



INTERNATIONAL TELECOMMUNICATION UNION

# ITU-T G.722.1

## Implementers Guide

TELECOMMUNICATION  
STANDARDIZATION SECTOR  
OF ITU

(25 October 2002)

SERIES G: TRANSMISSION SYSTEMS AND MEDIA,  
DIGITAL SYSTEMS AND NETWORKS

Digital terminal equipments – Coding of analogue signals  
by methods other than PCM

---

### **Implementors' Guide for G.722.1 Main Body and Annex B**

***(Coding at 24 and 32 kbit/s for hands-free  
operation in systems with low frame loss)***

---

**Attention:** This is not a publication made available to the public, but **an internal ITU-T Document** intended only for use by the Member States of the ITU, by ITU-T Sector Members and Associates, and their respective staff and collaborators in their ITU related work. It shall not be made available to, and used by, any other persons or entities without the prior written consent of the ITU-T.

## Implementers Guide for Recommendation G.722.1

---

### **Contact Information**

Rapporteur, ITU-T Study  
Group 16 / Question 10

Claude Lamblin  
France Telecom R&D /DIH  
Technopole Anticipa  
2 avenue Pierre Marzin  
22307 LANNION Cedex  
France

Tel: +33 2 96 05 13 03  
Fax: +33 2 96 05 35 30  
Email: [claude.lamblin@rd.francetelecom.com](mailto:claude.lamblin@rd.francetelecom.com)

Editor, ITU-T  
Recommendation G.722.1  
and Implementors' Guide

David Lindbergh  
Polycom Inc.  
100 Minuteman Road  
Andover, MA 01810  
USA

Tel: +1 978 292 5366  
Email: [lindbergh@92F1.com](mailto:lindbergh@92F1.com)

### **SUMMARY**

## **Implementors' Guide for Recommendation G.722.1**

This document contains the Implementers' Guide for the software C-code of ITU-T Recommendation G.722.1 and its Annex B that corrects defects reported at SG 16's meeting on 15-25 October 2002.

## **Table of Contents**

|                                                                        |    |
|------------------------------------------------------------------------|----|
| Implementors' Guide for Recommendation G.722.1 .....                   | ii |
| Implementers' Guide for G.722.1 and G.722.1 Annex B. ....              | 1  |
| 1.0 Summary.....                                                       | 1  |
| 2.0 Correction to fixed-point C source code of G.722.1 .....           | 1  |
| 2.1 Change to defs.h .....                                             | 1  |
| 2.2 Changes to decoder.c.....                                          | 1  |
| 3.0 Correction to floating-point C source code of G.722.1 Annex B..... | 2  |
| 3.1 Changes to decoder.c.....                                          | 2  |
| Annex A C-code attachment with corrections .....                       | 4  |

## Implementers' Guide for G.722.1 and G.722.1 Annex B.

## 1.0 Summary

This document is the Implementers' Guide for the software C-code of ITU-T Recommendation G.722.1 and its Annex B.

## 2.0 Correction to fixed-point C source code of G.722.1

In the fixed-point C source code of G.722.1, two files are changed:

- defs.h
- decoder.c

As described in [COM16-D237](#), these changes correct a serious problem where bit errors in the coded stream were not being detected in the decoder.

## 2.1 Change to defs.h

In file “defs.h”, in line 175, the variable *frame\_error\_flag* is defined as a pointer instead of an automatic variable, as shown below:

[illegible]

## 2.2 Changes to decoder.c

In file “decoder.c”, the same change involving the variable *frame\_error\_flag* is made in six places, in lines 129, 742, 771, 792, 806, and 824, as shown below:

[illegible]

```
*****

***** decoder.c.old
770: void test_4_frame_errors(Bit_Obj *bitobj,
771:                          Word16 frame_error_flag,
772:                          Word16 categorization_control,
***** DECODER.C
770: void test_4_frame_errors(Bit_Obj *bitobj,
771:                          Word16 *frame_error_flag,
772:                          Word16 categorization_control,
*****

***** decoder.c.old
791: {
792:     frame_error_flag = 1;
793:     move16();
***** DECODER.C
791: {
792:     *frame_error_flag = 1;
793:     move16();
*****

***** decoder.c.old
805: {
806:     frame_error_flag |= 2;
807:     logic16();
***** DECODER.C
805: {
806:     *frame_error_flag |= 2;
807:     logic16();
*****

***** decoder.c.old
823: {
824:     frame_error_flag |= 4;
825:     logic16();
***** DECODER.C
823: {
824:     *frame_error_flag |= 4;
825:     logic16();
*****
```

### 3.0 Correction to floating-point C source code of G.722.1 Annex B

In the floating-point C source code of G.722.1 Annex B, one file is changed:

- decoder.c

As described in [COM16-D237](#), these changes correct two problems:

- a) The noise fill energy was 26.8 dB too weak on the floating-point decoder, compared to the fixed-point source code. This has been corrected by defining a constant NOISE\_SCALE\_FACTOR, with the value of 22.0, and using this to scale the background noise.
- b) As described in [COM16-D219](#) (France Telecom, 2002-02), there was potential for an array overflow in certain circumstances. This has been corrected in the way suggested in sections 3.2.2 and 3.2.3 of COM16-D219.

### 3.1 Changes to decoder.c

The changes to decoder.c are shown below.

```
Comparing files decoder.c.old and DECODER.C
***** decoder.c.old
54:
```

```
55: #define GET_NEXT_BIT \
**** DECODER.C
54:
55: #define NOISE_SCALE_FACTOR 22.0F
56:
57: #define GET_NEXT_BIT \
****

**** decoder.c.old
541:         n++;
542:         if (fabs(*decoder_mlt_ptr) > 2.0*standard_deviation) {
543:             n += 3;
**** DECODER.C
543:         n++;
544:         if (fabs(*decoder_mlt_ptr) > 44.0F*standard_deviation) {
545:             n += 3;
****

**** decoder.c.old
547:         }
548:         temp1 = noise_fill_factor_cat5[n];
**** DECODER.C
549:         }
550:         if(n>19)n=19;
551:         temp1 = noise_fill_factor_cat5[n];
****

**** decoder.c.old
562:         if ((random_word & 1) == 0) temp1 = noifillneg;
563:         *decoder_mlt_ptr = temp1;
564:         random_word >>= 1;
**** DECODER.C
565:         if ((random_word & 1) == 0) temp1 = noifillneg;
566:         *decoder_mlt_ptr = temp1*NOISE_SCALE_FACTOR;
567:         random_word >>= 1;
****

**** decoder.c.old
572:         if ((random_word & 1) == 0) temp1 = noifillneg;
573:         *decoder_mlt_ptr = temp1;
574:         random_word >>= 1;
**** DECODER.C
575:         if ((random_word & 1) == 0) temp1 = noifillneg;
576:         *decoder_mlt_ptr = temp1*NOISE_SCALE_FACTOR;
577:         random_word >>= 1;
****

**** decoder.c.old
604:         if ((random_word & 1) == 0) temp1 = noifillneg;
605:         *decoder_mlt_ptr = temp1;
606:         random_word >>= 1;
**** DECODER.C
607:         if ((random_word & 1) == 0) temp1 = noifillneg;
608:         *decoder_mlt_ptr = temp1*NOISE_SCALE_FACTOR;
609:         random_word >>= 1;
****

**** decoder.c.old
614:         if ((random_word & 1) == 0) temp1 = noifillneg;
615:         *decoder_mlt_ptr = temp1;
616:         random_word >>= 1;
**** DECODER.C
617:         if ((random_word & 1) == 0) temp1 = noifillneg;
618:         *decoder_mlt_ptr = temp1*NOISE_SCALE_FACTOR ;
619:         random_word >>= 1;
****

**** decoder.c.old
634:         if ((random_word & 1) == 0) temp1 = noifillneg;
635:         *decoder_mlt_ptr++ = temp1;
636:         random_word >>= 1;
**** DECODER.C
637:         if ((random_word & 1) == 0) temp1 = noifillneg;
```

```
638:          *decoder_mlt_ptr++ = temp1*NOISE_SCALE_FACTOR;
639:          random_word >>= 1;
*****

***** decoder.c.old
641:          if ((random_word & 1) == 0) temp1 = noifillneg;
642:          *decoder_mlt_ptr++ = temp1;
643:          random_word >>= 1;
***** DECODER.C
644:          if ((random_word & 1) == 0) temp1 = noifillneg;
645:          *decoder_mlt_ptr++ = temp1*NOISE_SCALE_FACTOR;
646:          random_word >>= 1;
*****
```

## Annex A C-code attachment with corrections

The following electronic attachment contains the corrected files



G.722.1 Corrections\_oct02.zip

Electronic Attachment

--- END ---

---