NEC

User's Manual

μSAP77016-B04

G.723.1 Speech Codec Middleware

Target Devices

μPD77018
μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113
μPD77114
μPD77116

Document No. U13955EJ2V0UMJ1 (2nd edition)
Date Published October 2000 N CP (K)

[MEMO]

Windows is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

- **The information in this document is current as of May, 2000. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC's data sheets or data books, etc., for the most up-to-date specifications of NEC semiconductor products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC. NEC assumes no responsibility for any errors that may appear in this document.
- NEC does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC semiconductor products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC endeavours to enhance the quality, reliability and safety of NEC semiconductor products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC semiconductor products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment, and anti-failure features.
- NEC semiconductor products are classified into the following three quality grades:
  "Standard", "Special" and "Specific". The "Specific" quality grade applies only to semiconductor products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of a semiconductor product depend on its quality grade, as indicated below. Customers must check the quality grade of each semiconductor product before using it in a particular application.
  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.
  The quality grade of NEC semiconductor products is "Standard" unless otherwise expressly specified in NEC's data sheets or data books, etc. If customers wish to use NEC semiconductor products in applications not intended by NEC, they must contact an NEC sales representative in advance to determine NEC's willingness to support a given application.
  (Note)
  (1) "NEC" as used in this statement means NEC Corporation and also includes its majority-owned subsidiaries.
  (2) "NEC semiconductor products" means any semiconductor product developed or manufactured by or for NEC (as defined above).

M8E 00. 4

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Madrid Office
Madrid, Spain
Tel: 91-504-2787
Fax: 91-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Guarulhos-SP Brasil
Tel: 55-11-6462-6810
Fax: 55-11-6462-6829

J00.7

Major Revisions in This Edition

Page	Description
Throughout	Addition of μPD77113 and 77114 to target devices
p.27	Addition of description to **2.3.1 Extraction parameters for G.723.1 ANNEX A functions**
p.28	Addition of description to **2.3.2 Extraction parameters for G.723.1 ANNEX C functions**
p.33	Addition of description to **2.4.2 (2) SID frame compression format**

The mark ⋆ shows major revised points.

[MEMO]

INTRODUCTION

Readers

This manual has been prepared for users who design and develop application systems using the μPD77016 Family.

The μPD77016 Family includes the following: μPD77015, 77016, 77017, 77018, 77018A, 77019, 77110, 77111, 77112, 77113, 77114, and 77116[Note]. The target devices of this manual, however, only include the μPD77018, 77018A, 77019, 77110, 77111, 77112, 77113, 77114, and 77116.

Note Under development

Purpose

The manual is intended to give users an understanding of how to use the middleware that is used for support when designing or developing application systems using the μPD77016 Family.

Organization

This manual is broadly divided into the following sections.

Chapter 1 Outline
Chapter 2 Library Specifications
Chapter 3 Installation
Appendix Sample Program Source

How to Read This Manual

It is assumed that the readers of this manual have general knowledge of electrical engineering, logic circuits, microcontrollers, and C language.

To learn the hardware functions of the μPD7701× Family
→ Refer to the **μPD7701× Family User's Manual Architecture**

To learn the hardware functions of the μPD77111 Family
→ Refer to the **μPD77111 Family User's Manual Architecture**

To learn the instruction functions of the μPD77016 Family
→ Refer to the **μPD77016 Family User's Manual Instructions**

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	$\overline{\times\times\times}$ (overscore over pin or signal name)
Note:	Footnote for item marked with **Note** in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numerical representation:	Binary···×××× or 0b××××
	Decimal···××××
	Hexadecimal···0x××××

Related Documents The related documents listed below may include preliminary versions. However, preliminary versions are not marked as such.

Documents Related to Devices

Document Name / Part Number	Pamphlet	Data Sheet	User's Manual		Application Note
			Architecture	Instructions	Basic Software
μPD77016	U12395E	U10891E	U10503E	U13116E	U11958E
μPD77015		U10902E			
μPD77017					
μPD77018					
μPD77018A		U11849E			
μPD77019					
μPD77019-013		U13053E			
μPD77110		U12801E	Under preparation		
μPD77111					
μPD77112					
μPD77113		U14373E			
μPD77114					
μPD77116	—	U14624E	—	—	—

Documents Related to Development Tools

Document Name		Document No.
IE-77016-98, IE-77016-PC User's Manual	Hardware	U13044E
IE-77016-CM-LC User's Manual		U14139E
RX77016 User's Manual	Function	U14397E
	Configuration Tool	U14404E
RX77016 Application Note	HOST API	U14371E

Caution The documents listed above are subject to change without notice. Be sure to use the latest documents when designing.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 OUTLINE

1.1 Middleware

Middleware is the name given to a group of software that has been tuned so that it draws out the maximum performance of the processor and enables processing that is conventionally performed by hardware to be performed by software. The concept of middleware was introduced with the development of a new high-performance processor, the DSP, in order to facilitate operation of the environments integrated in the system.

By providing appropriate speech codec and image data compression/decompression-type middleware, NEC is offering users the kind of technology essential in the realization of a multimedia system for the μPD77016 Family, and is continuing promotion of system development.

The μSAP77016-B04 is a middleware product that provides ITU-T[Note]-recommended G.723.1 speech compression and decompression functions (including the recommended additions ANNEX A and ANNEX C). Unless otherwise stated, when describing this middleware, it is assumed the speech codec used is G.723.1.

Note International Telecommunication Union-Telecommunication Standardization Sector

1.2 G.723.1 Speech Codec

The G.723.1 speech codec is the 5.3 Kbps or 6.3 Kbps speech compression/decompression codec recommended by ITU-T, and is an algorithm for coding speech data that use ACELP and MP-MLQ.

By means of a telephone band filter (the ITU-T recommended G.712), the G.723.1 speech codec samples band-restricted analog input signals at 8 kHz. The digital signals that are obtained by converting this sampled data into 16-bit linear PCM data are then designed so that they operate as encoder inputs. Similarly, it is necessary to return the signals to analog form to output them from the decoder.

Signals in other I/O formats, such as the 64 Kbps PCM data prescribed by the ITU-T-recommended G.711, must be converted into 16-bit linear PCM data before coding, and converted back from 16-bit linear PCM data to a suitable format after decoding. The bit-string that is passed from the encoder to the decoder is defined by the ITU-T recommended G.723.1.

Remark ACELP: Algebraic Code Excited Linear Prediction
MP-MLQ: Multi-Pulse Maximum Likelihood Quantization

1.3 G.723.1 ANNEX A Speech Codec

ANNEX A, which is an additional recommendation to the ITU-T recommended G.723.1, is the silence compression function of the G.723.1, and is used in addition to the standard G.723.1 speech codec. Speech codecs with an additional silence compression function and those without an additional silence compression function cannot, however, be interconnected. The purpose of the silence compression function is to raise the total compression rate and reduce the bit rate by raising the compression rate in the parts where there is silence.

1.4 G.723.1 ANNEX C Speech Codec

ANNEX C, which is an additional recommendation to the ITU-T recommended G.723.1, is the variable bit rate channel codec of the G.723.1 speech codec, and is used in addition to the standard G.723.1 and ANNEX A. The bit rate supported by ANNEX C is 0.7 Kbps to 14.3 Kbps.

ANNEX C is designed as a part of the ITU-T H.324 Family, which is aimed at multimedia in mobile communications. Using this standard, the G.723.1 can be adapted to all wire and wireless transmission systems, although ANNEX C does not define a standard for functions that rely on transmission systems such as interleave or burst formatting systems.

1.5 System Outline

1.5.1 Features

- Compression coding at 5.3 and 6.3 Kbps (variable bit rate for ANNEX C)
- High bit-rate speech coding
- Codes and decodes 240 samples/frames at an 8 kHz sampling frequency
- All speech I/O data is 16-bit linear data

1.5.2 Operating environment

★ **(1) Target DSPs**

μPD77018
μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113
μPD77114
μPD77116 (Under development)

(2) Required memory

Table 1-1. Required Memory Capacity

Memory	Type	ANNEX A [Word]	ANNEX C [Word]
Instruction memory	–	5.6 K	7.9 K
X memory	RAM[Note]	2.1 K	2.2 K
	ROM	3.9 K	5.6 K
Y memory	RAM[Note]	2.0 K	2.0 K
	ROM	5.6 K	6.5 K

Note The scratch area within the RAM area is X: 1024 words, Y: 1024 words in ANNEX A, and X: 1108 words, Y: 284 words in ANNEX C.

(3) Software tools (Windows™)

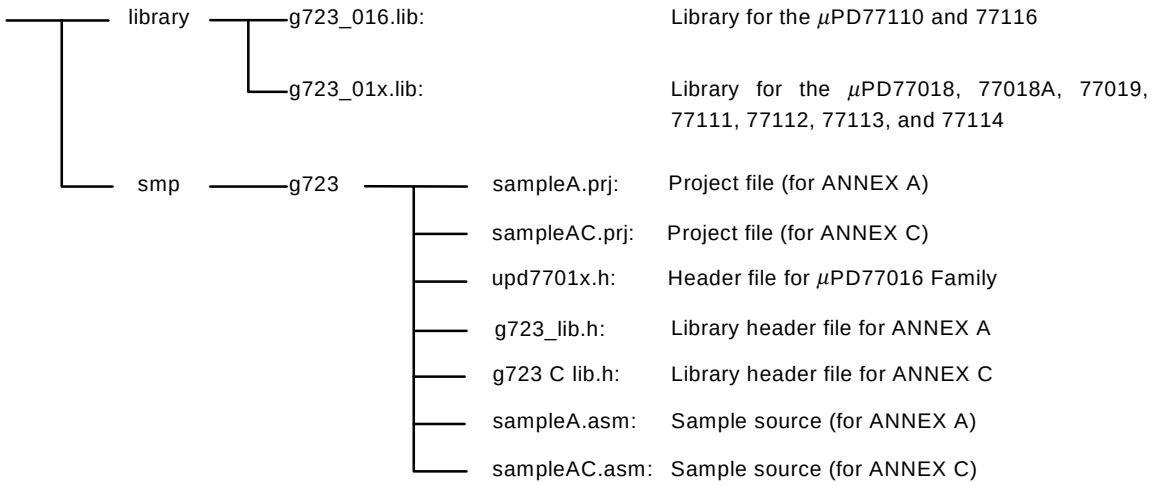
DSP tools: WB77016 (Workbench)
HSM77016 (High-speed simulator)

1.5.3 Performance

[Conditions] DSP: μPD77016 Family (33 MIPS when operating at 33 MHz)
[Necessary MIPS value for execution of a 1-frame process in real-time (30 ms)]
(G.723.1 ANNEX A) Compression processing: 17.5 MIPS Decompression processing: 1.5 MIPS
(G.723.1 ANNEX C) Compression processing: +1 MIPS Decompression processing: +1.5 MIPS

1.5.4 Directory configuration

The directory configuration of this middleware is shown below.



Library files g723_016.lib and g723_01x.lib are included in the following object file. Note that g723_016.lib cannot be used for the μPD77016, and g723_01x.lib cannot be used for the μPD77015 and 77017.

- g723aenc.rel (G.723.1 ANNEX A Encoder)
- g723adec.rel (G.723.1 ANNEX A Decoder)
- g723acom.rel (G.723.1 ANNEX A Encoder and Decoder shared section)
- g723cenc.rel (G.723.1 ANNEX C Encoder)
- g723cdec.rel (G.723.1 ANNEX C Decoder)
- g723ccom.rel (G.723.1 ANNEX C Encoder and Decoder shared section)

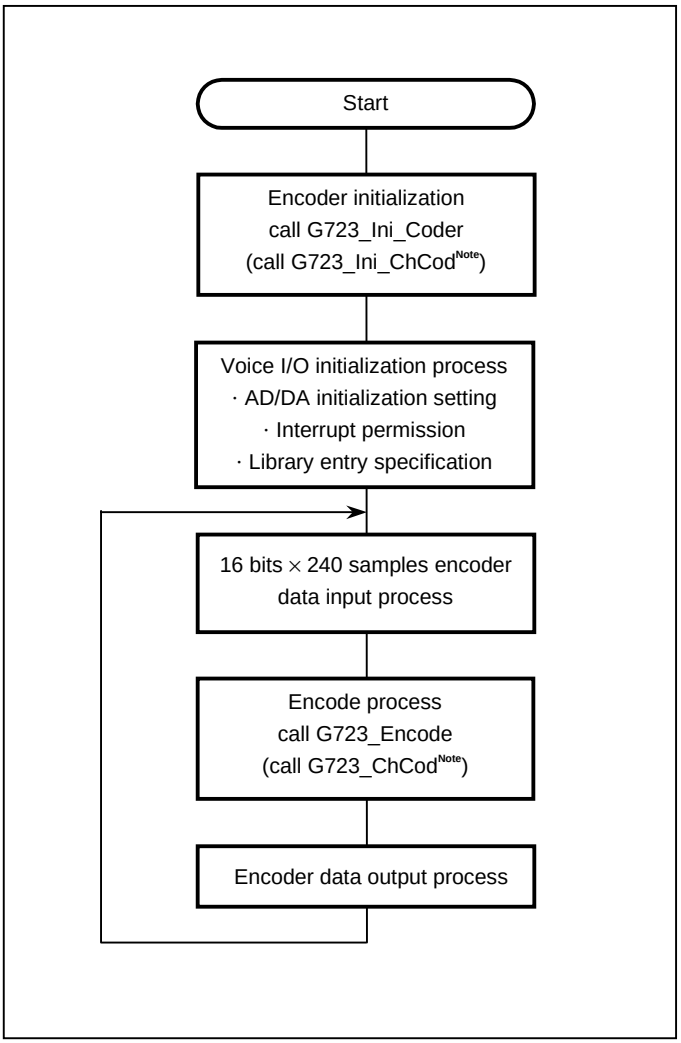
CHAPTER 2 LIBRARY SPECIFICATIONS

This chapter describes the function specifications and call conventions of this middleware.

2.1 G.723.1 Speech Codec Processing Flow

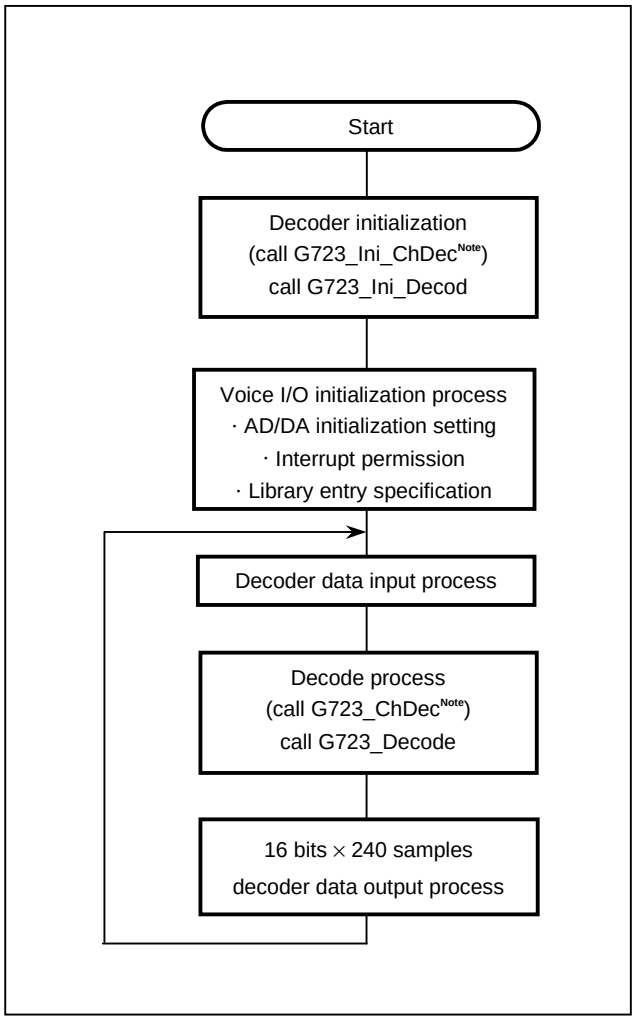
The processing flow of applications that use this middleware is shown in Figures 2-1 and 2-2 below.

Figure 2-1. Application Processing Flow (Encoder)



Note "G723_Ini_ChCod" and "G723_ChCod" are functions for ANNEX C. Be sure to call and use G723_Ini_ChCod and G723_ChCod after G723_Ini_Coder and G723_Encode only when ANNEX C is used.

Figure 2-2. Application Processing Flow (Decoder)



Note "G723_Ini_ChDec" and "G723_ChDec" are functions for ANNEX C. Be sure to call and use G723_Ini_ChDec and G723_ChDec before G723_Ini_Decode and G723_Decode only when ANNEX C is used.

2.2 Function Specifications

2.2.1 Encoder initialization functions

The encoder initialization functions set each encoder constant and initialize the coefficient table and the delay buffer.

(1) Scratch area initialization function

[Classification] Scratch area initialization process
[Function name] G723_Start_Codec
[Function outline] Clears the scratch area used by G.723.1 ANNEX A to 0.
[Format] call G723_Start_Codec
[Argument] None
[Return value] None
[Function] Clears the scratch area used by G.723.1 ANNEX A to 0.
[Registers used] R0, DP0, DP4
[Hardware resource]

Maximum stack level:	0
Maximum loop stack level:	1
Maximum number of repeat:	0
Maximum number of cycles:	4000

(2) Encoder for G.723.1 ANNEX A initialization function

[Classification] Encoder initialization process

[Function name] G723_Ini_Coder

[Function outline] Initializes the RAM area and sets the parameters used by G.723.1 ANNEX A encoder.

[Format] call G723_Ini_Coder

[Argument] Top address of the X/Y static variable area

Example
```
r0l= StaticAreaX                        ;
*IOArea+G723_STATIC_X_PTR:x = r0l       ;
r0l= StaticAreaY                        ;
*IOArea+G723_STATIC_Y_PTR:x = r0l       ;
dp0=IOArea                              ;
```

[Return value] None

[Function] Initializes the G.723.1 ANNEX A encoder and sets the parameters.

[Registers used] R0, R6, R7, DP0, DP4

[Hardware resource]

Maximum stack level:	2
Maximum loop stack level:	1
Maximum number of repeat:	256
Maximum number of cycles:	2000

(3) Encoder for G.723.1 ANNEX C initialization function

[Classification] Encoder initialization process

[Function name] G723_Ini_ChCod

[Function outline] Initializes the RAM area and sets the parameters used by G.723.1 ANNEX C encoder.

[Format] call G723_Ini_ChCod

[Argument] Top address of the I/O buffer and each set parameter

Example
```
clr(r0);
/*...*/ *EncLineBuff+12:x = r0h;
/*...*/ *EncLineBuff+13:x = r0h;
/*---*/ dp0 = LibEntry;
r0l = C_StaticAreaX;
/*...*/ *dp0++ = r0l;
r0l = C_StaticAreaY;
/*...*/ *dp0++ = r0l;
r0l = EncLineBuff;
/*...*/ *dp0++ = r0l;    // Input buffer
r0l = EncChanBuff;
/*...*/ *dp0++ = r0l;    // Output buffer
r0l = 1;
/*...*/ *dp0++ = r0l;    // EncodeSW (0:off 1:on)
r0l = 1;
/*...*/ *dp0++ = r0l;    // High/Low select (0:low 1:high)
/*---*/ dp4 = BitrateConfig;
/* */   *dp0++ = r0h;    // number of octets(dummy access)
/* */ r0 = *dp4++;
rep 6;
        /* */dp0++ = r0h         r0 = *dp4++;
/*---*/ dp0 = LibEntry;
```

[Return value] None

[Function] Initializes the G.723.1 ANNEX C encoder and sets the parameters.

[Registers used] R0, DP0, DP1, DP4

[Hardware resource]

Maximum stack level:	0
Maximum loop stack level:	0
Maximum number of repeat:	15
Maximum number of cycles:	34

2.2.2 Decoder initialization functions

The decoder initialization functions set each decoder constant and initialize the coefficient table and the delay buffer.

(1) Decoder for G.723.1 ANNEX A initialization function

[Classification] Decoder initialization process

[Function name] G723_Ini_Decod

[Function outline] Initializes the RAM area and sets the parameters used by G.723.1 ANNEX A decoder.

[Format] call G723_Ini_Decod

[Argument] Top address of the X/Y static variable area

Example
```
r0l= StaticAreaX                        ;
*IOArea+G723_STATIC_X_PTR:x = r0l       ;
r0l= StaticAreaY                        ;
*IOArea+G723_STATIC_Y_PTR:x = r0l       ;
dp0=IOArea                              ;
```

[Return value] None

[Function] Initializes the G.723.1 ANNEX A decoder and sets the parameters.

[Registers used] R0, R1, R6, R7, DP0, DP4, DP5

[Hardware resource]

Maximum stack level:	2
Maximum loop stack level:	1
Maximum number of repeat:	145
Maximum number of cycles:	1000

(2) Decoder for G.723.1 ANNEX C initialization function

[Classification] Decoder initialization process

[Function name] G723_Ini_ChDec

[Function outline] Initializes the RAM area and sets the parameters used by G.723.1 ANNEX C decoder.

[Format] call G723_Ini_ChDec

[Argument] Top address of the I/O buffer and each set parameter

Example
```
clr(r0);
/*---*/ dp0 = LibEntry + G723C_LIBENTRY_DECOFS;
r0l = DecLineBuff;
/*...*/ *dp0++ = r0l;   // Output buffer
r0l = DecChanBuff;
/*...*/ *dp0++ = r0l;   // Input buffer
/*   */ *dp0++ = r0h;   // clear error flag
/*---*/ dp4 = BitrateConfig;
/* */   *dp0++ = r0h;   // reserved
/* */   r0 = *dp4++;
rep 6;
        /* */   *dp0++ = r0h    r0 = *dp4++;
/*---*/ dp0 = LibEntry;
```

[Return value] None

[Function] Initializes the G.723.1 ANNEX C decoder and sets the parameters.

[Registers used] R0, DP0, DP1, DP4

[Hardware resource]

Maximum stack level:	0
Maximum loop stack level:	0
Maximum number of repeat:	15
Maximum number of cycles:	34

2.2.3 Encoder functions

Encoder functions generate the signals that compress the 240 items of sampled speech data that were input into 189, 159, or 32 bits. Note that in ANNEX C this is a variable rate.

(1) Encoder function for G.723.1 ANNEX A

[Classification] Encoder processing section

[Function name] G723_Encode

[Function outline] Compresses the 240 16-bit samples into 189, 159, or 32 bits

[Format] call G723_Encode

[Argument] Top address of the I/O buffer and each set parameter

```
Example  r0l= EncPcmBuff                        ; set IO parameter
         *IOArea+G723_ENC_BUFF_PTR:x = r0l      ;
         r0l= EncLineBuff                       ;
         *IOArea+G723_VOUT_PTR:x= r0l           ;
         *IOArea+G723_USE_VAD:x= ???            ; 0)VAD off, 1)VAD on
         *IOArea+G723_USE_HP:x= ???             ; 0)Hpf off, 1)Hpf on
         *IOArea+G723_WRK_RATE_E:x= ???         ; 0)6.3kbps, 1)5.3kbps
         dp0=IOArea                             ;
```

[Return value] Frame type and number of clipped subframes

```
Example  ??? = *IOArea+G723_E_FRAME_TYPE:x   ; 0)NoTx ,1)Active ,2)SID
         ??? = *IOArea+G723_COUNT_CLIP:x     ; clipped sub-frames 0..3
```

Compressed data: output buffers set by arguments

[Function] Compresses the input data from the codec (240 samples × 16 bits) into 189, 159, or 32 bits.

[Registers used] R0, R1, R2, R3, R4, R5, R6, R7, DP0, DP1, DP2, DP3, DP4, DP5, DP6, DP7, DN0, DN1, DN2, DN3, DN4, DN5, DN6, DN7, DMX, DMY

[Hardware resource]

Maximum stack level:	7
Maximum loop stack level:	4
Maximum number of repeat:	256
Maximum MIPS value:	17.5

(2) Encoder function for G.723.1 ANNEX C

[Classification] Encoder processing section

[Function name] G723_ChCod

[Function outline] Channel codes the data that was compressed by G723_Encode

[Format] call G723_ChCod

[Argument] Top address of the I/O buffer and each set parameter

Example
```
clr(r0);
/*...*/ *EncLineBuff+12:x = r0h;
/*...*/ *EncLineBuff+13:x = r0h;
/*---*/ dp0 = LibEntry;
r0l = C_StaticAreaX;
/*...*/ *dp0++ = r0l;
r0l = C_StaticAreaY;
/*...*/ *dp0++ = r0l;
r0l = EncLineBuff;
/*...*/ *dp0++ = r0l;    // Input buffer
r0l = EncChanBuff;
/*...*/ *dp0++ = r0l;    // Output buffer
r0l = 1;
/*...*/ *dp0++ = r0l;    // EncodeSW (0:off 1:on)
r0l = 1;
/*...*/ *dp0++ = r0l;    // High/Low select (0:low 1:high)
/*---*/ dp4 = BitrateConfig;
/* */   *dp0++ = r0h;    // number of octets(dummy access)
/* */ r0 = *dp4++;
rep 6;
        /* */dp0++ = r0h         r0 = *dp4++;
/*---*/ dp0 = LibEntry;
```

[Return value] Number of octets: r2l

Compressed data: output buffers set by arguments

[Function] Compresses the input data from the codec (240 samples × 16 bits) at a variable rate

[Registers used] R0, R1, R2, R3, R4, R5, R6, R7, DP0, DP1, DP2, DP3, DP4, DP5, DP6, DP7, DN0, DN1, DN2, DN3, DN4, DN5, DN6, DN7, DMX, DMY

[Hardware resource]

Maximum stack level:	3
Maximum loop stack level:	2
Maximum number of repeat:	636
Maximum MIPS value:	1.0

2.2.4 Decoder functions

Decoder functions decompress the signals that were compressed into 189, 159, or 32 bits back into 240 samples × 16 bits speech data.

(1) Decoder function for G.723.1 ANNEX A

[Classification] Decoder processing section

[Function name] G723_Decode

[Function outline] Decompresses the 189, 159, or 32 bits back into 240 16-bit samples

[Format] call G723_Decode

[Argument] Top address of the I/O buffer and each set parameter

```
Example  r0l= DecPcmBuff                    ; set IO parameter
         *IOArea+G723_DEC_BUFF_PTR:x = r0l ;
         r0l= DecLineBuff                   ;
         *IOArea+G723_VINP_PTR:x= r0l       ;
         *IOArea+G723_USE_PF:x= ???         ; 0)Psf off, 1)Psf on
         *IOArea+G723_CRC_RESULT:x = ???    ;CRC_result,0)Normal,else)Err

         dp0=IOArea                         ;
```

[Return value] Frame type, work rate, and number of error frames

```
Example  ???= *IOArea+G723_D_FRAME_TYPE:x  ; 0)NoTx ,1)Active ,2)SID
         ???= *IOArea+G723_WRK_RATE_D:x    ; 0)6.3kbps, 1)5.3kbps
         ???= *IOArea+ERR_FRM_COUNT:x      ; Num of Error frame 0..0x7ff
```

Decompressed data: output buffers set by arguments

[Function] Decompresses the data that was compressed into 189, 159, or 32 bits back into speech data (240 samples × 16 bits)

[Registers used] R0, R1, R2, R3, R4, R5, R6, R7, DP0, DP1, DP2, DP3, DP4, DP5, DP6, DP7, DN0, DN1, DN2, DN3, DN4, DN5, DN6, DN7, DMX

[Hardware resource]

Maximum stack level:	4
Maximum loop stack level:	3
Maximum number of repeat:	240
Maximum MIPS value:	1.5

(2) Decoder function for G.723.1 ANNEX C

[Classification] Decoder process section

[Function name] G723_ChDec

[Function outline] Performs channel decoding, and creates the code that extracts the G723_Decode function

[Format] call G723_ChDec

[Argument] Top address of the I/O buffer and each set parameter

Example
```
clr(r0);
/*---*/ dp0 = LibEntry + G723C_LIBENTRY_DECOFS;
r0l = DecLineBuff;
/*...*/ *dp0++ = r0l;          // Output buffer
r0l = DecChanBuff;
/*...*/ *dp0++ = r0l;          // Input buffer
/*   */ *dp0++ = r0h;          // clear error flag
/*---*/ dp4 = BitrateConfig;
/* */   *dp0++ = r0h;          // reserved
/* */   r0 = *dp4++;
rep 6;
        /* */   *dp0++ = r0h    r0 = *dp4++;
/*---*/ dp0 = LibEntry;
```

[Return value] mode (0: 6.3 K, 1: 5.3 K, 2: SID): r0l
Final address of bit stream + 1: r1l
Decompressed data: output buffers set by arguments

[Function] Decompresses the data that was compressed by a variable rate back into speech data (240 samples × 16 bits)

[Registers used] R0, R1, R2, R3, R4, R5, R6, R7, DP0, DP1, DP2, DP3, DP4, DP5, DP6, DP7, DN0, DN1, DN2, DN3, DN4, DN5, DN6, DN7, DMX

[Hardware resource]

Maximum stack level:	4
Maximum loop stack level:	3
Maximum number of repeat:	15
Maximum MIPS value:	1.5

2.2.5 Version acquisition function

The version acquisition function returns the version of the library.

[Classification]	Version information acquisition	
[Function name]	G723_GetVersion	
[Function outline]	Restores the version of the library	
[Format]	call G723_GetVersion	
[Argument]	None	
[Return value]	R0H	Major version number
	R0L	Minor version number
[Function]	Returns the version number of this middleware with a 32-bit value.	
	Example	If R0 = 0x00’0x0001’0x0100
		then Version: V1.01
[Registers used]	R0	

2.3 Description of External Interface

The parameters that extract functions G723_Encode and G723_Decode for G.723.1 ANNEX A, and functions G723_ChCod and G723_ChDec for G.723.1 ANNEX C at the time that each function is called are described here.

★ 2.3.1 Extraction parameters for G.723.1 ANNEX A functions

Secure scratch areas as follows. Scratch areas X and Y require "align at 0x20".

- Scratch area X: 1024-word area labeled as "lib_Scratch _x" in the X memory
- Scratch area Y: 1024-word area labeled as "lib_Scratch _y" in the Y memory

Secure static areas as follows. Static area X requires "align at 0x20" and static area Y requires "align at 0x200".

- Make the address of "lib_Scratch _x" the same as that of "lib_Scratch _y".

Table 2-1. Extraction Parameters for G.723.1 ANNEX A Functions

Function	Classification	Symbol[Note 6] (Offset Address)	Setting Value
Encoder/Decoder Shared Parameters			
Static X memory area	Control input[Note 1]	G723_STATIC_X_PTR	X memory addresses
Static Y memory area	Control input[Note 1]	G723_STATIC_Y_PTR	Y memory addresses
Encoder Parameters			
Encoder input buffer	Control input	G723_ENC_BUFF_PTR	X memory addresses
Encoder output buffer	Control input	G723_VOUT_PTR	X memory addresses
Coding bit rate	Control input	G723_WRK_RATE_E	0: 6.3 Kbps, 1: 5.3 Kbps
VAD control	Control input[Note 2]	G723_USE_VAD	0: VAD OFF, 1: VAD ON
Preprocessing HPF control	Control input[Note 2]	G723_USE_HP	0: HPF OFF, 1: HPF ON
Encode frame type	Status output	G723_E_FRAME_TYPE	0: No transfer, 1: Active, 2: SID
Number of clipped frames	Status output	G723_COUNT_CLIP	0 to 3
Decoder Parameters			
Decoder output buffer	Control input	G723_DEC_BUFF_PTR	X memory addresses
Decoder input buffer	Control input	G723_VINP_PTR	X memory addresses
Postprocessing filter control	Control input[Note 3]	G723_USE_PF	0: Filter OFF, 1: Filter ON
CRC flag	Control input	G723_CRC_RESULT	0: Normal, else: Error
Decode frame type	Status output	G723_D_FRAME_TYPE	0: No transfer, 1: Active, 2: SID
Decode bit rate	Status output[Note 4]	G723_WRK_RATE_D	0: 6.3 Kbps, 1: 5.3 Kbps
Frame error count	Status output[Note 5]	G723_ERR_FRM_COUNT	0 to 0x7FFF

Notes 1. Cannot be changed following G723_Start_Codec.

2. Cannot be changed during encoding.

3. Cannot be changed during decoding.

4. Valid only when there are active frames.

5. Reset to 0 if normal frame acknowledged.

6. Symbols show the value offset from the value of DP0 when each function is called.

★2.3.2 Extraction parameters for G.723.1 ANNEX C functions

Secure scratch areas as follows. Scratch areas X and Y require "align at 0x20".

- Scratch area X: 1108-word area labeled as "lib_Scratch _x" in the X memory
- Scratch area Y: 1024-word area labeled as "lib_Scratch _y" in the Y memory

Secure a scratch area consistent with whichever is larger: ANNEX A or ANNEX C (x indicates ANNEX C and y indicates ANNEX A).

Table 2-2. Extraction Parameters for G.723.1 ANNEX C Functions (1/2)

Function	Classification	Symbol[Note 4] (Offset Address)	Setting Value
Encoder/Decoder Shared Parameters			
Static X memory area	Control input[Note 1]	G723C_LibStatic_X	X memory addresses
Static Y memory area	Control input[Note 1]	G723C_LibStatic_Y	Y memory addresses
Encoder Parameters			
Encoder input buffer	Control input	G723C_LibEncInBuffer	X memory addresses
Encoder output buffer	Control input	G723C_LibEncOutBuffer	X memory addresses
Coding drive control	Control input	G723C_LibEncoderSW	0: OFF, 1: ON
High-level/low-level switch	Control input[Note 2]	G723C_LibEncHighLow	0: Low-level, 1: High-level
Coding result data size	Control input	G723C_LibEncOutSize	Number of bits
Channel upper rate when coding at 6.3 Kbps	Control input[Note 3]	G723C_LibEncULim63	Number of bps
Channel lower rate when coding at 6.3 Kbps	Control input[Note 3]	G723C_LibEncLLim63	Number of bps
Channel upper rate when coding at 5.3 Kbps	Control input[Note 3]	G723C_LibEncULim53	Number of bps
Channel lower rate when coding at 5.3 Kbps	Control input[Note 3]	G723C_LibEncLLim53	Number of bps
Channel upper rate when coding at SID	Control input[Note 3]	G723C_LibEncULimSID	Number of bps
Channel lower rate when coding at SID	Control input[Note 3]	G723C_LibEncLLimSID	Number of bps

Notes 1. Cannot be changed following G723_Start_Codec.

2. The high-level/low-level switch is common to 6.3 Kbps, 5.3 Kbps, and SID.

3. The values set for the upper and lower channel rate limits must be within the following range when encoding/decoding:
For 6.3 Kbps: Above 7034 and below 20634
For 5.3 Kbps: Above 6000 and below 17534
For SID: Above 1600 and below 4334

4. Symbols show the value offset from the value of DP0 when each function is called.

Table 2-2. Extraction Parameters for G.723.1 ANNEX C Functions (2/2)

Function	Classification	Symbol[Note 3] (Offset Address)	Setting Value
Decoder Parameters			
Decoder output buffer	Control input	G723C_LibDecOutBuffer	X memory addresses
Decoder input buffer	Control input	G723C_LibDecInBuffer	X memory addresses
Decode result error display	Status output[Note 1]	G723C_LibDecErrorInfo	Bit 2: FII, Bit 1: EFI, Bit 0: BFI
Reserved	–	G723C_LibDecReserved	–
Channel upper rate when decoding at 6.3 Kbps	Control input[Note 2]	G723C_LibDecULim63	Number of bps
Channel lower rate when decoding at 6.3 Kbps	Control input[Note 2]	G723C_LibDecLLim63	Number of bps
Channel upper rate when decoding at 5.3 Kbps	Control input[Note 2]	G723C_LibDecULim53	Number of bps
Channel lower rate when decoding at 5.3 Kbps	Control input[Note 2]	G723C_LibDecLLim53	Number of bps
Channel upper rate when decoding at SID	Control input[Note 2]	G723C_LibDecULimSID	Number of bps
Channel lower rate when decoding at SID	Control input[Note 2]	G723C_LibDecLLimSID	Number of bps

Notes 1. When decoding, input the value from the decode result error display as the CRC value of the code data for decoding. The decoding processing of the frames that correspond to this error display is the frame compensation process.

2. The values set for the upper and lower channel rate limits must be within the following range when encoding/decoding:

For 6.3 Kbps: Above 7034 and below 20634
For 5.3 Kbps: Above 6000 and below 17534
For SID: Above 1600 and below 4334

3. Symbols show the value offset from the value of DP0 when each function is called.

2.4 I/O Data Format

The I/O data format of the ITU-T-recommended G.723.1 is outlined here.

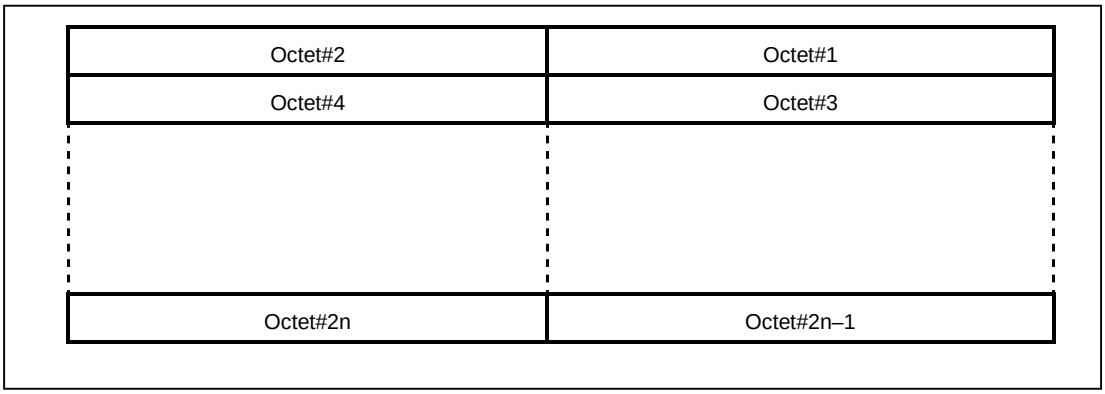
2.4.1 Encoder input/decoder output data format

Input the encoder-input data, which is PCM data that has been sampled at 8 kHz, in a 16-bit × 240-sample format. The decoder output data is output in the form of 16-bit × 240-sample PCM data that has been sampled at 8 kHz.

2.4.2 Encoder output/decoder input data format

The format of compressed data for encoder output and decoder input is as follows.

Figure 2-3. Compressed Data Frame Format



Remark 2n: Frame byte number.

Table 2-3. Bit Rate and Frame Byte Number of G.723.1 ANNEX A Compressed Data

Bit Rate	Frame Byte Number
6.3 Kbps	24
5.3 Kbps	20
SID	4

Table 2-4. Bit Rate and Frame Byte Number of G.723.1 ANNEX C Compressed Data

Bit Rate	Frame Byte Number
6.3 Kbps	27 to 78
5.3 Kbps	23 to 66
SID	6 to 17

(1) Compression format of active frame data

The formats of the 6.3 Kbps speech compression data (189-bit data) and the 5.3 Kbps speech compression data (159-bit data), which are the encoder function output and decoder function input respectively, are shown in Tables 2-5 and 2-6 below. Refer to ITU-T Recommendations for compression data details.

Table 2-5. 6.3 Kbps Speech Compression Data Format

TRANSMITTED OCTETS	PARx_By, …, …
1	LPC_B5 to LPC_B0, VADFLAG_B0, RATEFLAG_B0
2	LPC_B13 to LPC_B6
3	LPC_B21 to LPC_B14
4	ACL0_B5 to ACL0_B0, LPC_B23, LPC_B22
5	ACL2_B4 to ACL2_B0, ACL1_B1, ACL1_B0, ACL0_B6
6	GAIN0_B3 to GAIN0_B0, ACL3_B1, ACL3_B0, ACL2_B6, ACL2_B5
7	GAIN0_B11 to GAIN0_B4
8	GAIN1_B7 to GAIN1_B0
9	GAIN2_B3 to GAIN2_B0, GAIN1_B11 to GAIN1_B8
10	GAIN2_B11 to GAIN2_B4
11	GAIN3_B7 to GAIN3_B0
12	GRID3_B0, GRID2_B0, GRID1_B0, GRID0_B0, GAIN3_B11 to GAIN3_B8
13	MSBPOS_B6 to MSBPOS_B0, UB [Note]
14	POS0_B1, POS0_B0, MSBPOS_B12 to MSBPOS_B7
15	POS0_B9 to POS0_B2
16	POS1_B2, POS1_B0, POS0_B15 to POS0_B10
17	POS1_B10, POS1_B3
18	POS2_B3 to POS2_B0, POS1_B13 to POS1_B11
19	POS2_B11 to POS2_B4
20	POS3_B3 to POS3_B0, POS2_B15 to POS2_B12
21	POS3_B11 to POS3_B4
22	PSIG0_B5 to PSIG0_B0, POS3_B13, POS3_B12
23	PSIG2_B2 to PSIG2_B0, PSIG1_B4 to PSIG1_B0
24	PSIG3_B4 to PSIG3_B0, PSIG2_B5 to PSIG2_B3

Note UB designates an unused bit (value 0).

Table 2-6. 5.3 Kbps Speech Compression Data Format

TRANSMITTED OCTETS	PARx_By, …, …
1	LPC_B5 to LPC_B0, VADFLAG_B0, RATEFLAG_B0
2	LPC_B13 to LPC_B6
3	LPC_B21 to LPC_B14
4	ACL0_B5 to ACL0_B0, LPC_B23, LPC_B22
5	ACL2_B4 to ACL2_B0, ACL1_B1, ACL1_B0, ACL0_B6
6	GAIN0_B3 to GAIN0_B0, ACL3_B1, ACL3_B0, ACL2_B6, ACL2_B5
7	GAIN0_B11 to GAIN0_B4
8	GAIN1_B7 to GAIN1_B0
9	GAIN2_B3 to GAIN2_B0, GAIN1_B11 to GAIN1_B8
10	GAIN2_B11 to GAIN2_B4
11	GAIN3_B7 to GAIN3_B0
12	GRID3_B0, GRID2_B0, GRID1_B0, GRID0_B0, GAIN3_B11 to GAIN3_B8
13	POS0_B7 to POS0_B0
14	POS1_B3 to POS1_B0, POS0_B11 to POS0_B8
15	POS1_B11 to POS1_B4
16	POS2_B7 to POS2_B0
17	POS3_B3 to POS3_B0, POS2_B11 to POS2_B8
18	POS3_B11 to POS3_B4
19	PSIG1_B3 to PSIG1_B0, PSG0_B3 to PSG0_B0
20	PSIG3_B3 to PSIG3_B0, PSIG2_B3 to PSIG2_B0

(2) SID frame compression format

The format of the 32-bit inactive frame data, which is the encoder function output and decoder function input, is shown in Table 2-7 below. Refer to ITU-T Recommendations for compressed data details.

★ Note, however, that "0x3" is returned to "TRANSMITTED OCTETS = 1" and "0x0" to "TRANSMITTED OCTETS = 2 to 4" when G723_E_FRAME_TYPE = 0 or G723_D_FRAME_TYPE = 0.

Table 2-7. Bit Allocation of SID Data

TRANSMITTED OCTETS	PARx_By, …
1	LPC_B5 to LPC_B0, VADFLAG_B0, RATEFLAG_B0
2	LPC_B13 to LPC_B6
3	LPC_B21 to LPC_B14
4	GAIN_B5 to GAIN_B0, LPC_B23, LPC_B22

(3) G.723.1 ANNEX C (channel codec) additional compression data format

The format of ANNEX C data, which is the encoder function output and decoder function input, is shown in Table 2-8 below. Refer to ITU-T Recommendations for compressed data details.

Table 2-8. Bit Allocation of ANNEX C Compression Data

TRANSMITTED OCTETS	Channel Bit
1	UCB[7], UCB[6], UCB[5], UCB[4] , UCB[3], UCB[2], UCB[1], UCB[0]
2	U[2], U[1], U[0], UCB[12], UCB[11], UCB[10], UCB[9] ,UCB[8]
3	U[10], U[9], U[8], U[7], U[6], U[5], U[4], U[3]
Mp/8 + 2	…, …, U[Mp], U[Mp–1], U[Mp–2], …, …, …
:	:
$M_{All}/8 + 2$	UB, UB ,UB, $U[M_{All}–1]$, $U[M_{All}–2]$, $U[M_{All}–3]$, $U[M_{All}–4]$, $U[M_{All}–5]$

Remark UB designates an unused bit (value 0).

The bit rate of a protected bit string is $(M_{All} + 13) \times 1000/30$ bps.

[MEMO]

CHAPTER 3 INSTALLATION

3.1 Installation Procedure

This middleware is only offered in a 3.5-inch (1.44 MB) floppy disk format. The procedure for installing this disk in the host machine is as follows.

(1) Set this disk in the floppy disk drive. Copy the file under a directory that uses software tools (C:\DSPTools, for example). An example of when the file was copied from A drive to C drive is shown below.

```
a:\>xcopy /s *.* c:\DSPTools <CR>
```

(2) Check that the file has been copied. Refer to **1.5.4 Directory configuration** for details of each directory.

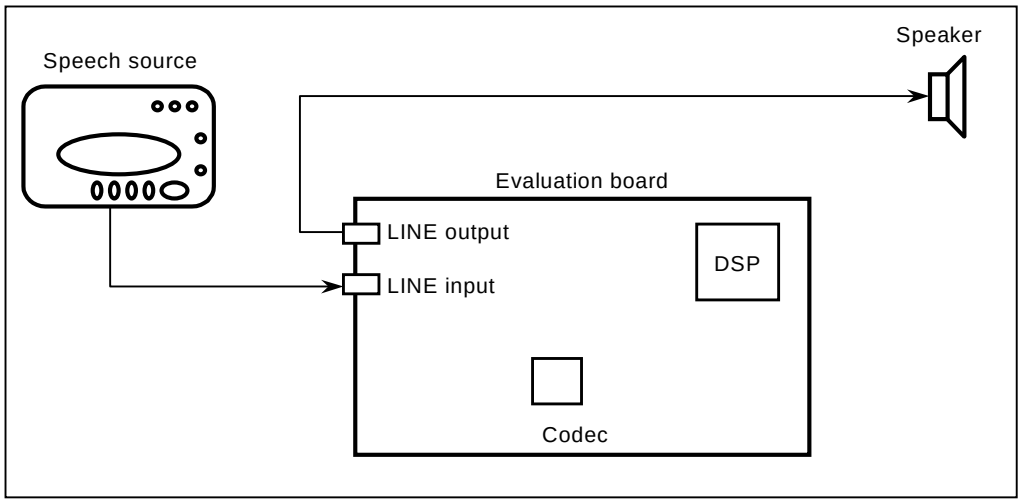
```
a:\>dir c:\DSPTools <CR>
```

3.2 Sample Creation Procedure

The sample program is installed in the sample directory (refer to **APPENDIX SAMPLE PROGRAM SOURCE** for details of the sampleA.asm and sampleAC.asm source programs).

The sample program is able to connect speech sources such as CDs and DATs with speakers, and compress/decompress the speech source in real time.

Figure 3-1. Sample Program Evaluation System



The following is an example of how to build a sample program for this middleware.

(1) Activate the WB77016 (workbench).

(2) Open the sampleA.prj (sampleAC.prj) project file.
Example Specify sampleA.prj (sampleAC.prj) with the Open Project command from the Project menu.

(3) Execute the build function and check that sampleA.lnk (sampleAC.lnk) has been created.
Example When the Build All command is selected from the Make menu, the sampleA.lnk (sampleAC.lnk) file is created.

(4) Download the program to the evaluation board.

3.3 Symbol Naming Conventions

The symbols, etc., in this library are allocated names in accordance with the conventions outlined below. Be careful not to duplicate these names when this middleware is being used in combination with other applications.

Table 3-1. Symbol Names

Classification	Convention	
	G.723.1 ANNEX A	G.723.1 ANNEX C
Function name	G723_×××	G723_Ch×××
Macro, constant name	G723_×××	
Section name	__G723_×××	__G723_C×××

APPENDIX SAMPLE PROGRAM SOURCE

A.1 For G.723.1 + ANNEX A (sampleA.asm)

```
/*------------------------------------------------------------------------- */
/* File Information                                                         */
/*------------------------------------------------------------------------- */
/* Name      : sampleA.asm                                                  */
/* Type      : ASM Programming Language source code                         */
/* Version   : V1.00                                                        */
/* Date      : 1998 June 10                                                 */
/* CPU       : uPD7701x Family                                              */
/* Assembler : WB77016 Ver2.21                                              */
/* About     : NEC uPD7701x Family G723.1 Speech CODEC Middle-Ware Library  */
/*             Sample program [1ch, OFF-LINE mode]                          */
/*------------------------------------------------------------------------- */
/* Copyright (c) NEC Corporation 1995,1996,1997,1998                        */
/* NEC CONFIDENTIAL AND PROPRIETAR                                          */
/* All rights reserved by NEC Coporation                                    */
/* Use of copyright notice does not evidence publication                    */
/*------------------------------------------------------------------------- */

/*---- include files ----*/
#include "uPD77016.h"
#include "g723_lib.h"

/*---- Global Variables ----*/
public   lib_Scratch_x                  /* Scrach area X */
public   lib_Scratch_y                  /* Scrach area Y */

/*---- Global Functions ----*/
extrn    G723_GetVersion                /* Get Version function */
extrn    G723_Start_Codec               /* Start G723 functions */
extrn    G723_Ini_Coder                 /* Initialize G723 Coder variables */
extrn    G723_Ini_Decod                 /* Initialize G723 Decoder variables */
extrn    G723_Encode                    /* Encode 1 Frame */
extrn    G723_Decode                    /* Decode 1 Frame */

$EJECT
/**/
/*************************************************************************/
/* Vector Table                                                          */
/*************************************************************************/
Vct_Ix   imseg at 0x0200
         call Init_Proc                                  ; reset
         jmp  G723_Proc                                  ;
         nop                                             ;
         nop                                             ;
         reti                                            ; not used
         nop                                             ;
         nop                                             ;
         nop                                             ;
         reti                                            ; not used
         nop                                             ;
```

```
nop                                     ;
nop                                     ;
reti                                    ; not used
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; INT0
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; INT1
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; INT2
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; INT3
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; SI#1
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; SO#1
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; SI#2
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; SO#2
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; HOST IN
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; HOST OUT
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; not used
nop                                     ;
nop                                     ;
nop                                     ;
reti                                    ; not used
nop                                     ;
nop                                     ;
nop                                     ;
```

```
/*****************************************************************************/
/* G723 OFF-Line Process                                                     */
/*****************************************************************************/
Smp_Ix   imseg
Init_Proc:
         clr(r0)                                          ; No Wait
         *DWTR:x = r0l                                    ;
         *IWTR:x = r0l                                    ;

         r0l = HST_WAIT                                   ; Wait mode
         *HST:x = r0l                                     ;
         ret                                              ;

$EJECT
/**/
/*--------------------------------------------------------------------------- */
/* Scratch_Area / Static_Area / IO_Buffer                                     */
/*         !!! 'lib_Scratch_x' must be equal to 'lib_Scratch_y' !!!           */
/*--------------------------------------------------------------------------- */
Scratch_X  xramseg align at 0x020
lib_Scratch_x:
         ds        G723_SCRATCH_X_BUFSIZE

Scratch_Y  yramseg align at 0x020
lib_Scratch_y:
         ds        G723_SCRATCH_Y_BUFSIZE

Static_X   xramseg align at 0x020
StaticAreaX:
         ds        G723_STATIC_X_BUFSIZE

Static_Y   yramseg align at 0x200
StaticAreaY:
         ds        G723_STATIC_Y_BUFSIZE

IO_Area_X  xramseg
IOArea:
         ds        G723_IOTABLE_SIZE

IO_Buff_X  xramseg
EncPcmBuff:
         ds        G723_FRAME
DecPcmBuff:
         ds        G723_FRAME
EncLineBuff:
         ds        24/2
DecLineBuff:
         ds        24/2

/*--------------------------------------------------------------------------- */
/* Variables                                                                  */
/*--------------------------------------------------------------------------- */
Smp_Xe1  xramseg
WrkMode:
         ds        1
```

```
/*************************************************************************/
/* G723 OFF-Line Process                                                 */
/*************************************************************************/
Smp_Ix   imseg
G723_Proc:
         call G723_Start_Codec                          ; Need only for Simulation test

         r0l = *HDT:x                                   ; WrkMode 1)Cod ,2)Dec
         *WrkMode:x = r0l                               ;

;;       call G723_GetVersion                           ; Library Version
;;       *HDT:x = r0h                                   ; comment_out for
;;       *HDT:x = r0l                                   ;          Simulator Test

         r0l = StaticAreaX                              ;
         *IOArea+G723_STATIC_X_PTR:x = r0l              ; Static Area X
         r0l = StaticAreaY                              ;
         *IOArea+G723_STATIC_Y_PTR:x = r0l              ; Static Area Y

         r0l = EncPcmBuff                               ;
         *IOArea+G723_ENC_BUFF_PTR:x = r0l              ; PCM Buffer for Encode
         r0l = EncLineBuff                              ;
         *IOArea+G723_VOUT_PTR:x= r0l                   ; Code Buffer for Encode
         r0l = *HDT:x                                   ;
         *IOArea+G723_USE_HP:x= r0l                     ; UseHp  0)No , 1)UseHp
         r0l = *HDT:x                                   ;
         *IOArea+G723_USE_VAD:x= r0l                    ; UseVAD 0)No , 1)UseHp

         r0l = DecPcmBuff                               ;
         *IOArea+G723_DEC_BUFF_PTR:x = r0l              ; PCM buffer for Decode
         r0l = DecLineBuff                              ;
         *IOArea+G723_VINP_PTR:x = r0l                  ; Code buffer for Decode
         r0l = *HDT:x                                   ;
         *IOArea+G723_USE_PF:x= r0l                     ; UsePf  0)No , 1)UsePf

         r0l = *WrkMode:x                               ;
         r0 = r0 & 0x0001                               ;
         if( r0==0 ) jmp  INI_LBC_DEC                   ;
                /*---*/      dp0 = IOArea              ;
                call G723_Ini_Coder                     ;
                jmp  LBC_LOOP                           ;

         INI_LBC_DEC:
                /*---*/      dp0 = IOArea              ;
                call G723_Ini_Decod                     ;

         LBC_LOOP:
                r0 = *HDT:x                             ; Control 0)Stop, 1)Continue
                if( r0==0 ) jmp  LBC_END                ;

                r0l = *WrkMode:x                        ;
                r0 = r0 & 0x0001                        ;
                if( r0==0 ) jmp  LBC_DEC                ;
                         call GetPcmData                ;
                         r0l = *HDT:x                   ;
                         *IOArea+G723_WRK_RATE_E:x = r0l ; WrkRate 0)6.3, 1)5.3 [kbps]
```

```
                        /*---*/     dp0 = IOArea            ;
                        call G723_Encode                    ;

                        call PutLineData                    ;
;;                      r0l = *IOArea+G723_E_FRAME_TYPE:x   ; comment_out for
;;                      *HDT:x = r0l                        ;        Simulator Test
                        jmp  LBC_NEXT                       ;

        LBC_DEC:
                r0l = *WrkMode:x                            ;
                r0 = r0 & 0x0002                            ;
                if( r0==0 ) jmp  LBC_NEXT                   ;
                        call GetLineData                    ;
                        r0l = *HDT:x                        ;
                        *IOArea+G723_CRC_RESULT:x = r0l     ; CRC_result 0)Normal, else)Err

                        /*---*/     dp0 = IOArea            ;
                        call G723_Decode                    ;

                        call PutPcmData                     ;
;;                      r0l = *IOArea+G723_D_FRAME_TYPE:x   ; comment_out for
;;                      *HDT:x = r0l                        ;        Simulator Test
;;                      r0l = *IOArea+G723_WRK_RATE_D:x     ;
;;                      *HDT:x = r0l                        ;
;;                      r0l = *IOArea+G723_ERR_FRM_COUNT:x  ;
;;                      *HDT:x = r0l                        ;

        LBC_NEXT:
                jmp  LBC_LOOP                               ;

  LBC_END:
        jmp  $                                              ;

$EJECT
/**/
/*************************************************************************/
/* Input PCM data for Encode process                                     */
/*************************************************************************/
Smp_Ix   imseg
GetPcmData:
        /*---*/     dp0 = EncPcmBuff                        ;
        loop G723_FRAME {                                   ;
                r0l = *HDT:x                                ;
                /*    */     *dp0++ = r0l                   ;
        }
        ret                                                 ;

/*************************************************************************/
/* Output decoded PCM data                                               */
/*************************************************************************/
Smp_Ix   imseg
PutPcmData:
        /*---*/     dp0 = DecPcmBuff                        ;
        loop G723_FRAME {                                   ;
                /*    */     r0l = *dp0++                   ;
```

```
                *HDT: = r0l                                     ;
        }
        ret                                                     ;

/************************************************************************/
/* Input Code data for Decode process                                   */
/************************************************************************/
Smp_Ix   imseg
GetLineData:
        /*---*/     dp0 = DecLineBuff                           ;
        loop 24/2 {                                             ;
                r0l = *HDT:x                                    ;
                /*    */     *dp0++ = r0l                       ;
        }
        ret                                                     ;

/************************************************************************/
/* Output Encoded Code data                                             */
/************************************************************************/
Smp_Ix   imseg
PutLineData:
        /*---*/     dp0 = EncLineBuff                           ;
        loop 24/2 {                                             ;
                /*    */     r0l = *dp0++                       ;
                *HDT:x = r0l                                    ;
        }
        ret                                                     ;
end
```

A.2 For G.723.1 + ANNEX A + ANNEX C (sampleAC.asm)

```
/*------------------------------------------------------------------------- */
/* File Information                                                         */
/*------------------------------------------------------------------------- */
/* Name      : sampleAC.asm                                                 */
/* Type      : ASM Programming Language source code                         */
/* Version   : V1.00                                                        */
/* Date      : 1998 Oct 10                                                  */
/* CPU       : uPD7701x Family                                              */
/* Assembler : WB77016 Ver2.21                                              */
/* About     : NEC uPD7701x Family G723.1 Speech CODEC Middle-Ware Library  */
/*             Sample program [1ch, OFF-LINE mode]                          */
/*------------------------------------------------------------------------- */
/* Copyright (c) NEC Corporation 1995,1996,1997,1998                        */
/* NEC CONFIDENTIAL AND PROPRIETARY                                         */
/* All rights reserved by NEC Coporation                                    */
/* Use of copyright notice does not evidence publication                    */
/*------------------------------------------------------------------------- */

/*---- include files ----*/
#include  "uPD77016.h"
#include  "g723_lib.h"
#include  "g723clib.h"

/*---- Global Variables ----*/
public    lib_Scratch_x                 /* Scrach area X */
public    lib_Scratch_y                 /* Scrach area Y */

/*---- Global Functions ----*/
extrn     G723_GetVersion               /* Get Version function */
extrn     G723_Start_Codec              /* Start G723 functions */
extrn     G723_Ini_Coder                /* Initialize G723 Coder variables*/
extrn     G723_Ini_Decod                /* Initialize G723 Decoder variables */
extrn     G723_Encode                   /* Encode 1 Frame*/
extrn     G723_Decode                   /* Decode 1 Frame*/

$EJECT
/**/
/*************************************************************************/
/* Vector Table                                                          */
/*************************************************************************/
Vct_Ix   imseg at 0x0200
         call Init_Proc                                  ; reset
         jmp  G723_Proc                                  ;
         nop                                             ;
         nop                                             ;
         reti                                            ; not used
         nop                                             ;
         nop                                             ;
         nop                                             ;
         reti                                            ; not used
         nop                                             ;
         nop                                             ;
         nop                                             ;
         reti                                            ; not used
```

```
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; INT0
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; INT1
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; INT2
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; INT3
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; SI#1
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; SO#1
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; SI#2
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; SO#2
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; HOST IN
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; HOST OUT
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; not used
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;
        reti                                                    ; not used
        nop                                                     ;
        nop                                                     ;
        nop                                                     ;

/*************************************************************************/
/* G723 OFF-Line Process                                                 */
/*************************************************************************/
Smp_Ix   imseg
```

```
Init_Proc:
        clr(r0)                                          ; No Wait
        *DWTR:x = r0l                                    ;
        *IWTR:x = r0l                                    ;

        r0l = HST_WAIT                                   ; Wait mode
        *HST:x = r0l                                     ;
        ret                                              ;

$EJECT
/**/
/*------------------------------------------------------------------------- */
/* Scratch_Area / Static_Area / IO_Buffer                                   */
/*       !!! 'lib_Scratch_x' must be equal to 'lib_Scratch_y' !!!           */
/*------------------------------------------------------------------------- */
Scratch_X  xramseg align at 0x020
lib_Scratch_x:
;;      ds      G723_SCRATCH_X_BUFSIZE
        ds      G723C_SCRATCH_X_BUFSIZE                 ; is bigger than G723_SCRATCH_X_BUFSIZE

Scratch_Y  yramseg align at 0x020
lib_Scratch_y:
        ds      G723_SCRATCH_Y_BUFSIZE

Static_X   xramseg align at 0x020
StaticAreaX:    ds      G723_STATIC_X_BUFSIZE
C_StaticAreaX:  ds      G723C_STATIC_X_BUFSIZE          ; Annex C

Static_Y   yramseg align at 0x200
StaticAreaY:    ds      G723_STATIC_Y_BUFSIZE
C_StaticAreaY:  ds      G723C_STATIC_Y_BUFSIZE          ; Annex C

IO_Area_X  xramseg
IOArea:
        ds      G723_IOTABLE_SIZE
LibEntry:
        ds      G723C_LIBENTRY_SIZE                     ; Annex C

IO_Buff_X  xramseg
EncPcmBuff:
        ds      G723_FRAME
DecPcmBuff:
        ds      G723_FRAME
EncLineBuff:
        ds      24/2 + 2
DecLineBuff:
        ds      24/2 + 1

EncChanBuff:    ds      48                              ; Annex C
DecChanBuff:    ds      48                              ; Annex C

/*------------------------------------------------------------------------- */
/* Variables                                                                */
/*------------------------------------------------------------------------- */
Smp_Xe1  xramseg
WrkMode:
```

```
        ds     1

/*-------------------------------------------------------------*/
/*           Annex C initialization                            */
/*-------------------------------------------------------------*/
AnnexC_Const yramseg
BitrateConfig:
        dw        77, 27, 65,  23, 16, 6                  /* Channel rate limits */

InitC imseg
InitAnnexC:
        clr(r0);
        /*...*/      *EncLineBuff+12:x = r0h;
        /*...*/      *EncLineBuff+13:x = r0h;

        /*---*/      dp0 = LibEntry;
        r0l = C_StaticAreaX;
        /*...*/      *dp0++ = r0l;
        r0l = C_StaticAreaY;
        /*...*/      *dp0++ = r0l;
        r0l = EncLineBuff;
        /*...*/      *dp0++ = r0l;                         // Input buffer
        r0l = EncChanBuff;
        /*...*/      *dp0++ = r0l;                         // Output buffer
        r0l = 1;
        /*...*/      *dp0++ = r0l;                         // EncodeSW (0:off 1:on)
        r0l = 1;
        /*...*/      *dp0++ = r0l;                         // High/Low select (0:low 1:high)
        /*---*/      dp4 = BitrateConfig;
        /*     */    *dp0++ = r0h;                         // number of octets(dummy access)
        /*     */    r0 = *dp4++;
        rep 6;
              /*      */    *dp0++ = r0h      r0 = *dp4++;

        /*---*/      dp0 = LibEntry;
        call G723_Ini_ChCod;

        clr(r0);
        /*---*/      dp0 = LibEntry + G723C_LIBENTRY_DECOFS;
        r0l = DecLineBuff;
        /*...*/      *dp0++ = r0l;                         // Output buffer
        r0l = DecChanBuff;
        /*...*/      *dp0++ = r0l;                         // Input buffer
        /*   */      *dp0++ = r0h;                         // clear error flag
        /*---*/      dp4 = BitrateConfig;
        /*     */    *dp0++ = r0h;                         // reserved
        /*     */    r0 = *dp4++;
        rep 6;
              /*      */    *dp0++ = r0h      r0 = *dp4++;

        /*---*/      dp0 = LibEntry;
        call G723_Ini_ChDec;

        ret;
```

```
/*************************************************************************/
/* G723 OFF-Line Process                                                 */
/*************************************************************************/
Smp_Ix   imseg
G723_Proc:
         call G723_Start_Codec                ; Need only for Simulation test

         r0l = *HDT:x                         ; WrkMode 1)Cod ,2)Dec
         *WrkMode:x = r0l                     ;

;;       call G723_GetVersion                 ; Library Version
;;       *HDT:x = r0h                         ; comment_out for
;;       *HDT:x = r0l                         ;        Simulator Test

         r0l = StaticAreaX                    ;
         *IOArea+G723_STATIC_X_PTR:x = r0l    ; Static Area X
         r0l = StaticAreaY                    ;
         *IOArea+G723_STATIC_Y_PTR:x = r0l    ; Static Area Y

         r0l = EncPcmBuff                     ;
         *IOArea+G723_ENC_BUFF_PTR:x = r0l    ; PCM Buffer for Encode
         r0l = EncLineBuff                    ;
         *IOArea+G723_VOUT_PTR:x = r0l        ; Code Buffer for Encode
         r0l = *HDT:x                         ;
         *IOArea+G723_USE_HP:x = r0l          ; UseHp  0)No , 1)UseHp
         r0l = *HDT:x                         ;
         *IOArea+G723_USE_VAD:x= r0l          ; UseVAD 0)No , 1)UseHp

         r0l = DecPcmBuff                     ;
         *IOArea+G723_DEC_BUFF_PTR:x = r0l    ; PCM buffer for Decode
         r0l = DecLineBuff                    ;
         *IOArea+G723_VINP_PTR:x = r0l        ; Code buffer for Decode
         r0l = *HDT:x                         ;
         *IOArea+G723_USE_PF:x = r0l          ; UsePf  0)No , 1)UsePf


;; Setup Lib-Entry for Annex C
         call InitAnnexC;

         r0l = *WrkMode:x                     ;
         r0 = r0 & 0x0001                     ;
         if( r0==0 ) jmp  INI_LBC_DEC         ;
                /*---*/      dp0 = IOArea     ;
                call G723_Ini_Coder           ;
                jmp  LBC_LOOP                 ;

          INI_LBC_DEC:
                /*---*/      dp0 = IOArea     ;
                call G723_Ini_Decod           ;
           LBC_LOOP:
                r0 = *HDT:x                   ; Control 0)Stop, 1)Continue
                if( r0==0 ) jmp  LBC_END      ;

                r0l = *WrkMode:x              ;
                r0 = r0 & 0x0001              ;
                if( r0==0 ) jmp  LBC_DEC      ;
```

```
                    call GetPcmData                    ;
                    r0l = *HDT:x                       ;
                    *IOArea+G723_WRK_RATE_E:x = r0l    ; WrkRate 0)6.3, 1)5.3 [kbps]

                    /*---*/     dp0 = IOArea           ;
                    call G723_Encode                   ;

                    /*---*/     dp0 = LibEntry         ; Annex C
                    call G723_ChCod                    ;

;                   call PutLineData                   ; comment for Annex C
                    call PutChannelData;

;;                  r0l = *IOArea+G723_E_FRAME_TYPE:x  ; comment_out for
;;                  *HDT:x = r0l                       ;         Simulator Test
                    jmp  LBC_NEXT                      ;

          LBC_DEC:
               r0l = *WrkMode:x                        ;
               r0 = r0 & 0x0002                        ;
               if( r0==0 ) jmp  LBC_NEXT               ;
;                     call GetLineData                 ;
                      call GetChannelData              ;

                      /*---*/     dp0 = LibEntry       ; Annex C
                      call G723_ChDec                  ;

;                     r0l = *HDT:x                     ;
                      r0l = *(LibEntry + G723C_LIBENTRY_ERRINFO):x;
                      *IOArea+G723_CRC_RESULT:x = r0l  ; CRC_result 0)Normal, else)Err

                      /*---*/     dp0 = IOArea         ;
                      call G723_Decode                 ;

                      call PutPcmData                  ;
;;                    r0l = *IOArea+G723_D_FRAME_TYPE:x ; comment_out for
;;                    *HDT:x = r0l                     ;       Simulator Test
;;                    r0l = *IOArea+G723_WRK_RATE_D:x  ;
;;                    *HDT:x = r0l                     ;
;;                    r0l = *IOArea+G723_ERR_FRM_COUNT:x ;
;;                    *HDT:x = r0l                     ;

          LBC_NEXT:
               jmp  LBC_LOOP                ;

  LBC_END:
         nop                                ;
         nop                                ;
         nop                                ;
         nop                                ;
         nop                                ;
         nop                                ;
         nop                                ;
         halt                               ;
```

```
$EJECT
/**/
/*************************************************************************/
/* Input PCM data for Encode process                                     */
/*************************************************************************/
Smp_Ix   imseg
GetPcmData:
        /*---*/     p0 = EncPcmBuff           ;
        loop G723_FRAME {                     ;
               r0l = *HDT:x                   ;
               /*    */      *dp0++ = r0l      ;
        }
        ret                                   ;

/*************************************************************************/
/* Output decoded PCM data                                               */
/*************************************************************************/
Smp_Ix   imseg
PutPcmData:
        /*---*/     dp0 = DecPcmBuff          ;
        loop G723_FRAME {                     ;
               /*    */      r0l = *dp0++      ;
               *HDT:x = r0l                   ;
        }
        ret                                   ;

/*************************************************************************/
/* Input Code data for Decode process                                    */
/*************************************************************************/
Smp_Ix       imseg
GetLineData:
        /*---*/     dp0 = DecLineBuff         ;
        loop 24/2 {                           ;
               r0l = *HDT:x                   ;
               /*    */      *dp0++ = r0l      ;
        }
        ret                                   ;

/*************************************************************************/
/* Output Encoded Code data                                              */
/*************************************************************************/
Smp_Ix   imseg
PutLineData:
        /*---*/     dp0 = EncLineBuff         ;
        loop 24/2 {                           ;
               /*    */      r0l = *dp0++      ;
               *HDT:x = r0l                   ;
        }
        ret                                   ;


/*=====================================================================**
 *           Annex C
 *=====================================================================*/
Smp_Ix   imseg
PutChannelData:
```

```
        /*---*/     p0 = EncChanBuff;
        clr(r0);
        /*...*/     r0l = *(LibEntry + G723C_LIBENTRY_OUTSIZE):x;
        r0 = r0 + 1;
        r0 = r0 srl 1;
        loop r0l {
               /*...*/      r0l = *dp0++;
               /*...*/      *HDT:x = r0l;
        }
        ret;

GetChannelData:
        /*---*/     dp0 = DecChanBuff;
        clr(r0);
        r0l = *HDT:x;                    /* r0l = the number of words */
        loop r0l {
               /*...*/      r0l = *HDT:x;
               /*...*/      *dp0++ = r0l;
        }
        ret;

end
```

NEC

Facsimile Message

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

From:

Name

Company

Tel. FAX

Address

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6462-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: 044-435-9608

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑