via the Rate 1/5 convolutional code specified in clause 4.4.1.4. This action results in a block of 740 coded bits {b(0), …, b(739)}.

Puncturing is performed using masks P(5;3), P^*(5;3) and P(2;3), shown in table 4.5. The puncturing mask P(5;3) is applied once at the beginning of the burst, the puncturing mask P^*(5;3) is applied once at the end of the burst and the puncturing mask P(2;3) is applied repetitively (41 times) during the middle portion of the burst. The result is a block of 648 coded bits {c(0), …, c(647)}.

5.3.1.2 Coding for 4,8 kbit/s fax

The definition of a 6 kbit/s radio interface rate data flow for 4,8 kbit/s transparent fax service is given in GMR-1 04.021 [8].

The user unit delivers to the encoder a bit stream organized in blocks of 60 information bits (data frames) every 10 ms. Four such blocks are dealt with together in the coding process {d(0),..., d(239)}. For non-transparent services those four blocks shall align with one 240-bit RLP frame. The 240 user bits are encoded via the Rate 1/3 convolutional code specified in clause 4.4.1.3. This action results in a block of 732 coded bits {b(0), …, b(731)}.

Puncturing is performed using masks P(1;5), P^*(1;5) and P(2;5), which are shown in table 4.4. The puncturing mask P(1;5) is applied once at the beginning of the burst, the puncturing mask P^*(1;5) is applied once at the end of the burst and the puncturing mask P(2;5) is applied repetitively (41 times) during the middle portion of the burst. The result is a block of 648 coded bits {c(0), …, c(647)}.

5.3.1.3 Coding for 9,6 kbit/s fax/data

The definition of a 12 kbit/s radio interface rate data flow for 9,6 kbit/s transparent fax service and 9,6 kbit/s non-transparent data service are given in GMR-1 04.021 [8].

The user unit delivers to the encoder a bit stream organized in blocks of 60 information bits (data frames) every 5 ms. Eight such blocks are dealt with together in the coding process {d(0),..., d(479)}. For non-transparent services those eight blocks shall align with two 240-bit RLP frames.

The 480 user bits are encoded via the Rate 1/2 convolutional code specified in clause 4.4.1.1. This action results in a block of 968 coded bits {b(0), …, b(967)}.

Puncturing is performed using masks P(2;5), P^*(2;5) and P(2;3), which are shown in table 4.3. The puncturing mask P(2;5) is applied once at the beginning of the burst, the puncturing mask P^*(2;5) is applied once at the end of the burst and the puncturing mask P(2;3) is applied repetitively (158 times) during the middle portion of the burst. The result is a block of 648 coded bits {c(0), …, c(647)}.

5.3.2 Interleaving

These 648 punctured coded bit c(k) are then intraburst-interleaved by mapping into a 81 × 8 matrix by rows and then permuting the columns and finally reading out block of data by columns.

$c'(i, j) = c(k)$, where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, .. 647$;

$e'(k) = c'(i, j)$, where $k = i + 81 \times j$, $i = 0, .. 80$ and $j = 0, .. 7$.

This sequence e′(k) of interleaved bits is then interburst-interleaved using a 3 burst interleaver. This e′(k) sequence is first mapped into a 3 × 648 matrix where each row corresponds to a burst and the columns contain the bits from the sequence e′(k). Let n (n = 0, ...) be the current TCH9 burst number.

$c''(i, j) = e'(k)$, where $i = n \bmod 3$ and $j = k$, for $k = 0, .. 647$;

$e''(k) = c''(i, j)$, where as $k = 0, .. 647$, $j = 0, \ldots, 647$;

and $i = ((n \bmod 3) - (j \bmod 3) + 3) \bmod 3$.

The TCH9 burst number is incremented with the transmission of each TCH9 burst. When TCH9 burst is stolen by the FACCH9, the TCH9 burst number is not incremented.

5.3.3 Scrambling, multiplexing and encryption

The block {e"(0), ..., e"(647)} is scrambled as described in clause 4.9 to produce the output block {x(0), ..., x(647)}. The 648 scrambled bits x(k) are multiplexed with the SACCH, as described in clause 7.1, to produce a block of multiplexed bits m(k). The 658 multiplexed bits m(k) are encrypted, as described in clause 8, to produce a block of 658 encrypted bits y(k) that is then multiplexed with the status field bits, as described in clause 7.3.1. The resultant block of encoded bits {e(0), …, e(661)} is mapped to the NT9 burst as described in GMR-1 05.002 [2].

6 Control channels

6.1 Broadcast control channel (BCCH)

The message delivered to the encoder has a fixed size of 192 information bits {d(0), ..., d(191)}.

6.1.1 Channel coding

A 16-bit CRC is applied to the 192 message bits as specified in clause 4.3. The resultant block of 208 CRC-protected bits {u(0), ..., u(207)} is then encoded via the Rate 1/2 convolutional code as specified in clause 4.4.1.1. Convolutional encoding produces a block of 424 coded bits {c(0),..., c(423)}.

6.1.2 Interleaving

The block of 424 bits are interleaved by first mapping the bits into a 53 × 8 matrix by rows and then permuting the columns and finally reading out blocks of data by columns.

$c'(i,j) = c(k)$, where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, 423$;

$e'(k) = c'(i,j)$, where $k = i + 53 \times j$, $i = 0, ..., 52$ and $j = 0, ..., 7$.

6.1.3 Scrambling and multiplexing

The block {e′(0), ..., e′(423)} is scrambled, as described in clause 4.9, to produce the output block {x(0), ..., x(423)}. The scrambled bits x(k) become the encoded bits e(k):

$e(k) = x(k)$, for $k = 0, ..., 423$.

The block of 424 encoded bits is mapped to a BCCH burst as described in GMR-1 05.002 [2].

6.2 Paging CHannel (PCH)

The message delivered to the encoder has a fixed size of 192 information bits {d(0), ..., d(191)}.

6.2.1 Channel coding

A 16-bit CRC is applied to the 192 message bits as specified in clause 4.3. The resultant block of 208 CRC-protected bits {u(0), ..., u(207)} is then encoded via the Rate 1/2 convolutional code as specified in clause 4.3. Convolutional encoding produces a block of 424 coded bits {c(0),..., c(423)}.

6.2.2 Interleaving

The block of 424 bits are interleaved by first mapping the bits into a 53 × 8 matrix by rows and then permuting the columns and finally reading out blocks of data by columns.

$c'(i,j) = c(k)$, where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, 423$;

$e'(k) = c'(i,j)$, where $k = i + 53 \times j$, $i = 0, ..., 52$ and $j = 0, ..., 7$.

6.2.3 Scrambling and multiplexing

The block of 424 bits are augmented by 8 pad bits as follows:

$e''(0), ..., e''(3) = 0$;

$e''(k+4) = e'(k)$, $k = 0, ..., 423$ and

$e''(428), ..., e''(431) = 0$.

The block {e"(0), ..., e"(431)} is scrambled, as described in clause 4.9, to produce the output block {x(0), ..., x(431)}. The scrambled bits x(k) become the encoded bits e(k):

$e(k) = x(k)$, for $k = 0, 431$.

The resultant 432 encoded bits is mapped to a DC6 burst as described in GMR-1 05.002 [2].

6.3 Access grant channel (AGCH)

The message delivered to the encoder has a fixed size of 192 information bits {d(0), ..., d(191)}.

6.3.1 Channel coding

The coding of the 192 information bits is done as specified for the PCH.

6.3.2 Interleaving

The interleaving is done as specified for the PCH.

6.4 Broadcast alerting channel (BACH)

The message delivered to the encoder has a fixed size of 36 information bits {d(0), …, d(35)}.

6.4.1 Channel coding

A 36-bit block d(k) is multiplexed into 9 4-bit symbols to produce 9 × 4 data blocks D with the elements d(i, j) defined as:

$d(i,j) = d(k)$, where $k = 0, …, 35$, $i = \text{INT}(k/4)$ and $j = k \bmod 4$.

A (15,9) Reed-Solomon code is applied to each 4-bit symbol in the data block D as specified in clause 4.7. Reed-Solomon encoding adds parity check symbols to produce a 15 × 4 data block C with the elements c(i, j). Each 4-bit symbol {c(i,0), …, c(i,3)} is mapped to one BACH burst as described in GMR-1 05.004 [3].

6.5 Random access channel (RACH)

The message delivered to the encoder consists of 16 Class 1 bits $\{d_1(0), ..., d_1(15)\}$ and 123 Class 2 bits $\{d_2(0), ..., d_2(122)\}$.

6.5.1 Channel coding

A 8-bit CRC is applied to the 16 Class 1 bits to form a block of 24 CRC-protected Class 1 bits $\{u_1(0), ..., u_1(23)\}$. A 12-bit CRC is applied to the 123 Class 2 bits to form a block of 135 CRC-protected Class 2 bits $\{u_2(0), ..., u_2(134)\}$. The CRCs are calculated as specified in clause 4.3.

An 8 bit masking function is applied to the 24 Class 1 bits in the following manner. The mask is the string RACH_SB_Mask broadcast in system information as specified in GMR-1 04.008 [9]. Note that the CRC on the Class 2 bits is not masked.

The MES shall be required to apply this mask to Class 1 bits and the GS shall accept RACH bursts with the string SB_Mask applied and the GS should also accept RACH bursts with the string of all zeros applied. Moreover, the GS should not set SB_Mask to all zeros.

The 8 bit mask {m(0), m(1), m(2), m(3),...m(6), m(7)} is XOR 'd with the 8 CRC bits of the message to give us $\{u'_1(0), u'_1(1), u'_1(2)…… u'_1(22), u'_1(23)\}$

Here m(0) is the MSB bit and m(7) is the LSB bit of the mask

Where $\{u'_1(0), u'_1(1), u'_1(2)….u'_1(15)\} = \{u_1(0), u_1(1), u_1(2), ...u_1(15)\}$

And

$u'_1(16) = u_1(16) \oplus m(0)$: where ⊕ denotes modulo 2 addition, i.e. XOR

$u'_1(17) = u_1(17) \oplus m(1)$

…………………………

$u'_1(23) = u_1(23) \oplus m(7)$

Rate 1/4 convolutional encoding is then performed as specified in clause 4.4.1.2 on the data block {d(0), ..., d(158)}, where:

$d(k) = u_2(k)$ for k = 0, ..., 134;

$d(k+135) = u'_1(k)$ for k = 0, ..., 23.

This produces a block of 652 coded bits {b(0), ..., b(651)}.

Puncturing is performed as follows. Coded bits {b(540), …, b(651)} are not punctured.
Puncture masks P(2; 1) = $[1\ 1\ 0\ 0]^T$ is applied repetitively (135 times) to the coded bits {b(0), …, b(539)} to remove the coded bits produced by $g_2(D)$ and $g_3(D)$ generator polynomials. This results in a block of 382 coded bits {c(0),..., c(381)}.

$c(2k) = b(4k)$	for k = 0, …, 134;
$c(2k+1) = b(4k+1)$	for k = 0, …, 134;
$c(k) = b(k+270)$	for k = 270, …, 381.

6.5.2 Interleaving

The block of 382 bits {c(0), … , c(381)} is divided into two data blocks $c_1(k)$ and $c_2(k)$ that are interleaved separately.

$c_1(k) = c(k+270)$	for k = 0, …, 111;
$c_2(k) = c(k)$	for k = 0, …, 269.

Data block $c_1(k)$ is interleaved by first mapping the bits into 14 × 8 matrix by rows, then permuting the columns and finally reading out blocks of data by columns.

$c_1'(i,j) = c_1(k)$,	where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, k = 0, 111;
$e_1'(k) = c_1'(i,j)$,	where $k = i + 14 \times j$, i = 0, …, 13 and j = 0, …, 7.

The first 264 bits {$c_2(0)$, $c_2(1)$, …, $c_2(263)$} of data block $c_2(k)$ are interleaved by first mapping into 33 × 8 matrix by rows, then permuting the columns and finally reading out blocks of data by columns.
The remaining 6 bits {$c_2(264)$, $_2(265)$, …, $c_2(269)$} are appended to the output of the interleaver.

$c_2'(i,j) = c_2(k)$,	where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, k = 0, 263;
$e_2'(k) = c_2'(i,j)$,	where $k = i + 33 \times j$, i = 0, …, 32 and j = 0, …, 7;
$e_2'(k) = c_2(k)$,	where k = 264, 265, …, 269.

A 494-bit block {e′(0), …, e′(493)} is formed from the two blocks of interleaved bits $e_1'(k)$ and $e_2'(k)$ and a block of repeated $e_1'(k)$ bits as following:

$e'(k) = e_1'(k)$	for k = 0, …111;
$e'(k+112) = e_2'(k)$	for k = 0, …269;
$e'(k+382) = e_1'(k)$	for k = 0, …111.

6.5.3 Scrambling and multiplexing

The block {e'(0), ..., e'(493)} is scrambled as described in clause 4.9 to produce the output block {x(0), ..., x(493)}.
The scrambled bits x(k) are mapped onto the multiplexed bits {m(0), ..., m(493)} as shown in table 6.1.

Table 6.1: Multiplexed data mapping

x(k)	m(k)
x(112) to x(247)	m(0) to m(135)
x(0) to x(111)	m(136) to m(247)
x(382) to x(493)	m(248) to m(359)
x(248) to x(381)	m(360) to m(493)

The multiplexed bits m(k) become the encoded bits e(k):

$e(k) = m(k)$, for $k = 0, ..., 493$.

The resultant block of 494 encoded bits is mapped to the RACH as described in GMR-1 05.002 [2].

6.6 Cell broadcast channel (CBCH)

The message delivered to the encoder has a fixed size of 184 information bits.

6.6.1 Channel coding

The 184 information bits are extended with 8 zero bits to produce a block of 192 bits {d(0), ..., d(191)}. The coding of the 192 bits {d(0), ..., d(191)} is done as specified for the PCH.

6.6.2 Interleaving

The interleaving is done as specified for the PCH.

6.7 Standalone dedicated control channel (SDCCH)

The message delivered to the encoder has a fixed size of 84 information bits.

6.7.1 Channel coding

A 16-bit CRC is applied to the 84 message bits {d(0),..., d(83)} as specified in clause 4.3. The resultant block of 100 CRC-protected bits {u(0), ..., u(99)} is then encoded via the Rate 1/4 convolutional code as specified in clause 4.4.1.2. Convolutional encoding produces a block of 416 coded bits {c(0),..., c(415)}.

6.7.2 Interleaving

The block of 416 bits {c(0),..., c(415)} is interleaved by first mapping the bits into a 52 × 8 matrix by rows and then permuting the columns and finally reading out blocks of data by columns into two blocks of 208 bits each. These two blocks are mapped to two consecutive bursts at the physical layer in the SDCCH.

$c'(i,j) = c(k)$, where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, 415$;

$e'(B,k) = c'(i,j)$, where $k = (i + 52 \times j) \bmod 208$ and $B = \mathrm{INT}((i+52 \times j)/208)$;

$i = 0, ..., 51$ and $j = 0, ..., 7$.

6.7.3 Scrambling, multiplexing and encryption

The block {e'(B,0), ..., e'(B,207)} is scrambled, as described in clause 4.9, to produce the output block {x(B,0), ..., x(B,207)}. The scrambled bits x(B,k) become multiplexed bits:

m(B,k) = x(B,k), for k = 0, ..., 207.

Each block of 208 multiplexed bits is encrypted, as described in clause 8, to produce the blocks of encrypted bits y(B,k). The encrypted bits then become encoded bits e(B,k):

e(B,k) = y(B,k), for k = 0, …, 207.

Blocks e(B,k) are mapped to a SDCCH burst as described in GMR-1 05.002 [2]. Block 0 (B = 0) shall always be map into a burst in a frame numbered xxx0 and Block 1 (B = 1) shall always be mapped into a burst in the next frame numbered xxx1.

6.8 Slow associated control channel (SACCH)

The message delivered to the encoder has a fixed size of 76 bits {d(0), ..., d(75)}. When no message is present, the data link layer shall substitute 76 bits of fill.

6.8.1 Channel coding

A 16-bit CRC is applied to the 76 message bits {d(0), ..., d(75)} as specified in clause 4.3. The resultant block of 92 CRC-protected bits {u(0), ..., u(91)} is then encoded via the Rate 1/2 convolutional code as specified in clause 4.4.1.1. Convolutional encoding produces a block of 192 coded bits {c(0), ..., c(191)}.

6.8.2 Interleaving

The block of 192 bits {c(0), ..., c(191)} is augmented by 8 pad bits and the resultant 200 bits are given by:

$c'(k) = c(k),$ for $k = 0, \ldots, 191;$

$c'(k) = 0,$ for $k = 192, \ldots, 199.$

The block of 200 bits {c'(0),..., c'(199)} are interleaved by first mapping the bits into a 10 × 20 matrix by rows and then permuting the columns and finally reading out blocks of data by columns. Each of the 10-bit blocks is mapped to 20 consecutive bursts by combining with either FACCH or traffic channel (TCH) data as described in clause 7.1. No scrambling is performed on the SACCH bits.

$c''(i,j) = c'(k),$ where $j = (9 \times k) \bmod 20$ and $i = \mathrm{INT}(k/20);$

$e'(B,k) = c''(i,j),$ where $k = i$ and $B = j$, $i = 0, \ldots, 9$ and $j = 0, \ldots, 19$.

Block B = j, j = 0, …, 19 shall always be mapped into a burst in a frame with a frame number (FN % 20) = j. This mapping rule is applicable to both GS and UT.

6.9 Fast associated control channel-3 (FACCH3)

The message delivered to the encoder has a fixed size of 76 information bits.

6.9.1 Channel coding

A 16-bit CRC is applied to the 76 message bits {d(0), ..., d(75)} as specified in clause 4.3. The resultant block of 92 CRC-protected bits {u(0), ..., u(91)} is then encoded via the Rate 1/4 convolutional code as specified in clause 4.4.1.1. Convolutional encoding produces a block of 384 coded bits {c(0), ..., c(383)}.

6.9.2 Interleaving

The 384 coded bits {c(0), …, c(383)} are divided into four 96-bit blocks, such that each block contains the coded bits generated by one of four generator polynomials used in convolutional encoding.

$c'(B,j) = c(k),$ where $k = 0, \ldots, 383$, $B = k \bmod 4$ and $j = \mathrm{INT}(k/4)$.

Each block {c′(B,0), …, c′(B,95)}, where B = 0, 1, 2, or 3, is interleaved by first mapping the bits into 12 × 8 matrix by rows, permuting the columns and finally reading out blocks of data by columns.

For $B = 0, 1, 2$ and 3

$c''(B, i,j) = c'(B, k),$ where $j = (5 \times k) \bmod 8$, $i = \mathrm{INT}(k/8)$, $k = 0, \ldots, 95;$

$e'(B, k) = c''(B, i,j),$ where $k = i + 12 \times j$, $i = 0, \ldots, 11$ and $j = 0, \ldots, 7$.

6.9.3 Scrambling, multiplexing and encryption

The block {e'(B,0), ..., e'(B,95)} is scrambled, as described in clause 4.9, to produce the output block {x(B,0), ..., x(B,95)}. The scramble bits x(B,k) become multiplexed bits m(B,k):

$m(B,k) = x(B,k),$ for $k = 0, \ldots, 95$.

The 96 multiplexed bits m(B,k) are encrypted, as described in clause 4.9, to produce a block of 96 encrypted bits y(B,k) that is then multiplexed with status field bits, as described in clause 7.3.2.2. The resultant block of encoded bits {e(B, 0), …, e(B,103)} is mapped to the NT3 burst for FACCH as described in GMR-1 05.002 [2].

6.10 Fast associated control channel-6 (FACCH6)

The message delivered to the encoder has a fixed size of 188 information bits.

6.10.1 Channel coding

A 16-bit CRC is applied to the 188 message bits {d(0),..., d(187)} as specified in clause 4.3. The resultant block of 204 CRC-protected bits {u(0), ..., u(203)} is then encoded via the Rate 1/2 convolutional code as specified in clause 4.4.1.1. Convolutional encoding produces a block of 416 coded bits {c(0), ..., c(415)}.

6.10.2 Interleaving

The block of 416 bits {c(0),..., c(415)} is interleaved by first mapping the bits into a 52 × 8 matrix by rows and then permuting the columns and finally reading out blocks of data by columns.

$c'(i,j) = c(k),$ where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, 415$;

$e'(k) = c'(i,j),$ where $k = i + 52 \times j$, $i = 0, \ldots, 51$ and $j = 0, \ldots, 7$.

6.10.3 Scrambling, multiplexing and encryption

The block of 416 bits are augmented by 4 pad bits to produce the following 420 bits:

e"(0), ..., e"(1) = 0;

e"(k+2) = e'(k), for k = 0, ... 415 and

e"(418), ..., e"(419) = 0.

The block {e"(0), ..., e"(419)} is scrambled as described in clause 4.9 to produce the output block {x(0), ..., x(419)}. The scrambled bits x(k) are multiplexed with the SACCH, as described in clause 7.1, to produce a block of multiplexed bits m(k). The 430 multiplexed bits m(k) are encrypted, as described in clause 8, to produce a block of 430 encrypted bits y(k) that is then multiplexed with the status field bits, as described in clause 7.3.1. The resultant block of encoded bits {e(0), …, e(433)} is mapped to the NT6 burst as described in GMR-1 05.002 [2].

6.11 Fast associated control channel-9 (FACCH9)

The message delivered to the encoder has a fixed size of 300 information bits {d(0), ..., d(299)}.

6.11.1 Channel coding

A 16-bit CRC is applied to the 300 message bits as specified in clause 4.3. The resultant block of 316 CRC-protected bits {u(0), ..., u(315)} is then encoded via the Rate 1/2 convolutional code as specified in clause 4.4.1.1. Convolutional encoding produces a block of 640 coded bits {c(0),..., c(639)}.

6.11.2 Interleaving

The block of 640 bits {c(0),..., c(639)} is interleaved by first mapping the bits into a 80×8 matrix by rows and then permuting the columns and finally reading out blocks of data by columns.

$c'(i,j) = c(k)$, where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, 639$;

$e'(k) = c'(i,j)$, where $k = i + 80 \times j$, $i = 0, ..., 79$ and $j = 0, ..., 7$.

6.11.3 Scrambling, multiplexing and encryption

The block of 640 bits is augmented by 8 pad bits and the resultant 648 bits are given by:

$e''(0), ..., e''(3) = 0$;

$e''(k+4) = e'(k)$, for $k = 0, ..., 639$ and

$e''(644), ..., e''(647) = 0$.

The block {e"(0), ..., e"(647)} is scrambled as described in clause 4.9 to produce the output block {x(0), ..., x(647)}. The scrambled bits x(k) are multiplexed with the SACCH, as described in clause 7.1, to produce a block of multiplexed bits m(k). The 658 multiplexed bits m(k) are encrypted, as described in clause 8, to produce a block of 658 encrypted bits y(k) that is then multiplexed with the status field bits, as described in clause 7.3.1. The resultant block of encoded bits {e(0), …, e(661)} is mapped to the NT9 burst as described in GMR-1 05.002 [2].

6.12 Terminal-to-terminal associated control channel (TACCH)

The message delivered to the encoder has a fixed size of 108 information bits {d(0), ..., d(107)}.

6.12.1 TACCH channel coding

A 16-bit CRC is applied to the 108 message bits {d(0), ..., d(107)} as specified in clause 4.3. The resultant block of 124 CRC-protected bits {u(0), ..., u(123)} is then encoded via the Rate 1/2 convolutional code as specified in clause 4.4.1.1. Convolutional encoding produces a block of 256 coded bits {c(0), ..., c(255)}.

6.12.2 Interleaving

The block of 256 bits {c(0),..., c(255)} are interleaved by first mapping the bits into a 32×8 matrix by rows and then permuting the columns and finally reading out blocks of data by columns. Each 16×8 bit block is mapped to two consecutive bursts at the physical layer in the TACCH.

$c'(i,j) = c(k)$, where $j = (5 \times k) \bmod 8$ and $i = \mathrm{INT}(k/8)$, $k = 0, ..., 255$;

$e'(B, k) = 0$, where $k = 0, 1$ and $B = 0, 1$;

$e'(B,k+2) = c'(i,j)$, where $k = (i + 32 \times j) \bmod 128$ and $B = \mathrm{INT}((i + 32 \times j)/128)$;

$k = 0, …, 127$ as $i = 0, ..., 31$ and $j = 0, ..., 7$;

$e'(B,k) = 0$, where $k = 130, 131$ and $B = 0, 1$.

6.12.3 Scrambling and multiplexing

The block {e'(B,0), ..., e'(B,131)} is scrambled as described in clause 4.9 to produce the output block {x(B,0), ..., x(B,131)}. The scrambled bits x(B,k) become the encoded bits e(B,k):

e(B,k) = x(B,k), for k = 0, …, 131.

Each 132-bit block of encoded bits is mapped to a DC2 burst at the physical layer in the terminal-to-terminal channel (TTCH). Block 0 shall always be mapped into a burst in a frame numbered xx0x and Block 1 shall always be mapped into a burst in the next frame numbered xx1x. For example, if Block 0 is mapped to a frame numbered xx00, then Block 1 is mapped to a frame numbered xx10; if Block 0 is mapped to a frame numbered xx01, then Block 1 is mapped to a frame numbered xx11.

6.12.4 Physical (PHY) header for TACCH

On the receive side, the physical layer entity shall analyse the first octet of the 108-bit message, d(i), i = 0, ..., 7. This field is the PHY header. It contains a 4-bit MES address or temporary terminal identification (TTID), d(i), i = 0, ..., 3, two reserved bits d(i), i = 4, 5 and a 2-bit message type field, d(i), i = 6, 7. The message type field shall indicate to the physical layer how to parse the TACCH message. The values and interpretation of the message type field are shown in table 6.2.

Table 6.2: PHY header interpretation

d(7)	d(6)	Interpretation
0	0	Header followed by a null message - PHY to disregard
0	1	Header followed by a 3-octet DL message and then another PHY header
1	0	Header followed by one DL message
1	1	Reserved

If the value of the message type field is "01" or "10" the PHY shall analyse the MES address field. If the value of the 4-bit TTID is the one assigned to the MES then the PHY passes the body of the message to the DL. If the TTID is not the one assigned to the MES, the PHY discards the message.

If the value of the message type field is "01" then the PHY shall analyse the next PHY header in the TACCH message offset by 4 octets, d(i), i = 32, ..., 39. In this header, the 4 bits, d(i), i = 32, ..., 35 contain another MES address and the 2 bits, d(i), i = 38, 39, contain another message type field. The message type field can contain the values "01" and "10" only. These bits shall be interpreted as before.

If the value of the message type field is "01", then the PHY shall analyse the next PHY header in the TACCH message offset by 4 octets, d(i), i = 64, ..., 71. In this header, the 4 bits, d(i), i = 64, ..., 67, contain another MES address and the 2 bits, d(i), i = 70, 71, contain another message type field. The message type field can contain the value "10" only. These bits shall be interpreted as before.

6.13 GPS broadcast channel (GBCH)

The message delivered to the encoder has a fixed size of 108 information bits {d(0), ..., d(107)}.

6.13.1 Channel coding

The coding of the 108 information bits is done as specified for the TACCH.

6.13.2 Interleaving

The interleaving is done as specified for the TACCH.

6.13.3 Scrambling and multiplexing

Scrambling is done as specified for the TACCH.

Each 132-bit block of encoded bits is mapped to a DC2 burst at the physical layer. Block 0 shall always be mapped into a burst in a frame numbered xxx0 and Block 1 shall always be mapped into a burst in the next frame numbered xxx1.

7 Logical channel multiplexing

7.1 SACCH multiplexing

For N = 6, or 9, the TCHN and the FACCHN channels are combined with the SACCH and multiplexed as follows.

$m(k) = x(k)$, $k = 0, ..., 51$;

$m(k + 52) = e'(B,k)$, $k = 0, ..., 9$, $B = FN \bmod 20$;

$m(k + 10) = x(k)$, $k = 52, ...M$;

where M = 419, or 647, when N = 6 or 9. The $e'(B,k)$, $k = 0, ..., 9$, $B = FN \bmod 20$, are the SACCH and are defined in clause 6.8.

For N = 3, the TCH3 and the FACCH3 are not combined with a SACCH, so that:

$m(k) = x(k)$, $k = 0, ..., 207$ for TCH3 and

$m(B,j) = x(B,j)$, $j = 0, ..., 95$ for FACCH3.

The output of the first intraburst multiplexing unit, bits m, is then encrypted as described in clause 8.

7.2 Status field

A status field consists of one or two field types. One type is the power control field denoted as s_p and the second type is the comfort noise field denoted as s_n.

7.2.1 Power control field

A status field block has a fixed size of 12 bits.

Power control field coding

A block of 12 status field bits {u(0), u(1), …, u(11)} is encoded via (24, 12) Golay code as specified in clause 4.6. Golay encoding produces a block of 24 coded bits {c(0), c(1), …, c(23)}.

Interleaving

The block of 24 coded bits {c(0), c(1), …, c(23)} is interleaved to produce a 24-bit block {c'(0), c'(1), …, c'(23)}, where c'(j) = c(k), j = 0, …, 23 and index k corresponding to each j shown in table 7.1.

Table 7.1: Index pairs for interleaving of coded power control field bits

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
k	6	0	18	12	13	19	1	7	20	14	8	2	15	21	3	9	4	22	10	16	5	11	17	23

The 12-bit mask m = {1,0,0,0,0,1,0,0,0,0,1,0} is then applied to the block of the interleaved bits as following:

$c''(j) = c'(k)$ for $k = 0, …, 11$;

$c''(j) = c'(k+12) + m(k)$ for $k = 0, …, 11$, where + denotes modulo-2 addition.

Finally, the 24-bit block {c"(0), c"(1), …, c"(23)} is divided into six 4-bit blocks as following:

$s_p(B',k) = c''(j),$ where j = 0, …, 23, B' = INT(j/4) and k = j mod 4.

These 4-bit blocks are transmitted over the six consecutive FACCH3 and/or DKAB bursts. The synchronization issues are addressed in GMR-1 05.010 [5].

7.2.2 Comfort noise field

A comfort noise block delivered by the user unit has a fixed size of 80 bits.

Comfort noise field coding

The coding of the 80 comfort noise field bits {d(0), …, d(79)} is done as specified for the TCH3 in clause 5.1.1.

Interleaving

The block of 104 interleaved bits {e'(0), …, e'(103)} is obtained as specified in clause 5.1.2.

The block of 104 interleaved comfort noise bits e'(k) is then divided into the 26 groups of 4-bits each as follows:

$s_n(B'', j) = e'(k)$ where B" = INT(k/4), j = k mod 4 and k = 0, …, 103.

These 4-bit groups are transmitted over the 26 consecutive dual keep-alive burst (DKAB) and FACCH3 bursts.
The synchronization issues are addressed in GMR-1 05.008 [4].

7.3 Status field with NTN bursts

7.3.1 Status field with NT6 and NT9 bursts

A block of 4 status field bits,{ $s_p(B',0)$, …, $s_p(B',3)$}, is multiplexed with the output of the encryption unit, bits y(0) to y(n), as follows:

$e(j) = y(j),$ where j = 0, …, 51;

$e(j+52) = s_p(B',j),$ where j = 0, …, 3, B'= FN mod 6;

$e(j+4) = y(j),$ where j = 52, …, n;

where n = 429, for N = 6 and n = 657, for N = 9. This results in the block of (n+5) encoded bits e(k).

7.3.2 Status field with NT3 bursts

7.3.2.1 Status field with NT3 bursts for encoded speech

A block of 4 status field bits,{ $s_p(B',0)$, …, $s_p(B',3)$}, is multiplexed together with the output of the encryption unit, bits y(0) to y(207), as follows:

$e(j) = y(j),$ where j = 0, …, 51;

$e(j+52) = s_p(B',j),$ where j = 0, …, 3, B' = FN mod 6;

$e(j+4) = y(j),$ where j = 52, …, 207.

This results in the block of 212 encoded bits e(k).

7.3.2.2 Status field with NT3 bursts for FACCH

A block of 8 status field bits, {s_n(B",0), …, s_n(B",3), s_p(B',0), …, s_p(B',3)}, is multiplexed with the output of the encryption unit, bits y(B,0) to y(B,95), as follows:

$e(B,j) = y(B,j)$, where j = 0, ..., 21;

$e(B,j+22) = s_n(B'',j)$, where j = 0, ..., 3;

$e(B,j+26) = s_p(B',j)$, where j = 0, ..., 3;

$e(B,j+8) = y(B,j)$, where j = 22, …, 95;

where B = FN mod 4 and B'= FN mod 6. Reference to B" selection is given in clause 5.2.2.

This results in the block of 104 encoded bits e(B,k).

7.3.3 Status field with keep-alive bursts (KAB)

A block of 8 status field bits, {s_n(B",0), …, s_n(B",3), s_p(B',0), …, s_p(B',3)}, is multiplexed into dual KAB bursts as follows:

$e(j) = s_p(B',j)$, where j = 0, 1;

$e(j+2) = s_n(B'',j)$, where j = 0, 1;

$e(k) = s_p(B',j)$, where j = 2, 3 and k = j + 2;

$e(k + 2) = s_n(B'',j)$, where j = 2, 3 and k = j + 2;

where B′ = FN mod6. Reference to B″ selection is given in clause 7.2.2.

The 8 encoded bits {e(0), …, e(7)} are mapped into the dual KAB bursts as described in GMR-1 05.002 [2].

8 Encryption

Encryption is performed on certain GMR-1 channels. Table 8.1 summarizes channels that are encrypted.

Table 8.1: Encrypted GMR-1 channels

Channel type	Channel name
TCH Channel	TCH3 TCH6 TCH9
DCCH Channel	FACCH3 FACCH6 FACCH9 SACCH SDCCH

The block diagram illustrating an encryption process is shown in figure 8.1. The block of multiplexed bits (m) is encrypted as described in GMR-1 03.020 [6] to produce an output block of encrypted bits (y).

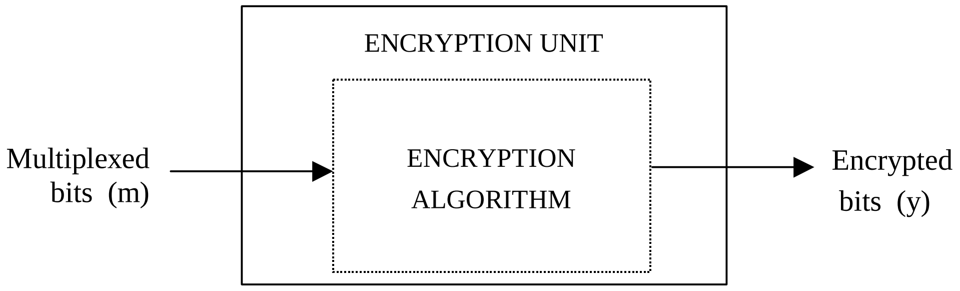


Figure 8.1: Encryption process block diagram

History

Document history		
V1.1.1	March 2001	Publication
V1.2.1	April 2002	Publication