Embedded SDK
(Software Development Kit)

VAD/CNG/DTX Library

SDK148/D
Rev. 0, 09/06/2002
# Contents

## About This Document

### Chapter 1
**Introduction**

1.1 Quick Start ................................................. 13  
1.2 Overview of VAD/CNG/DTX ................................. 13  
1.2.1 Background .............................................. 14  
1.2.2 Features and Performance ............................... 14

### Chapter 2
**Directory Structure**

2.1 Required Core Directories ................................ 17  
2.2 Optional (Domain-Specific) Directories .................. 18

### Chapter 3
**VAD/CNG/DTX Library Interfaces**

3.1 VAD/CNG/DTX Services ..................................... 21  
3.2 Interface ..................................................... 21  
3.3 Specifications ............................................... 25  
3.3.1 vcdEncoderInit ........................................... 26  
3.3.2 vcdEncoder ................................................ 27  
3.3.3 vcdEncoderDestroy ....................................... 28  
3.3.4 vcdDecoderInit ........................................... 29  
3.3.5 vcdDecoder ............................................... 30  
3.3.6 vcdDecoderDestroy ....................................... 31

### Chapter 4
**Building the VAD/CNG/DTX Library**

4.1 Building the VAD/CNG/DTX Library ......................... 33

### Chapter 5
**Linking Applications with the VAD/CNG/DTX Library**

5.1 Calling VAD/CNG/DTX Library ................................ 35  
5.2 Recommended Memory Map .................................. 36
Chapter 6
VAD/CNG/DTX Applications

6.1 Test and Demo Applications ................................................................. 39

Chapter 7
License

7.1 Limited Use License Agreement .............................................................. 41
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1-1</td>
<td>VAD/CNG/DTX library implementation performance</td>
<td>14</td>
</tr>
<tr>
<td>Table 3-1</td>
<td>vcdEncoderInit Arguments</td>
<td>26</td>
</tr>
<tr>
<td>Table 3-2</td>
<td>vcdEncoder Arguments</td>
<td>27</td>
</tr>
<tr>
<td>Table 3-3</td>
<td>vcdEncoderDestroy Arguments</td>
<td>28</td>
</tr>
<tr>
<td>Table 3-4</td>
<td>vcdDecoderInit Arguments</td>
<td>29</td>
</tr>
<tr>
<td>Table 3-5</td>
<td>vcdDecoder Arguments</td>
<td>30</td>
</tr>
<tr>
<td>Table 3-6</td>
<td>vcdDecoderDestroy Arguments</td>
<td>31</td>
</tr>
<tr>
<td>Table 5-1</td>
<td>Calling Order for vcdEncoder Interfaces</td>
<td>35</td>
</tr>
<tr>
<td>Table 5-2</td>
<td>Calling Order for vcdDecoder Interfaces</td>
<td>35</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2-1  Core Directories ................................................................. 17
Figure 2-2  DSP56858 Directories ....................................................... 18
Figure 2-3  vad_cng_dtx Directory Structure ................................. 19
## List of Examples

<table>
<thead>
<tr>
<th>Code Example</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1</td>
<td>C Header File vad_cng_dtx.h</td>
<td>21</td>
</tr>
<tr>
<td>3-2</td>
<td>Use of vcdEncoderInit Interface</td>
<td>26</td>
</tr>
<tr>
<td>3-3</td>
<td>Use of vcdEncoder Interface</td>
<td>27</td>
</tr>
<tr>
<td>3-4</td>
<td>Use of vcdEncoderDestroy Interface</td>
<td>28</td>
</tr>
<tr>
<td>3-5</td>
<td>Use of vcdDecoderInit Interface</td>
<td>29</td>
</tr>
<tr>
<td>3-6</td>
<td>Use of vcdDecoder Interface</td>
<td>30</td>
</tr>
<tr>
<td>3-7</td>
<td>Use of vcdDecoderDestroy Interface</td>
<td>31</td>
</tr>
<tr>
<td>5-1</td>
<td>linker.cmd file</td>
<td>36</td>
</tr>
</tbody>
</table>
About This Document
This manual describes the VAD/CNG/DTX algorithm for use with Motorola’s Embedded Software Development Kit (SDK).

Audience
This document targets software developers implementing VAD/CNG/DTX functions within software applications.

Organization
This manual is arranged in the following sections:

- **Chapter 1, Introduction**—provides a brief overview of this document
- **Chapter 2, Directory Structure**—provides a description of the required core directories
- **Chapter 3, VAD/CNG/DTX Library Interfaces**—describes all of the VAD/CNG/DTX Library functions
- **Chapter 4, Building the VAD/CNG/DTX Library**—tells how to execute the system library project build
- **Chapter 5, Linking Applications with the VAD/CNG/DTX Library**—describes the organization of the VAD/CNG/DTX Library
- **Chapter 6, VAD/CNG/DTX Applications**—describes the use of VAD/CNG/DTX Library through test/demo applications
- **Chapter 7, License**—provides the license required to use this product

Suggested Reading
We recommend that you have a copy of the following references:

2. *DSP568xx User’s Manual* for the DSP device you’re implementing
Conventions

This document uses the following notational conventions:

<table>
<thead>
<tr>
<th>Typeface, Symbol or Term</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
</table>
| Courier Monospaced Type  | Commands, command parameters, code examples, expressions, data types, and directives | ... *Foundational include files...  
... a data structure of type vad_tConfigure... |
| *Italic*                | Calls, functions, statements, procedures, routines, arguments, file names and applications | ... the pConfig argument...  
... defined in the C header file, aec.h...  
... makes a call to the Callback procedure... |
| **Bold**                | Reference sources, paths, emphasis | ... refer to the Targeting DSP56824 Platform manual....  
... see: C:\Program Files\Motorola\Embedded SDK\help\tutorials |
| **Bold/Italic**         | Directory name, project name | ... and contains these core directories:  
applications contains applications software....  
... CodeWarrior project, 3des.mcp, is..... |
| **Blue Text**           | Linkable on-line | ... refer to Chapter 7, License... |
| Number                  | Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value | 3V  
-10  
DES \(^{-1}\) |
| ALL CAPITAL LETTERS     | Variables, directives, defined constants, files libraries | INCLUDE_DSPFUNC  
#define INCLUDE_STACK_CHECK |
| Brackets [...]          | Function keys | ... by pressing function key [F7]... |
| Quotation marks "... "  | Returned messages | ... the message, "Test Passed" is displayed....  
... if unsuccessful for any reason, it will return "NULL".... |

Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document. As this template develops, this list will be generated from the document. As we develop more group resources, these acronyms will be easily defined from a common acronym dictionary. Please note that while the acronyms are in solid caps, terms in the definition should be initial capped ONLY IF they are trademarked names or proper nouns.

**ACK**  Acknowledge Message

**CL**  Capabilities List

**CLR**  Capabilities List Request

**CNG**  Comfort Noise Generation
CR  Capabilities Request
DCE  Data Circuit-terminating Equipment
DSP  Digital Signal Processor or Digital Signal Processing
DTE  Data Terminal Equipment
DTX  Discontinuous Transmission
ES   Escape Signal
FCS  Frame Check Sequence
I/O  Input/Output
IDE  Integrated Development Environment
LSB  Least Significant Bit
MAC  Multiply/Accumulate
MIPS Million Instructions Per Second
MR   Mode Request
MS   Mode Select
MSB  Most Significant Bit
NAK  Negative Acknowledge Message
OnCE™ On-Chip Emulation
OMR  Operating Mode Register
PC   Personal Computer
PSTN Public Switched Telephone Network
SDK  Software Development Kit
SP   Stack Pointer
SPI  Serial Peripheral Interface
SR   Status Register
SRC  Source
VAD  Voice Activity Detection
VCD  VAD/CNG/DTX

References

The following sources were referenced to produce this book:

Chapter 1
Introduction

Welcome to Motorola’s family of Digital Signal Processors, DSPs. This document describes the VAD/CNG/DTX Library, which is a part of Motorola’s comprehensive Software Development Kit, SDK, for its DSPs. In this document, you will find all the information required to use and maintain the VAD/CNG/DTX Library interface and algorithms.

Motorola provides these algorithms to you for use with Motorola DSPs to expedite your application development and reduce the time it takes to bring your own products to market.

Motorola’s VAD/CNG/DTX Library is licensed for your use on Motorola processors. Please refer to the Software License Agreement in Chapter 7 for license terms and conditions; please consult with your Motorola representative for premium product licensing.

1.1 Quick Start

Motorola’s Embedded SDK is targeted to a large variety of hardware platforms. To take full advantage of a particular hardware platform, use Quick Start from the Targeting DSP568xx Platform documentation.

For example, the Targeting DSP5685x Platform manual provides more specific information and examples about this hardware architecture. If you are developing an application for the DSP56858EVM board, or any other DSP56858 development system, refer to the Targeting DSP5685x Platform manual for Quick Start or other DSP56858-specific information.

1.2 Overview of VAD/CNG/DTX

The name of the library comes from three modules used to reduce the channel payload during portions of non-active speech.

- **VAD** - The Voice Activity Detection module determines if the current speech frame contains active or non-active voice
- **CNG** - The Comfort Noise Generation module substitutes the voice codec during non-active voice frames. A comfort noise is generated during these frames on the decoder side, based on energy level and spectral information transmitted by the encoder.
- **DTX** - The Discontinuous Transmission module detects a significant change in non-active voice characteristics and determines if there is encoded speech to be transmitted to the decoder.
Introduction

The ITU-T Recommendation G.711 - Appendix II describes the payload (bit-stream) format in non-active voice frames. The current VAD/CNG/DTX implementation complies with this ITU-T Recommendation.

1.2.1 Background

VAD/CNG/DTX is based on the software implementation example presented in section II.5 of ITU-T Recommendation G.711 - Appendix II, reusing modules from G.729AB voice codec.

The VAD/CNG/DTX library enhances G.711 and G.726 voice codecs with the capability of low payloads during non-voiced input signal. Other voice codecs (such as G.722, G.727, G.728) can also be enhanced by using this library to reduce the channel payload in non-active voice frames.

The library is basically a wrapper over the G.711 and G.726 libraries; the voice codecs are not part of the VAD/CNG/DTX library. To invoke the VAD/CNG/DTX library in an application, call the entry points of VAD/CNG/DTX instead of the entry points of G.711 or G.726 voice codecs, which are called from inside the library.

The VAD/CNG/DTX is based on G.729AB, so the speech is processed in frames of 10 milliseconds (80 samples at 8KHz), unlike the original G.711 or G.726, which individually process each sample of speech.

1.2.2 Features and Performance

The current version of the VAD/CNG/DTX library enhances G.711 and G.726 voice codecs.

VAD/CNG/DTX reduces the G.711 channel payload at least eight times for non-active voice frames. In discontinuous transmission, the number of bytes transmitted per frame is reduced to zero.

In non-active speech frames, the payload can vary between 1 and 11 bytes per frame, depending on the number of reflection coefficients (RCs) transmitted. The RCs carry spectral information and number between 0 and 10; the adopted linear prediction model order is 10. Sending 10 RCs provides a high quality reconstructed signal, while sending no RCs improves the MIPS performance by about 12%, but decreases output signal quality.

Based on G.729AB, the typical call rate of the encoder/decoder is 10ms (at each 80 samples of speech).

Table 1-1 details performance figures for the implementation of VAD/CNG/DTX library only; G.711 or G.726 vocoders are not included.

Table 1-1. VAD/CNG/DTX library implementation performance

<table>
<thead>
<tr>
<th>Table Heading</th>
<th>Code Size (words)</th>
<th>Tables Size (words)</th>
<th>Channel Status Size (words)</th>
<th>Stack Size (words)</th>
<th>Processing Load (MCPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 RCs</td>
</tr>
<tr>
<td>Encoder</td>
<td>—</td>
<td>—</td>
<td>376</td>
<td>—</td>
<td>2.76</td>
</tr>
<tr>
<td>Decoder</td>
<td>—</td>
<td>—</td>
<td>192</td>
<td>—</td>
<td>1.36</td>
</tr>
<tr>
<td>Voice Codec</td>
<td>3620</td>
<td>770</td>
<td>568</td>
<td>544</td>
<td>4.12</td>
</tr>
</tbody>
</table>
• These performance figures were obtained running on the DSP56858EVM board, using Metrowerks CodeWarrior for DSP56800E Release 1.1 and Motorola Embedded SDK 1.5.1E drivers
Chapter 2
Directory Structure

2.1 Required Core Directories

Figure 2-1 details required platform directories:

As shown in Figure 2-1, DSP56858EVM has no operating system (nos) support and contains these core directories:

- **applications** contains applications software that can be exercised on this platform
- **bsp** contains board support package specific for this platform
- **config** contains default hardware/software configurations for this platform
- **include** contains SDK header files which define the Application Programming Interface
- **sys** contains required system components
- **telephony** contains telephony libraries and library test files
- **tools** contains utilities used by system components

There are also optional directories that include domain-specific libraries.
2.2 Optional (Domain-Specific) Directories

Figure 2-2 demonstrates how the VAD/CNG/DTX is encapsulated in the domain-specific directory telephony.

The `vad_cng_dtx` directory includes the VAD/CNG/DTX specific algorithms. Figure 2-3 shows the `vad_cng_dtx` directory structure.
Figure 2-3. vad_cng_dtx Directory Structure

The vad_cng_dtx directory includes the following sub-directories:

- **asm_sources** includes all asm source files
- **c_sources** includes APIs for the VAD/CNG/DTX
- **test_vad_cng_dtx** includes c source code and configinfram
  - **c_sources** contains an example of test code for the VAD/CNG/DTX
  - **configinfram** contains the configuration files appconfig.c, appconfig.h and linker.cmd specific to VAD/CNG/DTX testing
Chapter 3
VAD/CNG/DTX Library Interfaces

3.1 VAD/CNG/DTX Services

The VAD/CNG/DTX library reduces the transmission rate during silence periods of speech in voice communications, lowering the bit rates for telephony applications. The Voice Activity Detection (VAD) algorithm classifies the input signal into active speech and non-active speech, or ambient noise. The Comfort Noise Generation (CNG) algorithm extracts a sufficient description of the ambient noise to minimize the transmission rate. The Discontinuous Transmission (DTX) algorithm detects the significant changes in ambient noise characteristics. When there are no significant changes, nothing is transmitted to the decoder.

3.2 Interface

The C interface for VAD/CNG/DTX library services is defined in the C header file `vad_cng_dtx.h`, shown in Code Example 3-1.

Code Example 3-1. C Header File `vad_cng_dtx.h`

```c
#include "port.h"
#include "g711.h"
#include "g726.h"
```

```c
#define __VAD_CNG_DTX_H__

#include "port.h"
#include "g711.h"
#include "g726.h"
```
/**=--------------------------------------------------------------------------*
  * Defines
  *============================================================================*/

#ifndef IN
#define IN
#endif

#ifndef OUT
#define OUT
#endif

#ifndef IN_OUT
#define IN_OUT
#endif

/**=--------------------------------------------------------------------------*
  * Constants
  *============================================================================*/

#define VCD_ENCODER_CHANNEL_DATA_SIZE 376
#define VCD_DECODER_CHANNEL_DATA_SIZE 192
#define VCD_FRAME_SIZE 80

/*===========================================================================
  * TypeDefs (Structures, Unions, Enums)
  *---------------------------------------------------------------------------*

typedef struct {
    Word32 Buffer[VCD_ENCODER_CHANNEL_DATA_SIZE/2];
} vcd_sEncStatus;

/*===========================================================================
  * G.711 specific section
  *---------------------------------------------------------------------------*/

#define VCD_G711_A_LAW 1    /* A law used in G.711 encoding */
#define VCD_G711_u_LAW 2    /* u law used in G.711 encoding */

/*===========================================================================
  * G.726 specific section
  *---------------------------------------------------------------------------*/

#define VCD_G726_16 82  /* G.726 at 16 kbits (lowest quality) frame */
#define VCD_G726_24 83  /* G.726 at 24 kbits frame */
#define VCD_G726_32 84  /* G.726 at 16 kbits frame */
#define VCD_G726_40 85  /* G.726 at 16 kbits (highest quality) frame */

*/

/*===========================================================================
  * Voice Activity Detector Library
  *---------------------------------------------------------------------------*

For More Information On This Product,
Go to: www.freescale.com
/**
typedef struct
{
    Word32 Buffer[VCD_DECODER_CHANNEL_DATA_SIZE/2];
} vcd_sDecStatus;

/* Information about the vocoder used, needed to initialize the vocoder */
typedef union
{
    Word16 g711Law;
    G726_Enc_sConfigure g726Params;
} vcd_uEncConfig;

typedef union
{
    Word16 g711Law;
    G726_Dec_sConfigure g726Params;
} vcd_uDecConfig;

/*============================================================================
FUNCTION PROTOTYPES
============================================================================*/

/* Functions needed for encoding */
Word16 vcdEncoderInit(
    IN Word16 EncType,                  /* type of encoder                  */
    IN Word16 nRCs,                     /* number of RCs transmitted        */
    IN vcd_uEncConfig *EncConfig,       /* initialization parameters        */
    IN_OUT vcd_sEncStatus *pEncChData   /* Encoder channel information      */
);

void vcdEncoder(
    IN Word16 *speech,                  /* input speech                      */
    OUT UInt8 *serial,                  /* parameters to be sent to the      */
                           /* channel                           */
    IN_OUT vcd_sEncStatus *pEncChData,  /* Encoder channel information      */
    IN Word16 enable_vad                /* VAD enable flag                   */
);

Word16 vcdEncoderDestroy(
    IN_OUT vcd_sEncStatus *pEncChData   /* Encoder channel information      */
);

/* Functions needed for Decoding */
Word16 vcdDecoderInit(
    IN Word16 DecType,                  /* type of decoder                  */
    IN vcd_uDecConfig *DecConfig,       /* initialization parameters        */
    IN_OUT vcd_sDecStatus *pDecChData   /* Decoder channel information      */
);

void vcdDecoder(

IN Uint8 *serial, /* parameters received from
   channel */
OUT Word16 *synth, /* decoded speech */
IN_OUT vcd_sDecStatus *pDecChData /* Decoder channel information */

Word16 vcdDecoderDestroy(
   IN_OUT vcd_sDecStatus *pDecChData /* Decoder channel status */
);

/*============================================================================
VARIABLES
============================================================================*/

/*============================================================================
FUNCTIONS
============================================================================*/

#endif __VAD_CNG_DTX_H__

#ifdef __cplusplus
}
#endif
3.3 Specifications

The following pages describe the VAD/CNG/DTX library functions.

Function arguments for each routine are described as \textit{in}, \textit{out}, or \textit{inout}. An \textit{in} argument means that the parameter value is an input only to the function. An \textit{out} argument means that the parameter value is an output only from the function. An \textit{inout} argument means that a parameter value is an input to the function, but the same parameter is also an output from the function.

Typically, \textit{inout} parameters are input pointer variables in which the caller passes the address of a preallocated data structure to a function. The function stores its results within that data structure. The actual value of the \textit{inout} pointer parameter is not changed.
### 3.3.1 vcdEncoderInit

**Call(s):**

```c
/* Functions needed for encoding */
Word16 vcdEncoderInit(
    IN Word16 EncType,                  /* type of encoder                  */
    IN Word16 nRCs,                     /* number of RCs transmitted        */
    IN vcd_uEncConfig *EncConfig,       /* initialization parameters        */
    IN_OUT vcd_sEncStatus *pEncChData   /* Encoder channel information      */
);
```

**Required Header:** “vad_cng_dtx.h”

**Arguments:**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EncType</td>
<td>Type of encoder to run</td>
</tr>
<tr>
<td>nRCs</td>
<td>Number of RCs to transmit into the transmission channel</td>
</tr>
<tr>
<td>*EncConfig</td>
<td>Encoder initialization parameters (see Code Example 3-1)</td>
</tr>
<tr>
<td>*pEncChData</td>
<td>Pointer to encoder channel status</td>
</tr>
</tbody>
</table>

**Description:** The `vcdEncoderInit` function initializes the encoder data structures (see Section 3.3.2). It is called once per transmission channel before the first call of the `vcdEncoder` function.

**Returns:** The `vcdEncoderInit` function iteratively returns the channel information used by the VAD/CNG/DTX algorithm.

**Special Issues:** The encoder channel status must be allocated prior to calling `vcdEncoderInit`.

The encoder initialization parameters (`EncConfig`) must be set prior to calling `vcdEncoderInit` and according to the encoder used.

**EncType** can have one of the following values:

- VCD_G711 : G.711 vocoder
- VCD_G726_16 : G.726 at 16 kbps (lowest quality)
- VCD_G726_24 : G.726 at 24 kbps
- VCD_G726_32 : G.726 at 32 kbps
- VCD_G726_40 : G.726 at 40 kbps (highest quality)

At any rate of the G.726 vocoder, `vcdEncoderInit` dynamically allocates memory to initialize the vocoder’s internal data structures. The dynamically-allocated memory is set free by calling `vcdEncoderDestroy`.

**Code Example:** In Code Example 3-2, the application executes an instance of `vcdEncoderInit`.

---

### Code Example 3-2. Use of vcdEncoderInit Interface

```c
#include "port.h"
#include "fcntl.h"
#include "vad_cng_dtx.h"
```
#define nRCs 10
#define VAD_ENABLED true
#define VOCODER_TYPE VCD_G711

int main(void)
{
    vcd_sEncStatus EncChData;
    vcd_sDecStatus DecChData;
    vcd_uEncConfig EncConfig;
    vcd_uDecConfig DecConfig;

    Word16 pSpeechBuffer[80];
    UInt8 EncodedData[80+1];
    Word16 synth[80];
    Word16 vad_enable = VAD_ENABLED;
    Word16 vocoder_type = VOCODER_TYPE;

    switch (vocoder_type)
    {
        case VCD_G711:
            EncConfig.g711Law= VCD_G711_A_LAW;
            DecConfig.g711Law= VCD_G711_A_LAW;
            break;
        case VCD_G726_40:
            EncConfig.g726Params.Flag_RATE= G726_ENC_RATE_40;
            EncConfig.g726Params.Flag_LAW= G726_ENC_A_LAW;
            DecConfig.g726Params.Flag_RATE= G726_DEC_RATE_40;
            DecConfig.g726Params.Flag_LAW= G726_DEC_A_LAW;
            vocoder_type= VCD_G726;
            break;
        case VCD_G726_32:
            EncConfig.g726Params.Flag_RATE= G726_ENC_RATE_32;
            EncConfig.g726Params.Flag_LAW= G726_ENC_A_LAW;
            DecConfig.g726Params.Flag_RATE= G726_DEC_RATE_32;
            DecConfig.g726Params.Flag_LAW= G726_DEC_A_LAW;
            vocoder_type= VCD_G726;
            break;
        case VCD_G726_24:
            EncConfig.g726Params.Flag_RATE= G726_ENC_RATE_24;
            EncConfig.g726Params.Flag_LAW= G726_ENC_A_LAW;
            DecConfig.g726Params.Flag_RATE= G726_DEC_RATE_24;
            DecConfig.g726Params.Flag_LAW= G726_DEC_A_LAW;
            vocoder_type= VCD_G726;
            break;
        case VCD_G726_16:
            EncConfig.g726Params.Flag_RATE= G726_ENC_RATE_16;
            EncConfig.g726Params.Flag_LAW= G726_ENC_A_LAW;
            DecConfig.g726Params.Flag_RATE= G726_DEC_RATE_16;
            DecConfig.g726Params.Flag_LAW= G726_DEC_A_LAW;
            vocoder_type= VCD_G726;
            break;
    }
    vcdEncoderInit(vocoder_type, nRCs, &EncConfig, &EncChData);
    vcdDecoderInit(vocoder_type, &DecConfig, &DecChData);
/* Put speech samples in pSpeechBuffer */
...

vcdEncoder(pSpeechBuffer, EncodedData, &EncChData, vad_enable);
vcdDecoder(EncodedData, synth, &DecChData);

vcdEncoderDestroy(&EncChData);
vcdDecoderDestroy(&DecChData);
3.3.2  vcdEncoder

Call(s):

```c
void vcdEncoder(
    IN Word16 *speech,                  /* input speech */
    OUT UInt8 *serial,              /* parameters to be sent to the channel */
    IN_OUT vcd_sEncStatus *pEncChData,  /* Encoder channel information */
    IN Word16 enable_vad                /* VAD enable flag */
);
```

Required Header: “vad_cng_dtx.h”

Arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*speech</td>
<td>in</td>
<td>Input speech</td>
</tr>
<tr>
<td>*serial</td>
<td>out</td>
<td>Parameters to be sent to the transmission channel</td>
</tr>
<tr>
<td>*pEncChData</td>
<td>inout</td>
<td>Pointer to encoder channel status structure</td>
</tr>
<tr>
<td>enable_vad</td>
<td>in</td>
<td>Switch to enable silence compression capabilities</td>
</tr>
</tbody>
</table>

Description: The vcdEncoder function is a generic encoder which includes VAD/DTX/CNG capabilities described by ITU-T G.711 Annex II and uses modules from ITU-T G.729AB. The vcdEncoder function is called once for each input signal frame for each transmission channel.

Returns: The vcdEncoder function returns the encoded parameters to be sent to the transmission channel and iteratively returns the channel information used by the VAD/CNG/DTX algorithm.

Special Issues: None

Code Example: In Code Example 3-2, the application executes an instance of vcdEncoder.
3.3.3 `vcdEncoderDestroy`

**Call(s):**

```c
Word16 vcdEncoderDestroy(
    IN_OUT vcd_sEncStatus *pEncChData   /* Encoder channel information */
);
```

**Required Header:** “vad_cng_dtx.h”

**Arguments:**

<table>
<thead>
<tr>
<th>*pEncChData</th>
<th>inout</th>
<th>Encoder channel information</th>
</tr>
</thead>
</table>

**Description:** The `vcdEncoderDestroy` function sets free the data structures needed to run the encoder (see Section 3.3.2). The `vcdEncoderDestroy` function is called once per transmission channel after the last call of the `vcdEncoder` function.

**Returns:** Upon successful completion, the `vcdEncoderDestroy` function returns an error code of “0”.

**Special Issues:** None

**Code Example:** In Code Example 3-2, the application executes an instance of `vcdEncoderDestroy`. 
3.3.4 vcdDecoderInit

Call(s):

```c
Word16 vcdDecoderInit(
    IN Word16 DecType,                  /* type of decoder                  */
    IN vcd_uDecConfig *DecConfig,       /* initialization parameters        */
    IN_OUT vcd_sDecStatus *pDecChData   /* Decoder channel information      */
); 
```

Required Header: “vad_cng_dtx.h”

Arguments:

<table>
<thead>
<tr>
<th>DecType</th>
<th>Type of decoder to run</th>
</tr>
</thead>
<tbody>
<tr>
<td>*DecConfig</td>
<td>Decoder initialization parameters (see Code Example 3-1)</td>
</tr>
<tr>
<td>*pDecChData</td>
<td>Pointer to decoder channel status</td>
</tr>
</tbody>
</table>

Description: The `vcdDecoderInit` function initializes the decoder data structures (see Section 3.3.5). The `vcdDecoderInit` function is called once per transmission channel before the first call of the `vcdDecoder` function.

Returns: The `vcdDecoderInit` function iteratively returns the channel information used by the VAD/CNG/DTX algorithm.

Special Issues: The decoder channel status must be allocated prior to calling `vcdDecoderInit`.

The decoder initialization parameters (`DecConfig`) must be set prior to calling `vcdDecoderInit` and according to the decoder used.

`DecType` can have one of the following values:

- VCD_G711 G.711 vocoder
- VCD_G726_16 G.726 at 16kbps (lowest quality)
- VCD_G726_24 G.726 at 24kbps
- VCD_G726_32 G.726 at 32kbps
- VCD_G726_40 G.726 at 40kbps (highest quality)

At any rate of the G.726 vocoder, `vcdDecoderInit` dynamically allocates memory to initialize the vocoder’s internal data structures. The dynamically-allocated memory is set free by calling `vcdDecoderDestroy`.

Code Example: In Code Example 3-2, the application executes an instance of `vcdDecoderInit`. 
3.3.5  vcdDecoder

Call(s):

```c
void vcdDecoder(
    IN Uint8 *serial,                   /* parameters received from channel */
    OUT Word16 *synth,                  /* decoded speech */
    IN_OUT vcd_sDecStatus *pDecChData   /* Decoder channel information */
)
```

Required Header: “vad_cng_dtx.h”

Arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*serial</td>
<td>in</td>
<td>Parameters received from the transmission channel</td>
</tr>
<tr>
<td>*synth</td>
<td>out</td>
<td>Decoded speech (output signal)</td>
</tr>
<tr>
<td>*pDecChData</td>
<td>inout</td>
<td>Pointer to decoder channel status structure</td>
</tr>
</tbody>
</table>

Description: The `vcdDecoder` function is a generic decoder which includes VAD/DTX/CNG capabilities described by ITU-T G.711 Annex II and uses modules from ITU-T G.729AB. The `vcdDecoder` function is called once for each bit-stream received for each transmission channel.

Returns: The `vcdDecoder` function returns the decoded speech and iteratively returns the channel information used by the VAD/CNG/DTX algorithm.

Special Issues: None

Code Example: In Code Example 3-2, the application executes an instance of `vcdDecoder`. 
### 3.3.6 `vcdDecoderDestroy`

Call(s):

```c
Word16 vcdDecoderDestroy(
    IN_OUT vcd_sDecStatus *pDecChData /* Decoder channel status */
);
```

**Required Header:** “vad_cng_dtx.h”

**Arguments:**

<table>
<thead>
<tr>
<th>*pDecChData</th>
<th>inout</th>
<th>Pointer to decoder channel information</th>
</tr>
</thead>
</table>

**Description:** The `vcdDecoderDestroy` function frees the data structures needed to run the decoder; see Section 3.3.5. The `vcdDecoderDestroy` function is called once per transmission channel after the last call of the `vcdDecoder` function.

**Returns:** Upon successful completion, the `vcdDecoderDestroy` function returns an error code of “0”.

**Special Issues:** None

**Code Example:** In Code Example 3-2, the application executes an instance of `vcdDecoderDestroy`. 
Chapter 4
Building the VAD/CNG/DTX Library

4.1 Building the VAD/CNG/DTX Library

The VAD/CNG/DTX Library combines all of the components described in the previous sections into one library: \texttt{vad\_cng\_dtx.lib}. This library’s code is not provided with the SDK; therefore, it cannot be built from the SDK. The VAD/CNG/DTX Library is provided in the \texttt{...\os\telephony\vad\_cng\_dtx} directory of the SDK directory structure.
Chapter 5
Linking Applications with the VAD/CNG/DTX Library

5.1 Calling VAD/CNG/DTX Library

The VAD/CNG/DTX library provides six entry points, described in Section 3. To invoke the VAD/CNG/DTX Library, the entry points must be called in the order shown in Table 5-1 and Table 5-2.

Table 5-1. Calling Order for vcdEncoder Interfaces

<table>
<thead>
<tr>
<th>Calling order</th>
<th>Function name</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 1             | vcdEncoderInit       | - Initializes the Encoder instance
|               |                      | - Called once per transmission channel             |
| 2             | vcdEncoder           | - Encodes a frame of speech
|               |                      | - Called multiple times                            |
| 3             | vcdEncoderDestroy    | - Frees the Encoder instance data structures
|               |                      | - Called once per transmission channel             |

Table 5-2. Calling Order for vcdDecoder Interfaces

<table>
<thead>
<tr>
<th>Calling order</th>
<th>Function name</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 1             | vcdDecoderInit       | - Initializes the Decoder instance
|               |                      | - Called once per transmission channel             |
| 2             | vcdDecoder           | - Decodes a frame of speech
|               |                      | - Called multiple times                            |
| 3             | vcdDecoderDestroy    | - Frees the Decoder instance data structures
|               |                      | - Called once per transmission channel             |
5.2 Recommended Memory Map

Code Example 5-1 contains a sample linker command file, `linker.cmd`, compatible with Code Warrior for DSP56800E Release 1.1. This sample is used in the test application for encoder and decoder.

Code Example 5-1.  linker.cmd file

```
#Code Warrior for DSP56800E Release 1.1
*******************************************************************************
#  Linker.cmd file for DSP56858 Internal RAM
#      using only internal program and data memory.
*******************************************************************************

MEMORY {

    .pInterruptVector   (RWX) : ORIGIN = 0x000000, LENGTH = 0x00008C
    .pIntRAM            (RWX) : ORIGIN = 0x00008C, LENGTH = 0x009F74
    .pIntROM            (RX)  : ORIGIN = 0x1F0000, LENGTH = 0x000400
    .xIntRAM            (RW)  : ORIGIN = 0x000100, LENGTH = 0x003F00
    .xStack             (RW)  : ORIGIN = 0x004000, LENGTH = 0x001000
    .xIntRAM_DynamicMem (RW)  : ORIGIN = 0x005000, LENGTH = 0x001000
    .xExtRAM_DynamicMem (RW)  : ORIGIN = 0x006000, LENGTH = 0x001000
    .xPeripherals       (RW)  : ORIGIN = 0x1FFC00, LENGTH = 0x000400
    .xCoreRegisters     (RW)  : ORIGIN = 0xFFFF00, LENGTH = 0x000100

*******************************************************************************

FORCE_ACTIVE {FconfigInterruptVector}
*******************************************************************************

SECTIONS {

    #*******************************************************************************

    .ApplicationInterruptVector :
        
            vector.c (.text)
        
} > .pInterruptVector

*******************************************************************************
```
.ApplicationCode :
{
  # Place all code into Program RAM

  *(.text)
  *(rtlib.text)
  *(fp_engine.text)
  *(user.text)
  *(COMMON.text)
  *(ENCODER.text)
  *(DECODER.text)

  # Place all data into Program RAM

  F_Pdata_start_addr_in_ROM = 0;
  F_Pdata_start_addr_in_RAM = .;
  pramdata.c (.data)
  F_Pdata_ROMtoRAM_length = 0;

  F_Pbss_start_addr = .;
  _P_BSS_ADDR = .;
  pramdata.c (.bss)
  F_Pbss_length = . - _P_BSS_ADDR;

} > .pIntRAM

****************************************************************************

/ApplicationData :
{
  # Define variables for C initialization code

  F_Xdata_start_addr_in_ROM = .;
  F_StackAddr = ADDR(.xStack);
  F_StackEndAddr = ADDR(.xStack) + SIZEOF(.xStack) - 1;
  F_Xdata_start_addr_in_RAM = .;

  # Define variables for SDK mem library

  # Data (X) Memory Layout

  _EX_BIT = 0;

  # Internal Memory Partitions (for mem.h partitions)

  _NUM_IM_PARTITIONS = 1;  # IM_ADDR_1 (no IM_ADDR_2 )

  # External Memory Partition (for mem.h partitions)

  _NUM_EM_PARTITIONS = 0;  # EM_ADDR_1
Linking Applications with the VAD/CNG/DTX Library

FmemEXbit = .;
WRITEH(_EX_BIT);
FmemNumIMpartitions = .;
WRITEH(_NUM_IM_PARTITIONS);
FmemNumEMpartitions = .;
WRITEH(_NUM_EM_PARTITIONS);
FmemIMpartitionAddr = ADDR(.xIntRAM_DynamicMem);
FmemIMpartitionSize = .;
WRITEH(SIZEOF(.xIntRAM_DynamicMem)*1);
FmemEMpartitionAddr = ADDR(.xExtRAM_DynamicMem);
FmemEMpartitionSize = .;
WRITEH(SIZEOF(.xExtRAM_DynamicMem)*1);

# Add rest of the data into Internal RAM
* (.const.data)
* (.data)
* (fp_state.data)
* (rtlib.data)
* (VAD_CNG_DTX_TABLES.data)
* (G726_INTERNAL_ROM.data)

F_Xdata_ROMtoRAM_length = 0;

F_Xbss_start_addr = .;
_X_BSS_ADDR = .;

* (rtlib.bss.lo)
* (.bss)
* (G726_INTERNAL_ROM.bss)

F_Xbss_length = . - _X_BSS_ADDR;  # Copy DATA

} > .xIntRAM

*****************************************************************************

FArchIO = 0x0000;
FArchCore = ADDR(.xCoreRegisters);
FArchInterrupts = ADDR(.pInterruptVector);
}

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005
Chapter 6
VAD/CNG/DTX Applications

6.1 Test and Demo Applications

To verify the VAD/CNG/DTX library, test and demo applications have been developed. Refer to the Targeting Motorola DSP568xx Platform Manual for the DSP you are using to see if the test and demo applications are available for your target.
Chapter 7
License

7.1 Limited Use License Agreement

LIMITED USE LICENSE AGREEMENT

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THIS SOFTWARE. BY USING OR COPYING THE SOFTWARE, YOU AGREE TO THE TERMS OF THIS AGREEMENT.

The software in either source code form ("Source") or object code form ("Object") (cumulatively hereinafter "Software") is provided under a license agreement ("Agreement") as described herein. Any use of the Software including copying, modifying, or installing the Software so that it is usable by or accessible by a central processing unit constitutes acceptance of the terms of the Agreement by the person or persons making such use or, if employed, the employer thereof ("Licensee") and if employed, the person(s) making such use hereby warrants that they have the authority of their employer to enter this license agreement. If Licensee does not agree with and accept the terms of this Agreement, Licensee must return or destroy any media containing the Software or materials related thereto, and destroy all copies of the Software.

The Software is licensed to Licensee by Motorola Incorporated ("Motorola") for use under the terms of this Agreement. Motorola retains ownership of the Software. Motorola grants only the rights specifically granted in this Agreement and grants no other rights. Title to the Software, all copies thereof and all rights therein, including all rights in any intellectual property including patents, copyrights, and trade secrets applicable thereto, shall remain vested in Motorola.

For the Source, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to use, copy, and make derivatives of the Source solely in a development system environment in order to produce object code solely for operating on a Motorola semiconductor device having a central processing unit ("Derivative Object").

For the Object and Derivative Object, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to copy, use, and distribute the Object and the Derivative Object solely for operating on a Motorola semiconductor device having a central processing unit.

Licensee agrees to: (a) not use, modify, or copy the Software except as expressly provided herein, (b) not distribute, disclose, transfer, sell, assign, rent, lease, or otherwise make available the Software, any derivatives thereof, or this license to a third party except as expressly provided herein, (c) not remove obliterate, or otherwise defeat any copyright, trademark, patent or proprietary notices, related to the Software (d) not in any form export, re-export, resell, ship or divert or cause to be exported, re-exported, resold, shipped, or diverted, directly or indirectly, the Software or a direct product thereof to any country which the United States government or any agency thereof at the time of export or re-export requires an export license or other government approval without first obtaining such license or approval.
THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS AND WITHOUT WARRANTY OF ANY KIND INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MOTOROLA BE LIABLE FOR ANY LIABILITY OR DAMAGES OF ANY KIND INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT OR INCIDENTAL OR CONSEQUENTIAL OR PUNITIVE DAMAGES OR LOST PROFITS OR LOSS OF USE ARISING FROM USE OF THE SOFTWARE OR THE PRODUCT REGARDLESS OF THE FORM OF ACTION OR THEORY OF LIABILITY (INCLUDING WITHOUT LIMITATION, ACTION IN CONTRACT, NEGLIGENCE, OR PRODUCT LIABILITY) EVEN IF MOTOROLA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THIS DISCLAIMER OF WARRANTY EXTENDS TO LICENSEE OR USERS OF PRODUCTS AND IS IN LIEU OF ALL WARRANTIES WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE.

Motorola does not represent or warrant that the Software is free of infringement of any third party patents, copyrights, trade secrets, or other intellectual property rights or that Motorola has the right to grant the licenses contained herein. Motorola does not represent or warrant that the Software is free of defect, or that it meets any particular requirements or need of the Licensee, or that it conforms to any documentation, or that it meets any standards.

Motorola shall not be responsible to maintain the Software, provide upgrades to the Software, or provide any field service of the Software. Motorola reserves the right to make changes to the Software without further notice to Licensee.

The Software is not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Software could create a situation where personal injury or death may occur. Should Licensee purchase or use the Software for any such unintended or unauthorized application, Licensee shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the Software.

The term of this Agreement is for as long as Licensee uses the Software for its intended purpose and is not in default of any provisions of this Agreement. Motorola may terminate this Agreement if Licensee is in default of any of the terms and conditions of this Agreement.

This Agreement shall be governed by and construed in accordance with the laws of the State of Arizona and can only be modified in a writing signed by both parties. Licensee agrees to jurisdiction and venue in the State of Arizona.

By using, modifying, installing, compiling, or copying the Software, Licensee acknowledges that this Agreement has been read and understood and agrees to be bound by its terms and conditions. Licensee agrees that this Agreement is the complete and exclusive statement of the agreement between Licensee and Motorola and supersedes any earlier proposal or prior arrangement, whether oral or written, and any other communications relative to the subject matter of this Agreement.
Index

A
ACK x

C
CL x
CLR x
CR xi

D
DSP xi
DSP56800 Family Manual ix
DSP56824 User’s Manual ix
DTE xi

E
Embedded SDK Programmer’s Guide ix

F
FCS xi

G
G.729AB Vocoder Library 33

I
I/O xi
IDE xi

L
LSB xi

M
MAC xi
MIPS xi
MS xi
MSB xi

N
NAK xi

O
OMR xi
OnCE xi
Optional Directories 18

P
PC xi
PSTN xi

R
Required Core Directories 17

S
SDK xi
SP xi
SPI xi
SR xi
SRC xi
Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and the Stylized M Logo are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

MOTOROLA and the Stylized M Logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners. © Motorola, Inc. 2002.

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447
ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26688334
Technical Information Center: 1–800–521–6274
HOME PAGE: http://www.motorola.com/semiconductors/

For More Information On This Product, Go to: www.freescale.com

SDK148/D