Title: Lucent Technologies' TDMA Half Rate Speech Codec

Source:

Michael D. Turner
Lucent Technologies
Wireless Technology Laboratory
Room 2A-203
67, Whippany Road
Whippany, NJ 07981
(v) 973 386 3579
(f) 973 386 2651
mdturner@lucent.com

Nageen Himayat
Lucent Technologies
Wireless Technology Laboratory
Room 14B-317
67, Whippany Road
Whippany, NJ 07981
(v) 973 884 3954
(f) 973 386 2651
nhimayat@lucent.com

James P. Seymour
Lucent Technologies
Wireless Technology Laboratory
Room 2A-246
67, Whippany Road
Whippany, NJ 07981
(v) 973 386 8096
(f) 973 386 2651
jpseymour@lucent.com

Andrea M. Tonello
Lucent Technologies
Wireless Technology Laboratory
Room 1A-210
67, Whippany Road
Whippany, NJ 07981
(v) 973 386 5192
(f) 973 386 2651
tonello@lucent.com

Abstract:

This contribution describes Lucent Technologies' TDMA half-rate speech codec proposal. Both the speech encoding/decoding and the channel/encoding decoding procedures are described. The text provided emphasizes the differences from TIA/EIA 136-410 speech coder, [1]. This document may also provide normative text for TDMA half-rate speech codec, in conjunction with TIA/EIA 136-410 and with TIA/EIA 136-131. Modifications to TIA/EIA 136-131 to incorporate TDMA half-rate codec are suggested in [4].

Recommendation:

FYI

Notice:

Copyright Statement:

Grant of License:

IPR Declaration:

1. Introduction

This document presents the TDMA Half-Rate codec proposal by emphasizing the differences from TIA/EIA 136-410 speech coder, [1]. The channel encoding/decoding procedures required for the TDMA half-rate codec are also specified.

2. General description of the Speech Codec

2.1. Principles of the ACELP Encoder

The codec frame and subframe structure is the same as that in the TIA/EIA 136-410 speech coder, each frame of duration 20 ms and 4 subframes per frame however with 0.75 ms lookahead. The LP analysis is performed once per frame. The set of LP parameters are quantized in line spectrum pair (LSP) domain using a multi-stage vector quantization. An open-loop pitch lag is estimated twice per frame as in TIA/EIA 136-410.

Then the following operations are repeated for each subframe:

- The target signal x(n) is computed in the same way as in TIA/EIA 136-410.
- The impulse response, h(n) is computed.
- Closed-loop pitch analysis is performed with a 1/3-resolution fractional pitch search in the same way as in TIA/EIA 136-410. The full lag is encoded in the first and third subframes and relatively encoded in the second and fourth subframes.
- The target signal x(n) is updated by removing the adaptive codebook contribution. An algebraic codebook (ACELP) is used for the innovative excitation.
- The gains of the adaptive and fixed codebook are vector quantized.
- Finally, the filter memories updated.

The bit allocation of the codec is shown in Table 2.1-1. For each 20 ms speech frame, 124 bits are produced, corresponding to a fixed coding rate of 6.2 kbps.

Table 2.1-1: Bit allocation of the 6.2 kbps coding algorithm.

Detailed bit allocation table t.b.a.

2.2. Principles of the ACELP Decoder

The signal flow of the decoder is Same as that in TIA/EIA 136-410.

2.2.1. Audio Interface

Same as that in TIA/EIA 136-410.

3. Speech Encoding

In this section the blocks of the speech encoder are described.

3.1. Pre-processing

Pre-processing and high-pass filtering are identical to that in TIA/EIA 136-410.

The high-pass filtered output is sent to the noise-suppression and linear prediction analysis functions.

3.2. Linear Prediction Analysis and Quantization

Linear prediction (LP) analysis is performed on the input data using an autocorrelation approach with an asymmetric window.

The autocorrelation coefficients produced from the original input data are sent to the noise-suppression voice-activity detector. The output data from the noise-suppression function is reanalyzed by the autocorrelation function then processed as described below.

The LP parameters are computed by Levinson algorithm and transformed to LSP domain for quantization and interpolation purposes.

3.2.1. Levinson-Durbin Algorithm

Same as that in TIA/EIA 136-410.

3.2.2. LP to LSP Conversion

Same as that in TIA/EIA 136-410.

3.2.3. LSP to LP Conversion

Same as that in TIA/EIA 136-410.

3.2.4. Quantization of LSPs

The LSP vector is quantized with a 4th order MA predictive multi-stage vector quantizer (MSVQ).

3.2.5. Interpolation of the LSPs

The LP parameters for each subframe are computed using a linear interpolation of the parameters with the previous frame. The interpolation coefficients accommodate the 0.75 ms lookahead.

3.3. Open-loop Pitch Analysis

Same as that in TIA/EIA 136-410.

3.4. Impulse Response Computation

Same as that in TIA/EIA 136-410.

3.5. Target Signal Computation

Same as that in TIA/EIA 136-410.

3.6. Adaptive Codebook Search

Same as that in TIA/EIA 136-410.

3.7. Algebraic Codebook Structure

The codebook structure is based on interleaved single-pulse permutation design similar but with fewer pulses as TIA/EIA 136-410.

3.8. Quantization of the Gains

The fixed codebook gain quantization is performed using a 4th order MA prediction with fixed coefficients. However adaptive codebook gain is also jointly quantized with fixed codebook gain.

3.9. Memory Update

Same as that in TIA/EIA 136-410.

4. Speech Decoding

In this section the blocks of the speech decoder are described.

4.1. Decoding and Speech Synthesis

The decoding is performed in the following order:

Decoding of the LP parameters: Quantized LSP vector recovered and the LP parameter interpolation are performed for each subframe.

Then for each subframe:

- **Decoding of the adaptive codebook vector:** The integer and fractional pitch lag components are recovered based on received index and the adaptive codebook vector is generated as in TIA/EIA 136-410.

- **Decoding of the innovative codebook:** The corresponding pulse structure is generated from the received indices. The pitch sharpening procedure is applied if pitch lag is less than subframe size 40 as in TIA/EIA 136-410.

- **Decoding of the adaptive and fixed codebook gains:** The gains are recovered from the received index.

- **Computing the reconstructed speech:** Same as that in TIA/EIA 136-410.

4.2. Post-processing

4.2.1. Algebraic-codebook Post-processing

An additional post-processing filter is applied in order to reduce the perceptually adverse effects of the sparse excitation [3]. The filter alters the innovation signal to create a new innovation which has the energy spread over the subframe. The filter alters mainly the phase of the innovation through a “semi-random” impulse response. The filtering is performed by circular convolution, using one of three stored impulse responses each with a different amount of spreading. The filter selection is controlled by a voicing decision, based on the filtered received pitch gain. A strong increase in fixed-codebook gain is also detected to avoid spreading of onsets.

4.2.2. Adaptive Post-filtering

The adaptive post-filter structure is identical to that used in TIA/EIA 136-410. A special filter has been added to perceptually enhance the coded residual noise and is described in the Noise Suppression section below.

4.2.3. High-pass Filtering and Up-scaling

Same as that in TIA/EIA 136-410.

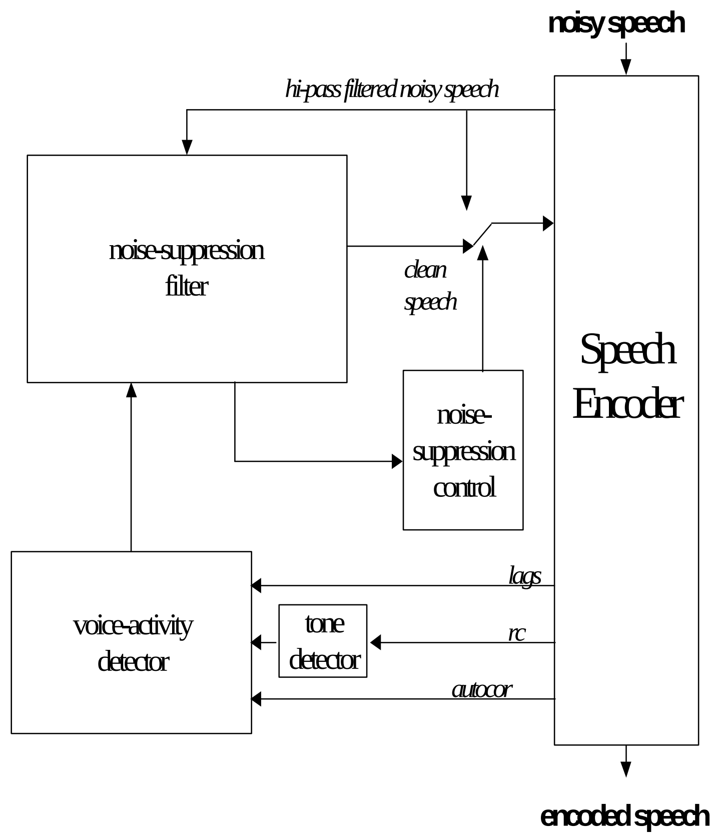
5. Noise Suppression

On calls, that are determined to contain a background noise level such that the voice quality at the receiving end is degraded, a noise suppression algorithm is applied on the uplink to reduce the noise level. This reduces the annoying effect of the noise on the listener and prevents the speech encoder from being degraded in its ability to determine parameters as accurately as for clean speech. The algorithm consists of two components. The first component is integrated with the speech encoder and performs the bulk of the noise suppression. The second component is integrated with the speech decoder and is used as a noise post-filter for further perceptual enhancement of the residual noise.

5.1. Noise Suppression in the Speech Encoder

The encoder noise suppressor receives input data immediately after the speech encoder pre-processor high-pass filtering. The voice-activity detector relies on parameters computed by the speech encoder functions. The output from noise suppression is sent to the LP analysis function.

Figure 5.1-1: Noise-suppression in the Speech Encoder.



The processing blocks of noise-suppression in the speech encoder are described below.

5.1.1. Tone Detector

The tone detector is the same as used for TIA/EIA 136-410, Annex D to detect information tones. The output indicates the presence of information tones and thereby prevents the VAD from updating the noise threshold.

5.1.2. Encoder Voice Activity Detector

The Voice Activity Detector (VAD) is based on the VAD from the TIA/EIA 136-410, Annex D. The VAD is used to identify noise only frames.

5.1.3. Noise Suppression Filter

The noise suppression filter uses the VAD input to collect characteristics of the noise-only portion of the incoming signal such as level, spectral shape, duration, etc. This information is used to model the noise background and to construct an inverse filter which applied to both noise-only and speech + noise regions to suppress the contribution of the noise. Details of the noise suppression filter are t.b.a.

5.1.4. Noise Suppression Control

Noise suppression should only be applied to frames for which there will be noticeable improvement in the speech quality. Therefore it is desirable to turn off the algorithm when either the noise level is too low for the listener to discern any improvement, or the noise level is so high such that the SNR makes the algorithm fail to improve the call quality. It is also desirable to turn off the noise suppression when music is present. A music detector is used to prevent the noise filter from corrupting music such as from a call being put “on hold”.

5.2. Noise Suppression in the Speech Decoder

5.2.1. Decoder Voice Activity Detector

Before calling the post processing function, the decoder calls the Decoder VAD function. This is similar to the encoder VAD except it is computationally simpler since it operates on frame energy rather than filtered energy.

5.2.2. Pitch Sharpening for Noise

For frames classified as noise by the decoder VAD, pitch sharpening in the decoder is bypassed.

5.2.3. Perceptual Enhancement of Coded Noise

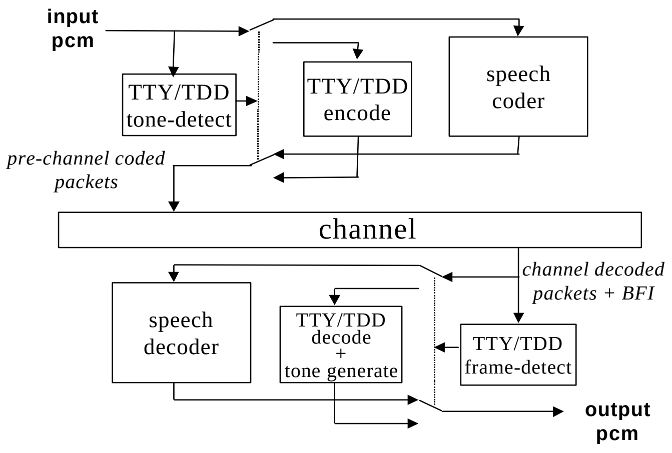
When the decoder VAD indicates a frame as noise-only, a special filter is applied to perceptually enhance the coded residual noise.

6. TTY/TDD Processing

Baudot tones are processed by detecting the characters being transmitted from the TTY/TDD device into the encoder and conveying the character information rather than the tones to the decoder. Because one Baudot character spans a minimum of 8 speech processing frames, the information for each character being transmitted is sent 8 times to the decoder. This redundancy allows the decoder to correctly regenerate the character despite frame errors (FERs) and random bit errors in the speech packet.

The TTY characters are encoded into the speech packet in a robust way to allow reliable detection in the decoder. When Baudot tones are not present, the vocoder processes signals through the speech encoder and decoder.

Figure 6-1: TTY/TDD encoder/decoder.



6.1. TTY/TDD Encoder

The TTY/TDD processing in the encoder has a detector, which is constantly checking the input PCM stream for the presence of Baudot tones. Each frame is classified as being non-TTY (NON_TTY), TTY character, or silence in-between TTY characters (TTY_SILENCE). If no tones are detected, the frame is labeled as NON_TTY and passed to the speech encoder. If Baudot tones are detected, the TTY/TDD encoder builds a TTY/TDD frame (shown in Table 6.1-1).

When Baudot tones are initially detected, the encoder builds a TTY_SILENCE frame constructed from a 5-bit TTY_SILENCE code and a 2-bit TTY_SILENCE header. This frame is transmitted while attempting to detect the incoming character.

When a character is detected, the encoder builds a TTY character frame constructed from the 5-bit character code along with a 2-bit sequence header. The header consists of a sequential counter and allows the decoder to better distinguish between consecutive transmissions of the same character.

Table 6.1-1: Bit allocation of the 6.2 kbps TTY/TDD frame.

Mode	Parameter	Total per frame
TTY/TDD	Reserved	2
	TTY/TDD header	2
	TTY/TDD character	5
	TTY/TDD header repeated	2
	TTY/TDD character repeated	5
	TTY/TDD header repeated	2
	TTY/TDD character repeated	5
	Zeros	32
	Unused	69
	Total	124

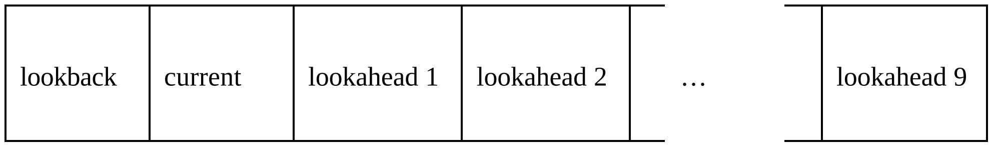
For each character the encoder detects, the same TTY/TDD frame is transmitted 8 consecutive times. When silence is detected in-between characters, the encoder sends the frame for TTY_SILENCE.

When speech returns it is passed to the speech encoder. No special message is sent to indicate that speech is being processed the decoder knows to stop regenerating Baudot tones when the TTY/TDD frame is not detected.

6.2. TTY/TDD Decoder

The TTY/TDD decoder detector monitors each incoming frame, looking for the unique TTY/TDD frame format. Each frame is classified as in the encoder along with one new type – a frame erasure (BFI). If a frame is classified as a TTY character, the character is identified by its header and character information. The decoder maintains a TTY/TDD history buffer that stores these classifications for 11 frames; there are 9 frames of lookahead, 1 current frame, and 1 frame of lookback (Figure 6.2-1). This buffer is initialized to be NON_TTY and is updated every frame. As long as the current frame is NON_TTY, it is passed directly to the speech decoder.

Figure 6.2-1: TTY/TDD receiver buffer structure.



When the TTY tones first arrive, the current frame will transition from NON_TTY to TTY_SILENCE to signal the decoder that TTY characters are coming. At this point, the decoder checks its lookahead buffer to make sure this is not an erroneous message. At this point, the decoder will expect either TTY characters or TTY_SILENCE messages. If TTY_SILENCE reaches the current frame, the decoder mutes its output. When the information for a character reaches the current frame for the first time, it checks its lookahead to make sure that the next 7 frames of lookahead also contain the same information since the encoder should have sent the same character information for 8 consecutive frames. If there is a discrepancy, or if there are frame erasures, a vote is taken to see which character was most likely to have been sent. If a character is determined to be present, its Baudot tones are regenerated and written to the vocoder output buffer.

False alarms are avoided by forcing the TTY decoder to vote on the current frame every time there is a transition from NON_TTY to anything else. This voting process is also used to protect the TTY processing from random bit errors that may have been injected by the wireless channel.

7. Channel Encoding

The channel encoding process is different for the base-to-mobile and the mobile-to-base transmissions, due to their asymmetric transmission capacities.

[Editors Note: Channel coding techniques for the half-rate codec are non-adaptive and are based on fixed coding/interleaving schemes and modulation.]

7.1. Speech Data Classes

The speech codec produces 124-output bits. For channel encoding, these data bits are divided into sub-classes according to the perceptual significance of each bit. Channel coding utilizes these sub-classes to provide a level of error protection for each class, which is commensurate with its perceptual significance. Table 7.1-1 defines the speech classes for the half-rate speech codec.

Table 7.1-1: Speech Data Classes

Speech Class	Number of Bits	Description
Class 1A	63	The perceptually critical Class 1 bits. These bits are protected by a cyclic redundancy check (CRC). If the CRC fails upon reception, the Class 1A frame, as well as the Class 1B bits, are discarded. The frame is then reconstructed based on past data.
Class 1B	11	These bits are perceptually important but are not included in the CRC computation. Therefore erroneous bits in this class do not cause a frame error. However, these bits are not used for decoding should a frame error occur.
Class 2	50	The least perceptually significant bits. Some of the Class 2 bits are perceptually more important as an error in one of these bits may affect the performance of several other Class 2 bits.

7.2. Ordering of the Speech Encoder Bit Stream

The 124 speech bits, described in Table 7.1-1, are provided to the channel encoder in descending order of perceptual importance. The speech bits at the input to the channel encoder are labeled as S(0)-S(123). Class 1A bits are labeled as S(0)-S(62), Class 1B bits are labeled as S(63)-S(73), and the remaining bits, S(74)-S(123), are Class 2 bits. Table 7.2-1 describes the bit order in which the channel encoder receives data from the speech encoder.

Table 7.2-1: Ordering of the speech encoder bit stream

Detailed speech bit order t.b.a

7.3. Channel Encoding for Base-to-Mobile Transmission

The channel encoding for TDMA half-rate, base-to-mobile transmission, is based on two users co-sharing commonly assigned time-slots, [4].

Figure 7.3-1: Channel encoding for base-to-mobile transmission

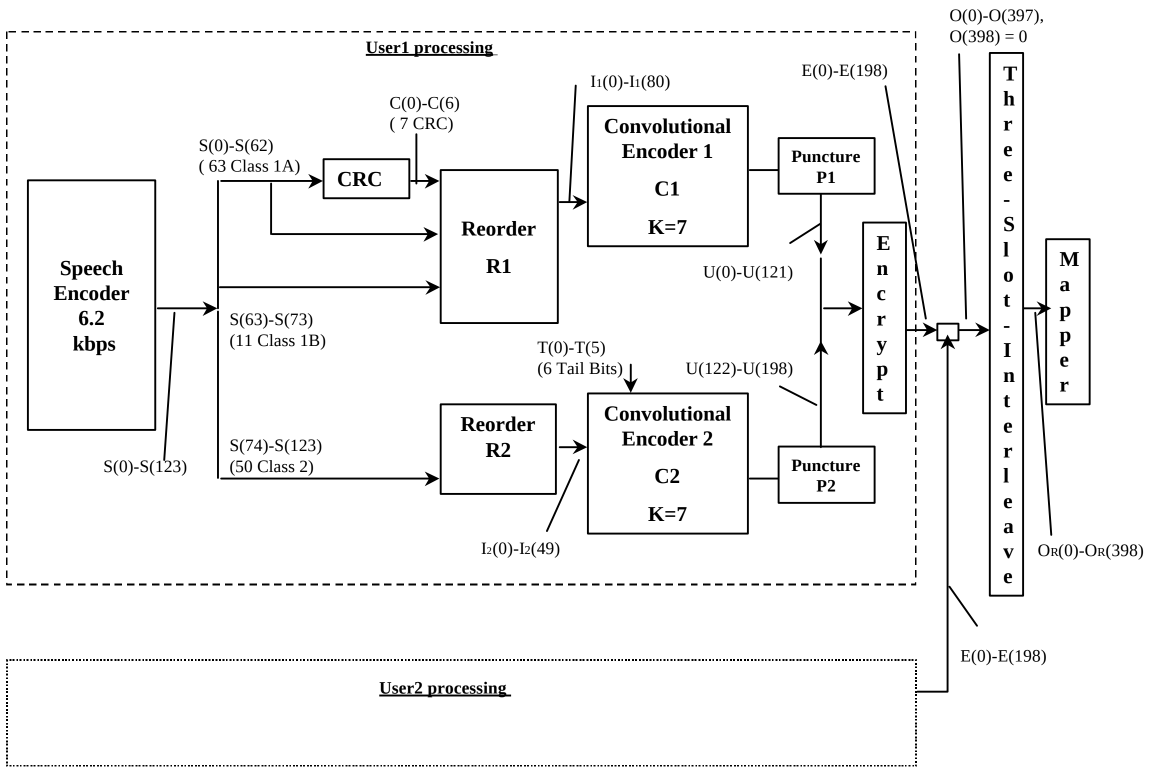


Figure 7.3-1 illustrates the channel encoding used for base-to-mobile transmission. In this case, two users shall be assigned common time-slots for transmission within each TDMA frame. Each user's data must be encoded separately but the channel encoding procedures applied must be identical for both users. If data is to be encrypted prior to transmission then the encryption mask must be applied before interleaving the data.

The following discussion applies to both users' channel encoding procedures. A 7 bit cyclic redundancy check (CRC) is computed over the 63 Class 1A bits to produce the CRC bits C(0)-C(6). The Class 1A, Class 1B and the CRC bits are combined and convolutionally encoded together as part of a single data frame. Prior to encoding the bits are reordered according to R1. The input to the convolutional encoder, C1, comprises 81 bits labeled as $I_1(0)$-$I_1(80)$. Note that a tail-biting code is used for C1 and no tail bits are transmitted. The encoder C1 is a rate ½, constraint length 7 (memory-order 6) code, which is punctured according to P1 to produce 122 output data bits, U(0)-U(121).

The 50 Class 2 bits form a separate frame and are encoded using the convolutional encoder C2, which is also a rate ½, constraint length 7 encoder. The bits are reordered, according to R2, prior to the encoding operation. The resulting reordered bits, $I_2(0)$-$I_2(49)$, are combined with 6 tail bits, T(0)-T(5), to force the encoder to end in a known state. The output of C2 is punctured according to P2 to provide 77 output bits, numbered U(122)-U(198), for a total of 199 encoded bits.

At this stage it is possible to apply a voice privacy mask to each user's encoded data. The encrypted (or clear) data stream of both users' is then combined in a single stream. Bits E(0)-E(198) of the first user are labeled O(0)-O(198). Bits E(0)-E(198) of the second user are labeled O(199)-O(397). Bit O(398) is set to zero to produce a total output of 399 bits.

The combined output stream, O(0)-O(398), is interleaved over three slots.

The TDMA half-rate solution for the base-to-mobile direction, works in conjunction with 8-PSK modulation and the 8-PSK slot format defined for the base-to-mobile link in TIA/EIA 136-131, [2]. Therefore the output of the interleaver, $O_R(0)$-$O_R(398)$, is mapped into complex symbols using an 8-PSK mapper. *[Editor's Note: This paragraph is informative and should not be included in the actual standard.]*

7.3.1. Cyclic Redundancy Check (CRC)

A 7-bit CRC is computed for the 63 Class 1A bits in the frame. The CRC parity bits are calculated according to the following CRC polynomial.

$$g(X) = 1 + X + X^2 + X^4 + X^5 + X^7 \qquad (7.3.1\text{-}1)$$

The input polynomial consists of Class 1A bits S(0)-S(62) and is defined as

$$a(X) = S(0)X^{62} + S(1)X^{61} + \ldots + S(61)X^1 + S(62)X^0 \qquad (7.3.1\text{-}2)$$

The CRC encoding and decoding procedure is based on TIA-136-410, [1].

7.3.2. Pre-Encoder Bit Ordering

The bit order presented to the convolutional encoders, C1 and C2, is described by the R1 and the R2 ordering arrays respectively. The bits I_1 and I_2 are produced according to the following equations.

$$I_1(j) = R1(j) \quad j = 0,\ldots,80 \qquad (7.3.2\text{-}1)$$

$$I_2(j) = R2(j) \quad j = 0,\ldots,49 \tag{7.3.2-2}$$

The following tables describe the R1 and the R2 arrays. The index into the arrays increases row-wise. Index "0" is the first element of the first row, and the final index is the last element of the last row.

Table 7.3.2-1: Bit Order for C1 described by R1 (base-to-mobile)

C(0), C(1), C(2), S(0), S(2), S(4), S(6), S(8), S(10), S(12), S(14), S(16), S(18), S(20), S(22), S(24), S(26), S(28), S(30), S(32), S(36), S(38), S(40), S(42), S(44), S(46), S(48), S(50), S(52), S(54), S(56), S(58), S(60), S(62), S(64), S(66), S(68), S(70), S(72), S(73),S(71), S(69), S(67), S(65), S(63), S(61), S(59), S(57), S(55), S(53), S(51), S(49), S(47), S(45), S(43), S(41), S(39), S(37), S(35), S(33), S(31), S(29), S(27), S(25), S(23), S(21), S(19), S(17), S(15), S(13), S(11), S(9), S(7), S(5), S(3), S(1), C(3), C(4), C(5), C(6)

Table 7.3.2-2: Bit Order for C2 described by R2 (base-to-mobile)

S(74), S(76), S(78), S(80), S(82), S(84), S(86), S(88), S(90), S(92), S(94), S(96), S(98), S(100), S(102), S(104), S(106), S(108), S(110), S(112), S(114), S(116), S(118), S(120), S(122),S(123), S(121), S(119), S(117), S(115), S(113), S(111), S(109), S(107), S(105), S(103), S(101), S(99), S(97), S(95), S(93), S(91), S(89), S(87), S(85), S(83), S(81), S(79), S(77), S(75)

The tail bits T(0)-T(5) are set to zero and appended at the end of the stream I_2, after the bit $I_2(49)$. T(0) is appended first and T(5) is appended last.

7.3.3. Convolutional Encoding

The encoder C1 encodes 81 input bits, labeled $I_1(0)$-$I_1(80)$. The convolutional encoder is a rate ½, constraint length (K) 7 code. There are 64 states in this code and six memory elements. Table 7.3.3-1 shows all the states and their outputs to a given input. The two generator polynomials for this rate ½ code may be represented in octal form as

$$\{g_0(D), \quad g_1(D)\} = \{0133, \quad 0171\} \tag{7.3.3-1}$$

This corresponds to the polynomial representation:

$$g_0(D) = 1 + D^2 + D^3 + D^5 + D^6 \tag{7.3.3-2}$$

$$g_1(D) = 1 + D + D^2 + D^3 + D^6 \tag{7.3.3-3}$$

The output from the convolutional coder alternates between these two polynomials, starting with $g_0(D)$ providing the first output bit. The puncture pattern P1 describes the exact output of the encoder. After puncturing, C1 produces 122 output bits. The tail-biting convolutional encoding process may be viewed in the following manner. Initially the encoder's memory elements are cleared and the last six input bits,

$I_1(75)$, $I_1(77)$,…,$I_1(80)$, are fed in to the encoder sequentially, starting with $I_1(75)$. The input bits $I_1(0)$-$I_1(80)$ are then read in starting at $I_1[0]$ and concluding with $I_1(80)$. For each input bit, $I_1(i)$, two output bits are produced. The puncture pattern P1 determines which output bits are included in the output stream U. Initializing the state of the encoder with input bits $I_1(75)$-$I_1(80)$ ensures that when the same data bits are presented as input to the encoder, after $I_1(0)$-$I_1(74)$ have been presented, the final memory state of the encoder will be the same as the starting state. The value of this initial and final state will depend on the values of bits $I_1(75)$-$I_1(80)$ and hence will be unknown to the decoder.

The convolutional encoder C2 is identical to C1 (rate ½, K = 7) and is also described by the input-output relationship described in Table 7.3.3-1. The puncture pattern P2 applied to the output of this encoder is different and 77 output bits are produced after puncturing. The encoder C2 produces a data frame that is terminated with the help of tail-bits. Initially the encoder's memory elements are cleared, and the encoder starts in state zero. The input bits $I_2(0)$-$I_2(49)$ are read in, starting at $I_2(0)$ and concluding with bit $I_2(49)$. For each input bit, $I_2(i)$, two output bits are produced. The puncture pattern P2 determines, which output bits are included in the output stream U. The final 6 input bits presented to C2 are the 6 tail bits T(0)-T(5), which are set to zero. This ensures that after bits {$I_2(0)$-$I_2(49)$, T(0)-T(5)} are transmitted, the encoder will end in the all-zero state.

Table 7.3.3-1: Input - Output Relationship of the Convolutional Encoder

	INPUT = 0		**INPUT = 1**			**INPUT = 0**		**INPUT = 1**	
state	**g0**	**g1**	**g0**	**g1**	**state**	**g0**	**g1**	**g0**	**g1**
0	0	0	1	1	32	0	1	1	0
1	1	1	0	0	33	1	0	0	1
2	1	0	0	1	34	1	1	0	0
3	0	1	1	0	35	0	0	1	1
4	0	0	1	1	36	0	1	1	0
5	1	1	0	0	37	1	0	0	1
6	1	0	0	1	38	1	1	0	0
7	0	1	1	0	39	0	0	1	1
8	1	1	0	0	40	1	0	0	1
9	0	0	1	1	41	0	1	1	0
10	0	1	1	0	42	0	0	1	1
11	1	0	0	1	43	1	1	0	0
12	1	1	0	0	44	1	0	0	1
13	0	0	1	1	45	0	1	1	0
14	0	1	1	0	46	0	0	1	1
15	1	0	0	1	47	1	1	0	0
16	1	1	0	0	48	1	0	0	1
17	0	0	1	1	49	0	1	1	0
18	0	1	1	0	50	0	0	1	1
19	1	0	0	1	51	1	1	0	0
20	1	1	0	0	52	1	0	0	1
21	0	0	1	1	53	0	1	1	0
22	0	1	1	0	54	0	0	1	1
23	1	0	0	1	55	1	1	0	0
24	0	0	1	1	56	0	1	1	0
25	1	1	0	0	57	1	0	0	1
26	1	0	0	1	58	1	1	0	0

27	0	1	1	0	59	0	0	1	1
28	0	0	1	1	60	0	1	1	0
29	1	1	0	0	61	1	0	0	1
30	1	0	0	1	62	1	1	0	0
31	0	1	1	0	63	0	0	1	1

7.3.4. Puncturing

The puncture pattern applied to the outputs of the convolutional encoders, C1 and C2, is described by the puncturing arrays P1 and P2 respectively. The patterns P1 and P2 are binary-valued arrays with sizes equaling the size of the output sequences produced by C1 and C2 respectively. Therefore the array described by P1 has 162 elements and that by P2 has 112 elements. A "0" at a particular index of the array implies that the corresponding output bit will not be transmitted.

The following tables describe the arrays P1 and P2. Index into the puncturing array increases row-wise. Index "0" is the first element of the first row, while the last index is the last element of the last row.

Table 7.3.4-1: Puncture Pattern P1 for Encoder C1 (base-to-mobile)

1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,
1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0	1,1,1,0,1,1,1,0
1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,
1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1			

The puncturing specified by P1 requires that every fourth output bit of C1 is punctured, thereby achieving an effective rate of 2/3 for C1.

Table 7.3.4-2: Puncture Pattern P2 for Encoder C2 (base-to-mobile)

1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,0,1,0,	1,1,1,0,1,0,1,0,
1,1,1,0,1,0,1,0	1,1,1,0,1,0,1,0,	1,1,1,0,1,0,1,0,	1,1,1,0,1,0,1,0,	1,1,1,0,1,0,1,0,	1,1,1,0,1,1,1,0,
1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,				

C2 produces 77 output bits for 56 input bits.

7.3.5. Interleaving

Figure 7.3.5-1: Three-slot interleaving for base-to-mobile transmission

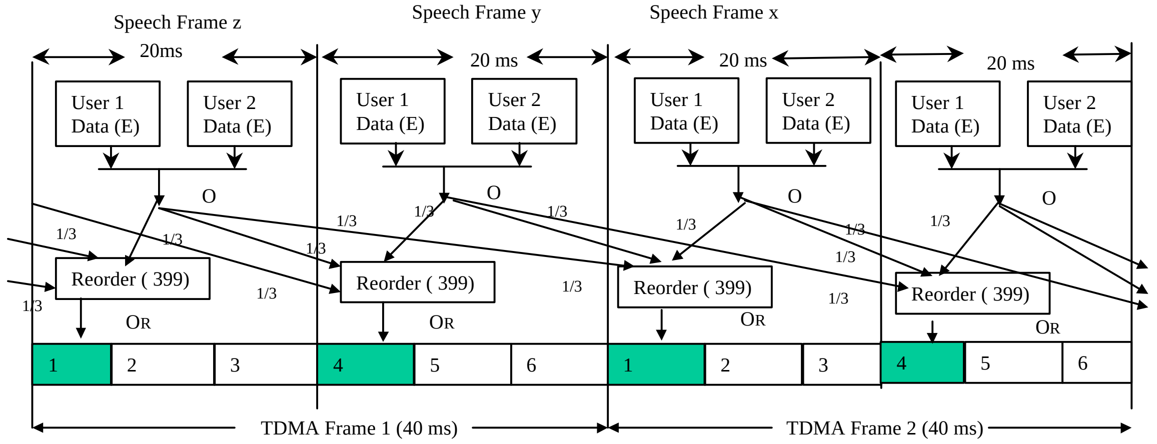


Figure 7.3.5-1 serves an example to demonstrate the principles of three-slot-interleaving. Here time-slots 1 and 4 are assigned for transmission and two users are required to co-share the assigned time-slots. Alternative assignments using slots 2, 5 and 3,6 are also possible, [4]. The three-slot interleaving technique spreads the encoded data sequences of both the users over three consecutive time slots assigned. Each user produces an encoded data frame comprising 199 bits at 20-millisecond intervals. In Figure 7.3.5-1, it is assumed that the encoded data frames shown are ready for transmission; that is the initial 20-millisecond time-period required to buffer the speech data is not shown. The combined encoded output of both users produced after a 20-millisecond period, is labeled O(0)-O(397). An additional output bit, O(398) = 0, is added to match the slot capacity of 399 data bits. The 399 data bits produced are divided into three segments to be transmitted over three consecutive slots. The data to be transmitted in an assigned slot is reordered according to the interleaving array defined in Table 7.3.5-1. Within a time slot only one third of the encoded data output produced by both users, which is available for transmission in the most recent speech frame, is transmitted. This most recently encoded data is combined with one third of the encoded data available in the previous 20-millisecond-interval, and also with one third of the encoded data available 40 milliseconds prior to the most recent speech frame.

The interleaving array determining the data to be transmitted in a given slot is defined in Table 7.3.5-1. The frame “x” refers to the most recent speech frame. The frame “y” refers to the previous speech frame that starts 20 milliseconds prior to the most recent speech frame. Finally the frame “z” refers to the speech frame that starts 40 milliseconds prior to the most recent speech frame.

Table 7.3.5-1: Interleaving Table for base-to-mobile transmission

Row Number	Number of Elements	Frame	Output Bits
0	36	z	O(0), O(12), O(129), O(199), O(211), O(328), O(24), O(36), O(141), O(223), O(235), O(340), O(48), O(60), O(153), O(247), O(259), O(352), O(72), O(84), O(165), O(271), O(283), O(364), O(96), O(108), O(177), O(295), O(307), O(376), O(120), O(126), O(189), O(319), O(325), O(388)
1	36	y	O(1), O(13), O(130), O(200), O(212), O(329), O(25), O(37), O(142), O(224), O(236), O(341), O(49), O(61), O(154), O(248), O(260), O(353), O(73), O(85), O(166), O(272), O(284), O(365), O(97), O(109), O(178), O(296), O(308), O(377), O(121), O(127), O(190), O(320), O(326), O(389),

2	36	x	O(2), O(14), O(131), O(201), O(213), O(330), O(26), O(38), O(143), O(225), O(237), O(342), O(50), O(62), O(155), O(249), O(261), O(354), O(74), O(86), O(167), O(273), O(285), O(366), O(98), O(110), O(179), O(297), O(309), O(378), O(122), O(128), O(191), O(321), O(327), O(390)
3	36	z	O(3), O(15), O(132), O(202), O(214), O(331), O(27), O(39), O(144), O(226) , O(238), O(343), O(51), O(63), O(156), O(250), O(262), O(355), O(75), O(87), O(168), O(274), O(286), O(367), O(99), O(111), O(180), O(298), O(310), O(379), O(123), O(195), O(192), O(322), O(394), O(391),
4	36	y	O(4), O(16), O(133), O(203), O(215), O(332), O(28), O(40), O(145), O(227), O(239), O(344), O(52), O(64), O(157), O(251), O(263), O(356), O(76), O(88), O(169), O(275), O(287), O(368), O(100), O(112), O(181), O(299), O(311), O(380), O(124), O(196), O(193), O(323), O(395), O(392),
5	36	x	O(5), O(17), O(134), O(204), O(216), O(333), O(29), O(41), O(146), O(228), O(240), O(345), O(53), O(65), O(158), O(252), O(264), O(357), O(77), O(89), O(170), O(276), O(288), O(369), O(101), O(113), O(182), O(300), O(312), O(381), O(125), O(197), O(194), O(324), O(396), O(393),
6	30	z	O(6), O(18), O(135), O(205), O(217), O(334), O(30), O(42), O(147), O(229), O(241), O(346), O(54), O(66), O(159), O(253), O(265), O(358), O(78), O(90), O(171), O(277), O(289), O(370), O(102), O(114), O(183), O(301), O(313), O(382),
7	30	y	O(7), O(19), O(136), O(206), O(218), O(335), O(31), O(43), O(148), O(230), O(242), O(347), O(55), O(67), O(160), O(254), O(266), O(359), O(79), O(91), O(172), O(278), O(290), O(371), O(103), O(115), O(184), O(302), O(314), O(383),
8	30	x	O(8), O(20), O(137), O(207), O(219), O(336), O(32), O(44), O(149), O(231), O(243), O(348), O(56), O(68), O(161), O(255), O(267), O(360), O(80), O(92), O(173), O(279), O(291), O(372), O(104), O(116), O(185), O(303), O(315), O(384),
9	30	z	O(9), O(21), O(138), O(208), O(220), O(337), O(33), O(45), O(150), O(232), O(244), O(349), O(57), O(69), O(162), O(256), O(268), O(361), O(81), O(93), O(174), O(280), O(292), O(373), O(105), O(117), O(186), O(304), O(316), O(385),
10	30	y	O(10), O(22), O(139), O(209), O(221), O(338), O(34), O(46), O(151), O(233), O(245), O(350), O(58), O(70), O(163), O(257), O(269), O(362), O(82), O(94), O(175), O(281), O(293), O(374), O(106), O(118), O(187) ,O(305), O(317), O(386),
11	30	x	O(11), O(23), O(140) , O(210), O(222), O(339), O(35), O(47) ,O(152), O(234) ,O(246), O(351), O(59), O(71), O(164) ,O(258), O(270), O(363), O(83), O(95), O(176), O(282), O(294), O(375) O(107), O(119), O(188), O(306), O(318), O(387),
12	3	x	O(198), O(397), O(398)

The output data bits from the three frames x, y and z, (numbered O(0)-O(198) corresponding to the first user and O(199)-O(397) belonging to the second user), are placed into the interleaving array as shown in Table 7.3.5-1. The data bits are then read row-wise. Hence, the first 36 data bits to be transmitted will be the output bits described by the first row of the interleaving array (row number 0), starting from bits O(0) and ending with O(388) of speech frame 'z". The next 36 data bits will be read from row number 1 until finally the bits in row number 12, O(198), O(397) and O(398), belonging to speech frame "x", are transmitted.

7.3.6. Symbol Mapping

The required 8-PSK mapping is described in proposed changes to TIA/EIA 136-131, [4].

[Editor's Note: Section 7.3.6 is informative and should not be included in the actual standard.]

7.4. Channel Decoding for Base-to-Mobile Transmission

In the channel decoder, the data is first deinterleaved to reconstruct the data frame. This is carried out as an inverse process of the interleaving specified in Section 7.3.5. If encryption was carried out prior to interleaving, the data must be decrypted at this stage. Then the convolutionally encoded data must be decoded. A number of techniques are applicable for decoding. For instance, a Viterbi decoder exploiting soft-channel information may be used.

The pre-ordering applied prior to the convolutional encoding must be reversed before the CRC is calculated from the decoded Class 1A bits. The same CRC generator polynomial is used as described in Section 7.3.1. The calculated CRC must be compared with the received CRC to check the validity of the Class 1A frame. The result of this validity check must be provided to the speech decoder.

7.5. Channel Encoding for Mobile-to-Base Transmission

Figure 7.5-1: Channel encoding for mobile-to-base transmission

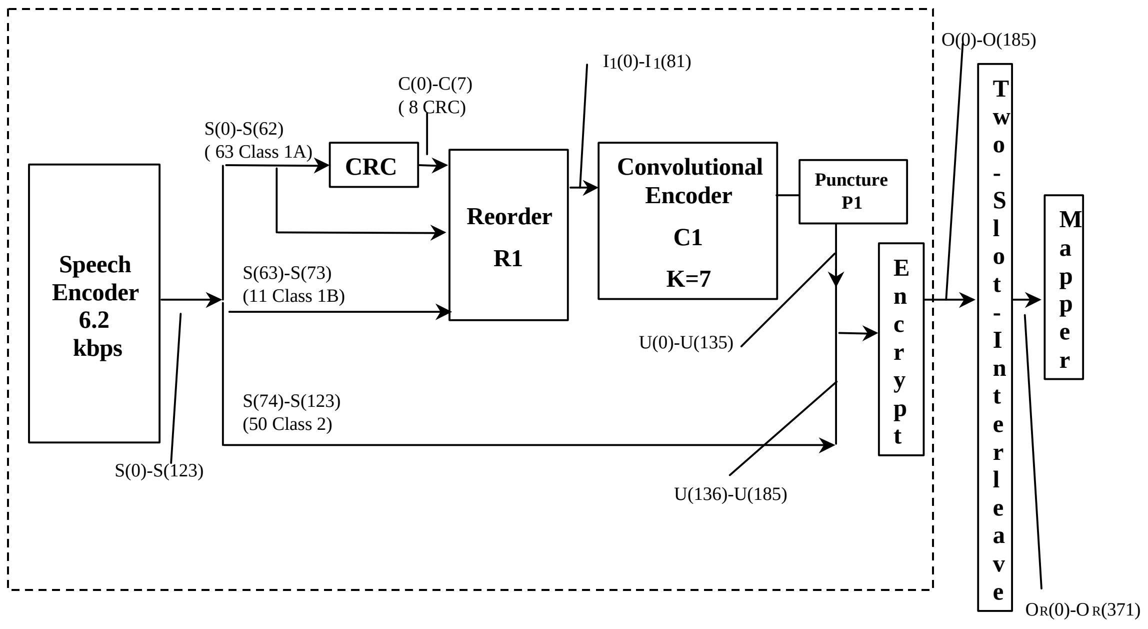


Figure 7.5-1 describes the channel encoding provided for mobile-to-base transmission. An 8 bit CRC is computed over the 63 Class 1A bits to produce 8 CRC bits labeled C(0)-C(7). The Class 1A, Class 1B and the CRC bits are combined and convolutionally encoded together as part of a single data frame. Prior to encoding, the bits are reordered according to R1. The input to the convolutional encoder, C1, comprises 82 bits labeled as $I_1(0)$-$I_1(81)$. A tail-biting code is used for C1. The encoder C1 is a rate ½,

constraint length 7 code, which is punctured according to P1 to produce 136 output data bits, U(0)-U(135).

The 50 Class 2 bits are left uncoded. They are labeled U(136)-U(185) and a total of 186 encoded data bits are produced per speech frame. If data must be encrypted prior to transmission, encryption must be done prior to interleaving.

The encrypted (or clear) output bits O(0)-O(185) are combined with other encoded frames of the same user and interleaved over two slots. The interleaving technique is based on combining two encoded data frames, produced within a 40-millisecond TDMA frame, and spreading it over two consecutive time slots assigned to each user.

The TDMA half-rate solution for mobile-to-base-transmissions, works in conjunction with 8-PSK modulation and the 8-PSK-slot format defined for the mobile-to-base link in TIA/EIA 136-131, [2]. Therefore the output of the interleaver, OR(0)-OR(371), is mapped into complex symbols using an 8-PSK mapper. *[Editor's Note: This paragraph is informative and should not be included in the actual standard.]*

7.5.1. Cyclic Redundancy Check (CRC)

An 8-bit CRC checksum is computed for the 63 Class 1A bits in the frame. The CRC parity bits are calculated according to the following CRC polynomial.

$$g(X) = 1 + X^2 + X^2 + X^4 + X^6 + X^7 + X^8 \qquad (7.5.1\text{-}1)$$

The input polynomial is based on S(0)-S(62) and is described in Section 7.3.1.

The CRC encoding and decoding procedures are consistent with EIA/TIA-136-410, [1].

7.5.2. Pre-Encoder Bit Ordering

The bit order presented to the convolutional encoder C1 is described by the array R1. The sequence I1 can be computed via the following equation.

$$I_1(j) = R1(j) \quad j = 0, \ldots, 81 \qquad (7.5.2\text{-}1)$$

Table 7.5.2-1: Bit Order for C1 described by R1 (mobile-to-base)

C(0), C(1), C(2), C(3), S(0), S(2), S(4), S(6), S(8), S(10), S(12), S(14), S(16), S(18), S(20), S(22), S(24), S(26), S(28), S(30), S(32), S(36), S(38), S(40), S(42), S(44), S(46), S(48), S(50), S(52), S(54), S(56), S(58), S(60), S(62), S(64), S(66), S(68), S(70), S(72), S(73),S(71), S(69), S(67), S(65), S(63), S(61), S(59), S(57), S(55), S(53), S(51), S(49), S(47), S(45),S(43), S(41), S(39), S(37), S(35), S(33), S(31), S(29), S(27), S(25), S(23), S(21), S(19), S(17), S(15), S(13), S(11), S(9), S(7), S(5), S(3), S(1),C(4), C(5), C(6), C(7)

Table 7.5.2-1 describes the R1 array. The index into the array is assumed to increase row-wise. Index "0" is the first element of the first row, and index "81" is the last element of the last row.

7.5.3. Convolutional Encoding

The convolutional encoder C1 is identical to that defined for the base-to-mobile case in Section 7.3.3.

7.5.4. Puncturing

The puncture pattern applied to the output of the convolutional encoder C1 is described by the array P1. P1 is a binary-valued sequence, with size equaling the size of the output sequence produced by C1. Therefore the sequence described by P1 has 164 elements. A “0” at a particular location index implies that the corresponding output bit will not be transmitted.

Table 7.5.4-1 describes P1. Index into the puncturing array increases row-wise. The index “0” is the first element of the first row, while the index 163 is the last element of the last row.

Table 7.5.4-1: Puncture Pattern P1 for Encoder C1 (mobile-to-base)

1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,
1,1,1,1,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,
1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,0,1,1,1,0,	1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,
1,1,1,1,1,1,1,0,	1,1,1,1,1,1,1,0,	1,1,1,1,			

The pattern specified by P1 produces 136 output bits for 82 input bits.

7.5.5. Interleaving

Figure 7.5.5-1: Two –slot interleaving for mobile-to-base transmission

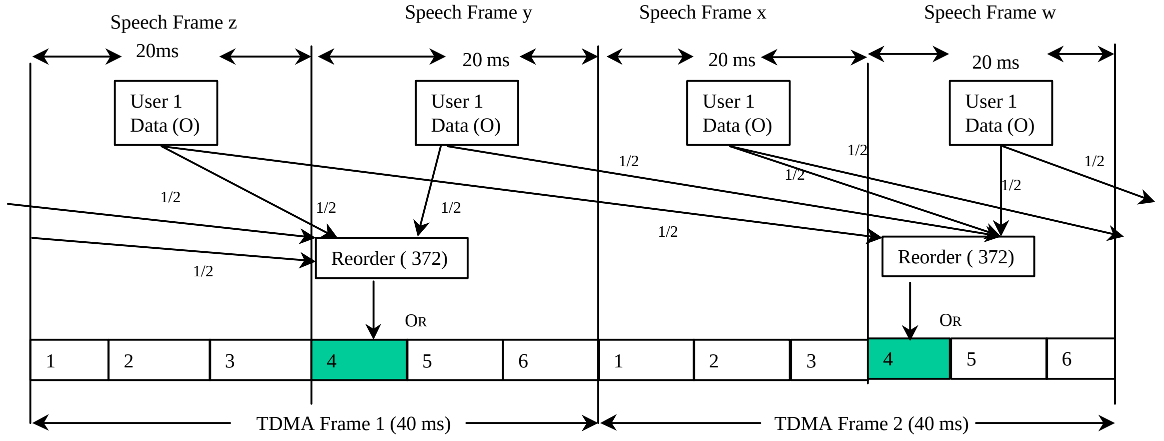


Figure 7.5.5-1 illustrates an example two-slot interleaving scenario for mobile-to-base transmission. Each time-slot carries half of four different encoded data frames of the assigned user, which are generated after

four 20-millisecond time-periods (two 40-millisecond TDMA time frames). It is assumed that encoded data shown is ready for transmission in the next available time-slot; that is the speech buffering delay of 20 milliseconds is not shown.

The interleaving array determining the data to be transmitted in a given slot is defined in Table 7.5.5-1. Frame "w" refers to the most recent speech frame. Frame "x" refers to the previous speech frame, produced 20 milliseconds prior to the most recent frame. Finally, the frames "y" and "z" refer to the frames produced 40 and 60 milliseconds prior to the most recent frame, respectively.

Table 7.5.5-1: Interleaving Table for mobile-to-base transmission

Row Number	Number Of Elements	Frame Number	Output Bits
0	21	w	O(0),O(13),O(8),O(33),O(53),O(28),O(73),O(93),O(48),O(113),O(129),O(68), O(138),O(146),O(84),O(154),O(162),O(100),O(170),O(178),O(116),
1	21	x	O(0),O(13),O(8),O(33),O(53),O(28),O(73),O(93),O(48),O(113),O(129),O(68), O(138),O(146),O(84),O(154),O(162),O(100),O(170),O(178),O(116),
2	21	z	O(121),O(134),O(76),O(142),O(150),O(92),O(158),O(166),O(108),O(174),O(182),O(124), O(5),O(23),O(18),O(43),O(63),O(38),O(83),O(103),O(58),
3	21	y	O(121),O(134),O(76),O(142),O(150),O(92),O(158),O(166),O(108),O(174),O(182),O(124), O(5),O(23),O(18),O(43),O(63),O(38),O(83),O(103),O(58),
4	21	w	O(1),O(15),O(10),O(35),O(55),O(30),O(75),O(95),O(50),O(115),O(131),O(70), O(139),O(147),O(86),O(155),O(163),O(102),O(171),O(179),O(118),
5	21	x	O(1),O(15),O(10),O(35),O(55),O(30),O(75),O(95),O(50),O(115),O(131),O(70), O(139),O(147),O(86),O(155),O(163),O(102),O(171),O(179),O(118),
6	21	z	O(123),O(135),O(78),O(143),O(151),O(94),O(159),O(167),O(110),O(175),O(183),O(126), O(6),O(25),O(20),O(45),O(65),O(40),O(85),O(105),O(60),
7	21	y	O(123),O(135),O(78),O(143),O(151),O(94),O(159),O(167),O(110),O(175),O(183),O(126), O(6),O(25),O(20),O(45),O(65),O(40),O(85),O(105),O(60),
8	21	w	O(2),O(17),O(12),O(37),O(57),O(32),O(77),O(97),O(52),O(117),O(132),O(72), O(140),O(148),O(88),O(156),O(164),O(104),O(172),O(180),O(120),
9	21	x	O(2),O(17),O(12),O(37),O(57),O(32),O(77),O(97),O(52),O(117),O(132),O(72), O(140),O(148),O(88),O(156),O(164),O(104),O(172),O(180),O(120),
10	21	z	O(125),O(136),O(80),O(144),O(152),O(96),O(160),O(168),O(112),O(176),O(184),O(128), O(7),O(27),O(22),O(47),O(67),O(42),O(87),O(107),O(62),
11	21	y	O(125),O(136),O(80),O(144),O(152),O(96),O(160),O(168),O(112),O(176),O(184),O(128), O(7),O(27),O(22),O(47),O(67),O(42),O(87),O(107),O(62),
12	21	w	O(3),O(19),O(14),O(39),O(59),O(34),O(79),O(99),O(54),O(119),O(133),O(74) ,O(141),O(149),O(90),O(157),O(165),O(106),O(173),O(181),O(122),
13	21	x	O(3),O(19),O(14),O(39),O(59),O(34),O(79),O(99),O(54),O(119),O(133),O(74) ,O(141),O(149),O(90),O(157),O(165),O(106),O(173),O(181),O(122),
14	21	z	O(127),O(137),O(82),O(145),O(153),O(98),O(161),O(169),O(114),O(177),O(185),O(130), O(9),O(29),O(24),O(49),O(69),O(44),O(89),O(109),O(64),
15	21	y	O(127),O(137),O(82),O(145),O(153),O(98),O(161),O(169),O(114),O(177),O(185),O(130), O(9),O(29),O(24),O(49),O(69),O(44),O(89),O(109),O(64),

16	9	w	O(4),O(21),O(16),O(41),O(61),O(36),O(81),O(101),O(56),
17	9	x	O(4),O(21),O(16),O(41),O(61),O(36),O(81),O(101),O(56),
18	9	z	O(11),O(31),O(26),O(51),O(71),O(46),O(91),O(111),O(66),
20	9	y	O(11),O(31),O(26),O(51),O(71),O(46),O(91),O(111),O(66),

The output data bits available within the four frames, w, x, y and z, (numbered O(0)-O(185) belonging to the assigned user are placed into the interleaving array as shown. The data bits are then read row-wise. The first 21 data bits to be transmitted are the bits in first row of the interleaving array (row number 0); starting with bit O(0) and ending with bit O(116) of the frame “w”. The next 21 data bits are read from row number 1 until finally the bits in row number 20, O(91), O(111) and O(66), of the frame “y” are transmitted.

7.5.6. Symbol Mapping

8-PSK symbol mapping as described in Section 7.3.6 is used.

[Editor’s Note: Section 7.5.6 is informative and should not be included in the actual standard.]

7.6. Channel Decoding for Mobile-to-Base Link

In the channel decoder, the data is first deinterleaved to reconstruct the data frame. This is carried out as an inverse process of the interleaving specified in Section 7.5.5. If encryption was carried out prior to interleaving, the data must be decrypted at this stage. Then the convolutionally encoded portion of the data must be decoded. A number of techniques are applicable for decoding. For instance, a Viterbi decoder exploiting soft-channel information may be used.

The pre-ordering applied prior to the convolutional encoding must be reversed before the CRC is calculated from the decoded Class 1A bits. The same CRC generator polynomial is used as described in Section 7.5.1. The calculated CRC must be compared with the received CRC to check the validity of the Class 1A frame. The result of this validity check must be provided to the speech decoder.

8. References

8.1. Normative References

The following TIA/EIA standard contains provisions, which through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All TIA/EIA standards are subject to revision; all users of this standard are therefore encouraged to investigate the possibility of applying the most recent edition of the TIA/EIA standard listed below.

1. TIA/EIA 136-410 (1999), TDMA Cellular/PCS – Radio Interface Enhanced Full-Rate Speech Codec.

2. TIA/EIA 136-131 (1999), Digital Traffic Channel Layer 1

8.2. Informative References

3. ITU-T Recommendation G.729 Annex D (1998), *6.4 kbit/s CS-ACELP speech coding algorithm*.

4. UWCC.GTF.HRP.99.05.26_, Lucent Technologies, (1999), *Proposed changes to TIA/EIA 136-131 to include TDMA half rate speech codec*.