Continuous Speech Processing API for Linux and Windows Operating Systems

Programming Guide

October 2004

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

This Continuous Speech Processing API for Linux and Windows Operating Systems Programming Guide as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without express written consent of Intel Corporation.

Copyright © 2000-2004 Intel Corporation

BunnyPeople, Celeron, Chips, Dialogic, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel Centrino, Intel Centrino logo, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Xeon, Intel XScale, IPLink, Itanium, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, Sound Mark, The Computer Inside., The Journey Inside, VTune, and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.

Publication Date: October 2004

Document Number: 05-1699-004

Intel Converged Communications, Inc.
1515 Route 10
Parsippany, NJ 07054

For **Technical Support**, visit the Intel Telecom Support Resources website at:
http://developer.intel.com/design/telecom/support

For **Products and Services Information**, visit the Intel Telecom Products website at:
http://www.intel.com/design/network/products/telecom

For **Sales Offices** and other contact information, visit the Where to Buy Intel Telecom Products page at:
http://www.intel.com/buy/wtb/wtb1028.htm

Table of Contents

Revision History 7

About This Publication 9

1 **Product Description** 13

- 1.1 Key Features 13
- 1.2 CSP Components 14
- 1.3 Echo Canceller 17
  - 1.3.1 Echo Canceller Overview 17
  - 1.3.2 Tap Length 18
  - 1.3.3 Adaptation Modes 18
- 1.4 Voice Activity Detector (VAD) 19
- 1.5 Pre-Speech Buffer 19
- 1.6 Barge-In and Voice Event Signaling 20
- 1.7 Streaming to the Host 20
- 1.8 Silence Compressed Streaming 21
- 1.9 Streaming to the TDM Bus 21
- 1.10 Supported Data Formats 21
  - 1.10.1 Supported Data Formats on DM3 Boards. 22
  - 1.10.2 Supported Data Formats on Springware Boards 22
- 1.11 Comparison with Existing Features 23
- 1.12 CSP Support on Springware Versus DM3 Boards 23

2 **Event Handling** 27

3 **Error Handling** 29

4 **Application Development Guidelines** 31

- 4.1 Programming Considerations 31
  - 4.1.1 Introduction 31
  - 4.1.2 Reserving Extra Time Slots for Streaming to TDM Bus 32
  - 4.1.3 Opening a Voice Channel 32
  - 4.1.4 Assigning Time Slots 32
  - 4.1.5 Configuring the CSP Device Channel Using ec_setparm( ) 33
  - 4.1.6 Configuring the CSP Device Channel Using dx_setparm( ) 33
  - 4.1.7 Setting Up VAD Event Notification 33
  - 4.1.8 Setting Up EC Convergence Event Notification 33
  - 4.1.9 Setting Up Streaming or Recording 34
  - 4.1.10 Setting Up Silence Compressed Streaming 34
  - 4.1.11 Playing a Prompt 34
  - 4.1.12 Collecting Events 35
  - 4.1.13 Performing Voice Processing 35
  - 4.1.14 Cleaning Up 35
- 4.2 Interoperability Considerations for Springware Boards 35
  - 4.2.1 Transaction Record 35
  - 4.2.2 DSP-Based Fax 35

4.2.3 ISDN 36

5 **Using the Voice Activity Detector** 37

5.1 Voice Activity Detector Operating Modes 37
5.1.1 Overview 37
5.1.2 Sending a VAD Event to the Host Application 38
5.1.3 Stopping Play When Speech is Detected (Barge-In) 39
5.1.4 Voice-Activated or Constant Recording 39
5.1.5 Silence Compressed Streaming 39
5.1.6 Sample VAD Scenarios 40
5.2 VAD Operation 43
5.3 Fine-Tuning VAD Performance 44
5.3.1 Overview 44
5.3.2 Fine-Tuning VAD Performance on Springware Boards 44
5.3.3 Fine-Tuning VAD Performance on DM3 Boards 45

6 **Buffers and Data Flow** 47

6.1 Types of Buffers 47
6.2 Data Flow 49
6.3 Buffer Usage Tips 52

7 **Echo Canceller Convergence Notification** 53

8 **Building Applications** 55

8.1 CSP Library Integration with Voice Libraries 55
8.2 Compiling and Linking 56
8.2.1 Include Files 56
8.2.2 Required Libraries 57
8.2.3 Variables for Compiling and Linking 58

Glossary 59

Index 63

Figures

1 CSP Components and Data Flow 15
2 Echo Cancellation Using an External Reference Signal 16
3 Echo Canceller 17
4 Example of Silence Compressed Streaming Operation 40
5 Example of Voice Activity Detector (VAD) Operation 43
6 Data Flow from Application to Firmware (Springware Boards) 50
7 Data Flow from Application to Firmware (DM3 Boards) 51
8 CSP, SRL and Voice Libraries. 55

Tables

1 Feature Comparison 23
2 CSP Support on Springware Boards and DM3 Boards 24
3 VAD and Barge-In Operating Modes 38
4 Sample VAD Scenarios 41
5 Types of Buffers Used in CSP (Springware Boards) 48
6 Types of Buffers Used in CSP (DM3 Boards) 49

Revision History

This revision history summarizes the changes made in each published version of this document.

Document No.	Publication Date	Description of Revisions
05-1699-004	October 2004	Product Description chapter: In Supported Data Formats on DM3 Boards section, revised information about WAVE and VOX file formats. In Supported Data Formats on Springware Boards section, revised information about WAVE and VOX file formats. Types of Buffers Used in CSP (DM3 Boards) table: Revised low-end range for ECCH_XFERBUFFERSIZE on DM3 boards. Echo Canceller Convergence Notification chapter: Added further detail on TEC_CONVERGED event (PTR #31514).
05-1699-003	November 2003	Key Features section: Removed bullet item “Ability to re-arm or re-enable the voice activity detector.” This feature is no longer supported. It is superseded by the new silence compressed streaming feature (supported on DM3 boards only). Silence Compressed Streaming section: Added new section in Product Description chapter on silence compressed streaming. Supported Data Formats on Springware Boards section: Added note in Product Description chapter about DXCH_EC_TAP_LENGTH and data formats supported. Configuring the CSP Device Channel Using ec_setparm( ) section: Added note in Application Development Guidelines chapter about DXCH_EC_TAP_LENGTH and data formats supported. Setting Up Silence Compressed Streaming section: Added new section in Application Development Guidelines chapter on silence compressed streaming. Silence Compressed Streaming section: Added new section on silence compressed streaming in Using the Voice Activity Detector chapter.
05-1699-002	August 2002	Global change: Modifying tap length is now supported on DM3 boards. Global change: DX_MAXTIME termination condition is now supported on DM3 boards. Product Description chapter, CSP Components section: Revised text to indicate that sending echo-cancelled data over the TDM bus to another board is now supported on DM3 boards. Product Description chapter, Streaming to the TDM Bus section: New section. Product Description chapter, Supported Data Formats on DM3 Boards section: Revised list of coders supported for playing files. Buffers and Data Flow chapter, Types of Buffers Used in CSP (Springware Boards) table: Revised table and added new information on firmware buffers. Building Applications chapter, Required Libraries section: Removed certain library files. No longer required in Linux. Building Applications chapter, Variables for Compiling and Linking section: New section.
05-1699-001	December 2001	Initial version of document. Much of the information contained in this document was previously contained in the *Continuous Speech Processing Software Reference for UNIX and Windows*, document number 05-1398-003.

About This Publication

The following topics provide information about this *Continuous Speech Processing API for Linux and Windows Operating Systems Programming Guide*:

- Purpose
- Applicability
- Intended Audience
- How to Use This Publication
- Related Information

Purpose

This publication provides guidelines for building applications using the Continuous Speech Processing (CSP) software and voice software in a Linux* or Windows* environment.

It is a companion guide to the *Continuous Speech Processing API Library Reference* which provides details on functions and parameters in the CSP library.

Applicability

This document version (05-1699-004) is published for Intel® Dialogic® System Release 6.1 Feature Release 1 for Linux operating system.

This document may also be applicable to later Intel® Dialogic® system releases on Linux and Windows as well as Intel® NetStructure™ Host Media Processing (HMP) software releases. Check the Release Guide for your software release to determine whether this document is supported.

Intended Audience

This publication is written for the following audience:

- Distributors
- System Integrators
- Toolkit Developers
- Independent Software Vendors (ISVs)
- Value Added Resellers (VARs)
- Original Equipment Manufacturers (OEMs)

How to Use This Publication

Refer to this publication after you have installed the hardware and the system software which includes the CSP software.

This publication assumes that you are familiar with the Linux or Windows operating system and the C programming language. It is helpful to keep the *Voice API Library Reference* and *Voice API Programming Guide* handy as you develop your application.

The information in this guide is organized as follows:

- Chapter 1, "Product Description" introduces you to CSP, its key features, how CSP works.
- Chapter 2, "Event Handling" defines an event and describes how to handle an event.
- Chapter 3, "Error Handling" presents information on how to obtain error codes and handle errors.
- Chapter 4, "Application Development Guidelines" presents guidelines for application development, including interoperability considerations for Springware boards.
- Chapter 5, "Using the Voice Activity Detector" discusses the voice activity detector (VAD) in detail. It tells you how to set parameters for various outcomes, and how to fine-tune VAD performance.
- Chapter 6, "Buffers and Data Flow" discusses the types of buffers used by CSP, buffer usage tips, and data flow.
- Chapter 7, "Echo Canceller Convergence Notification" explains how to send a notification event to the application when the echo canceller has converged.
- Chapter 8, "Building Applications" provides information on compiling and linking, such as header files to be included and library files to be linked.
- Glossary provides a definition of terms used in this guide.

Related Information

Refer to the following documents and Web site for more information on developing your application:

- For details on all voice functions and parameters in the CSP library, see the *Continuous Speech Processing API Library Reference*.
- For instructions for running the CSP demo, see the *Continuous Speech Processing API Demo Guide*.
- For information about Voice library features and guidelines for building applications using Voice software, see the *Voice API Programming Guide*.
- For details on all voice functions, parameters, and data structures in the Voice library, see the *Voice API Library Reference*.
- For details on the Standard Runtime Library (a device-independent library that consists of event management functions and standard attribute functions), supported programming models, and programming guidelines for building all applications, see the *Standard Runtime Library API Programming Guide*.

- For details on all functions and data structures in the Standard Runtime Library library, see the *Standard Runtime Library API Library Reference*.
- For information on the system release, system requirements, software and hardware features, supported hardware, and release documentation, see the Release Guide.
- For details on known problems and late-breaking updates or corrections to the release documentation, see the Release Update. Be sure to check the Release Update for the software release you are using for any updates or corrections to this publication. Release Updates are available on the Telecom Support Resources website at *http://resource.intel.com/telecom/support/documentation/releases/index.htm*
- For details on installing the system software, see the *System Release Installation Guide*.
- For guidelines on building applications using Global Call software (a common signaling interface for network-enabled applications, regardless of the signaling protocol needed to connect to the local telephone network), see the *Global Call API Programming Guide*.
- For details on all functions and data structures in the Global Call library, see the *Global Call API Library Reference*.
- For details on configuration files (including FCD/PCD files) and instructions for configuring products, see the Configuration Guide for your product or product family.

Product Description 1

This chapter provides information about Continuous Speech Processing product features. The following topics are covered:

- Key Features 13
- CSP Components. 14
- Echo Canceller. 17
- Voice Activity Detector (VAD) 19
- Pre-Speech Buffer 19
- Barge-In and Voice Event Signaling 20
- Streaming to the Host 20
- Silence Compressed Streaming 21
- Streaming to the TDM Bus 21
- Supported Data Formats 21
- Comparison with Existing Features. 23
- CSP Support on Springware Versus DM3 Boards. 23

1.1 Key Features

The Continuous Speech Processing (CSP) software provides a high-level interface to Intel® telecom boards and is a building block for creating host-based automatic speech recognition (ASR) applications. CSP gives you the ability to stream voice-activated, pre-speech buffered, echo-cancelled voice data to an ASR engine.

CSP consists of a library of functions, device drivers, firmware, sample demonstration programs and technical documentation to help you create leading-edge ASR applications. It is a significant enhancement to existing echo cancellation resource (ECR) and barge-in technology.

Key features of CSP include:

- Full-duplex operation which means the capability of simultaneously sending and receiving (playing and recording) voice data on a single CSP channel.
- Echo canceller that significantly reduces echo in the incoming signal (up to 64 ms on select DM3 boards and up to 16 ms on Springware boards).
- Voice activity detector (VAD) that determines when significant audio energy is detected on the channel and enables data to be sent only when speech is present, thereby reducing CPU loading.
- Voice event signaling capability which means that when the VAD detects significant energy in the incoming signal, the CSP firmware can optionally send a message to the host application.

- Barge-in capability which allows a party to speak or enter keypad digits without waiting for the end of a prompt.
- Pre-speech buffering which reduces the problem of clipped speech and increases recognition accuracy.
- Ability to modify certain VAD parameters on the fly. You can modify certain VAD parameters, such as the speech threshold, while streaming or recording is in progress.
- Ability to generate both TDX_BARGEIN and TDX_PLAY events when a prompt is interrupted, rather than just TDX_BARGEIN event.
- Ability to send an external reference signal (echo-reference signal) from another device across the TDM bus to the CSP voice channel. Using this feature allows you to share the echo canceller and VAD resource on one CSP voice channel with other devices.
- Ability to stream echo-cancelled data across the TDM bus to another device.
- Demonstration program that illustrates the key features of CSP.

The following features are available on DM3 boards only:

- Echo canceller convergence event notification. An event can be sent to the host application when the echo canceller has converged (echo component has been significantly reduced).
- A more powerful voice activity detector (VAD). The VAD performs sophisticated calculations using a combination of energy and zero-crossing mode to accurately determine the start of speech.
- Silence compressed streaming. Only signal samples containing energy are sent to the host application. Periods of silence are significantly reduced from the echo-cancelled data.

For a description of technical terms, see the Glossary.

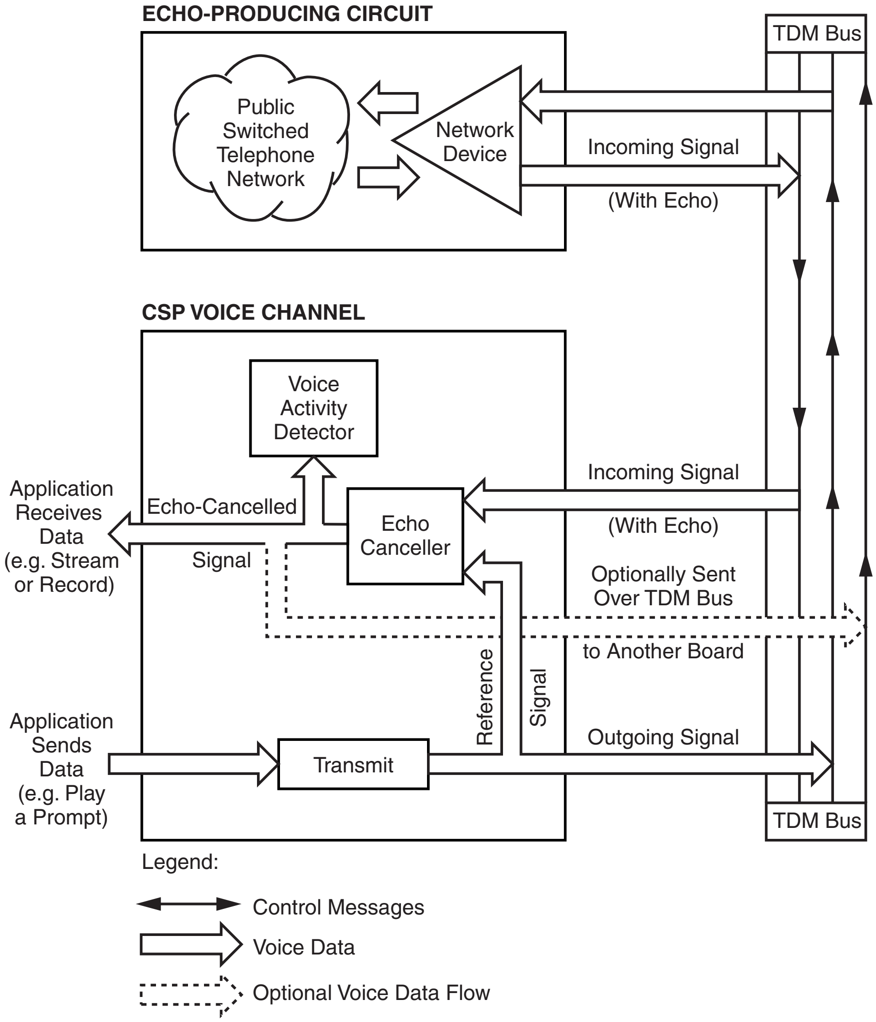
1.2 CSP Components

The CSP software consists of several CSP components, many of which reside in the firmware level of the board:

- echo canceller
- voice activity detector (VAD)
- pre-speech buffer
- barge-in and voice event signaling
- streaming or recording

Figure 1 depicts the data flow from the network to the CSP voice channel. This figure shows how echo is introduced in the signal in the network and how it is cancelled. It also illustrates the option of sending echo-cancelled data over the TDM bus to another board, regardless of whether this second board is CSP-capable or not.

Figure 1. CSP Components and Data Flow


ECHO-PRODUCING CIRCUIT
Public Switched Telephone Network
Network Device
TDM Bus
Incoming Signal
(With Echo)
CSP VOICE CHANNEL
Voice Activity Detector
Application Receives Data (e.g. Stream or Record)
Echo-Cancelled
Signal
Echo Canceller
Incoming Signal
(With Echo)
Optionally Sent Over TDM Bus
to Another Board
Reference
Signal
Application Sends Data (e.g. Play a Prompt)
Transmit
Outgoing Signal
TDM Bus
Legend:
Control Messages
Voice Data
Optional Voice Data Flow

To help you understand how CSP works, consider the following scenario of an ASR auto-attendant application.

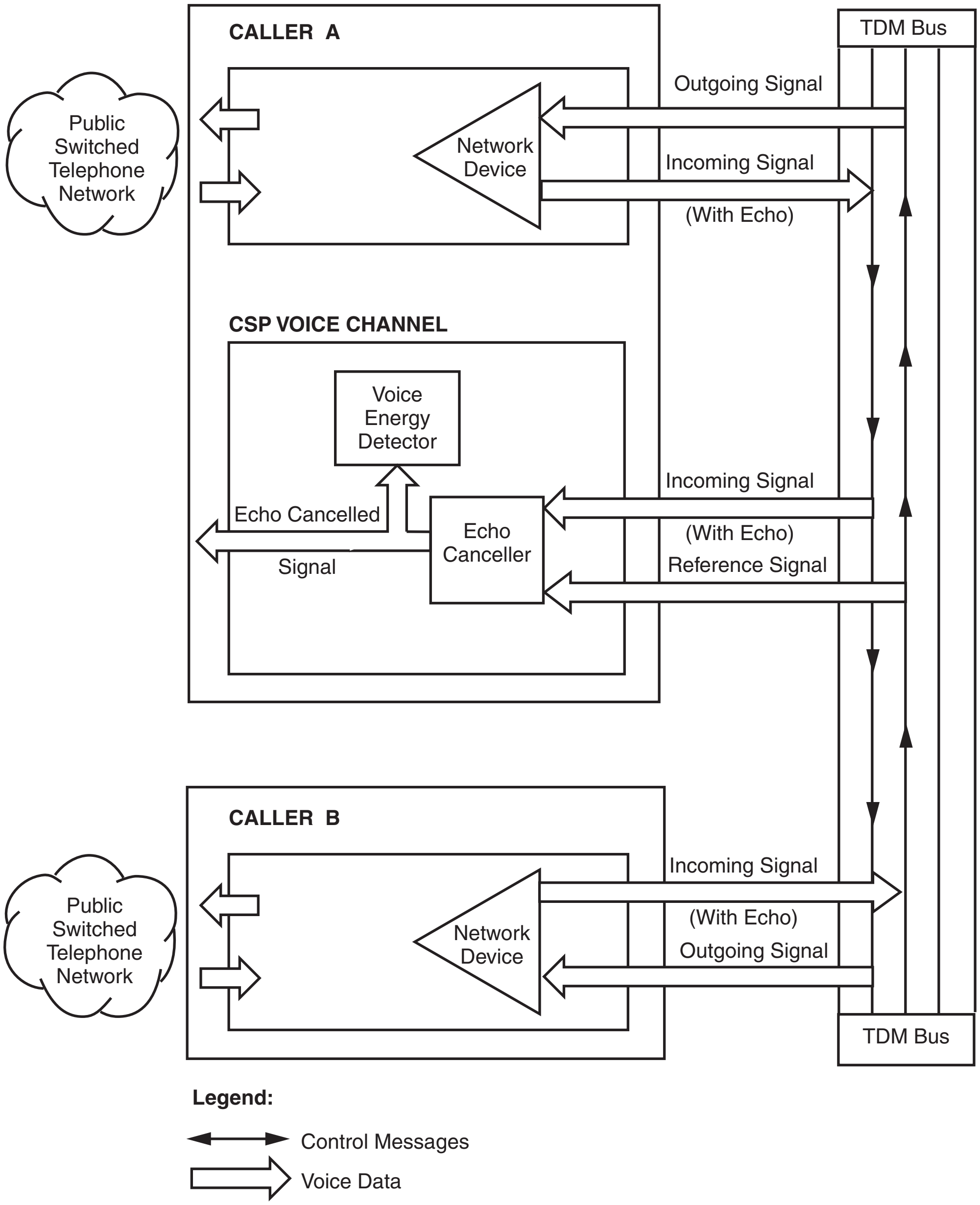
A caller telephones XYZ company, listens to the welcome greeting (the outgoing signal), and interrupts the prompt by speaking the name of the person she wishes to contact, for example, “Steve Smith” (this is the incoming signal).

If the CSP software detects energy above the configurable threshold in the incoming signal, CSP then terminates the prompt (this is barge-in), removes echo from the incoming signal and forwards the “Steve Smith” signal, with the pre-threshold energy that has been buffered, to the ASR application. The application recognizes the name, correctly responds to the request, and connects the caller to the intended audience.

Note: The CSP software does **not** include an ASR or text-to-speech (TTS) resource. Therefore, the CSP software does not determine whether energy is speech.

Figure 2 illustrates how you can send a reference signal from a non-CSP device over the TDM bus to the CSP voice channel. This external reference signal can be from a secondary network device.

Figure 2. Echo Cancellation Using an External Reference Signal


CALLER A
TDM Bus
Public Switched Telephone Network
Network Device
Outgoing Signal
Incoming Signal
(With Echo)
CSP VOICE CHANNEL
Voice Energy Detector
Echo Cancelled
Signal
Echo Canceller
Incoming Signal
(With Echo)
Reference Signal
CALLER B
Public Switched Telephone Network
Network Device
Incoming Signal
(With Echo)
Outgoing Signal
TDM Bus
Legend:
Control Messages
Voice Data

In Figure 2, caller A and caller B are in conversation. There is a full-duplex connection between these two callers. Caller A's signal is received on the same physical device on which the echo canceller is located. Caller B's signal is from a secondary network device.

The CSP voice channel listens to two signals: the incoming signal from caller A (which contains echo) and the incoming signal from caller B (which serves as an external reference signal to reduce the echo).

By using caller B's incoming signal as a reference signal, the echo canceller produces an echo-cancelled signal for caller A.

1.3 Echo Canceller

This section discusses the echo canceller component:

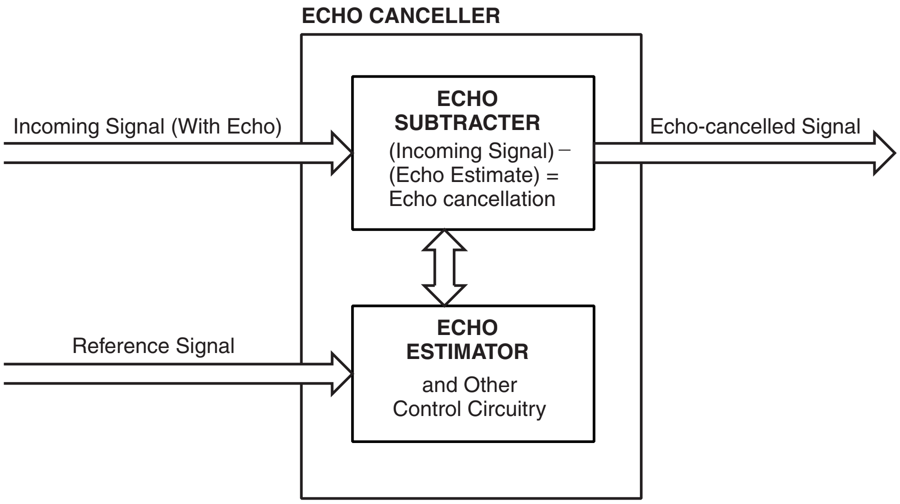
- Echo Canceller Overview
- Tap Length
- Adaptation Modes

1.3.1 Echo Canceller Overview

The echo canceller is a component in the CSP software that is used by applications to eliminate echo in the incoming signal. In the scenario described in Section 1.2, "CSP Components", on page 14, the incoming signal is the utterance "Steve Smith." Because of the echo canceller, the "Steve Smith" signal has insignificant echo and can be processed more accurately by the speech recognition engine.

Figure 3 shows a close-up view of how the echo canceller works. After the incoming signal is processed by the echo canceller, the resulting signal no longer has significant echo and is then sent to the host application.

Figure 3. Echo Canceller


ECHO CANCELLER
Incoming Signal (With Echo)
ECHO SUBTRACTER
(Incoming Signal) − (Echo Estimate) = Echo cancellation
Echo-cancelled Signal
Reference Signal
ECHO ESTIMATOR
and Other Control Circuitry

If echo cancellation is not used, the incoming signal usually contains an echo of the outgoing prompt. Without echo cancellation, an application must ignore all incoming energy until the prompt and its echo terminate. These applications typically have an announcement that says, "At the tone, please say the name of the person you wish to reach."

With echo cancellation, the caller may interrupt the prompt, and the incoming speech signal can be passed to the ASR application.

The echo canceller is controlled by the ECCH_ECHOCANCELLER parameter in **ec_setparm( )**. It is turned on by default. For more information, see the *Continuous Speech Processing API Library Reference*.

1.3.2 Tap Length

The duration of an echo is measured in tens of milliseconds. An echo canceller can remove some limited number of these milliseconds, and this number is known as the length of the echo canceller. The length of an echo canceller is sometimes given as "taps," where each tap is 125 microseconds long.

The longer the tap length, the more echo is cancelled from the incoming signal. However, this means more processing power is required. When determining the tap length value, consider the length of the echo delay in your system as well as your overall system configuration.

On DM3 boards, the media load which is downloaded when you start the board determines what tap length values are supported. Some DM3 boards support one value only, 128 taps (16 ms). Other DM3 boards support 512 taps (64 ms). For information on media loads, see the appropriate Configuration Guide for your product or product family. For information on tap length support on DM3 boards, see the Release Guide for the system release you are using.

On Springware boards, to configure the tap length of the echo canceller, use DXCH_EC_TAP_LENGTH in **ec_setparm( )**. For more information, see the *Continuous Speech Processing API Library Reference*.

1.3.3 Adaptation Modes

On DM3 boards, the adaptation mode parameter is not supported.

The echo canceller has two adaptation modes:

- Fast mode, for rapid convergence. Fast mode is used immediately after the echo canceller is reset and once energy is detected on the reference signal. Fast mode means that a higher adaptation gain factor (AGF) is used. Convergence occurs at a faster rate, but the residual error or echo is greater than in slow mode.
- Slow mode, for slower convergence. Slow mode is used the rest of the time. This second mode is entered automatically after a few seconds of fast mode. Slow mode means that a lower adaptation gain factor (AGF) is used. Convergence occurs at a slower rate, but residual error or echo is lower.

To configure the mode, use the ECCH_ADAPTMODE parameter in **ec_setparm( )**. For more information, see the *Continuous Speech Processing API Library Reference*.

Regardless of the parameter value, the echo canceller always starts with a higher automatic gain factor (fast mode) after it is reset, and then switches to a lower automatic gain factor (slow mode).

By starting with fast mode, then switching to slow mode, the echo canceller can converge rapidly and achieve smaller residual echo.

Two factors are used in determining the switch from fast to slow mode:

- Echo Return Loss Enhancement (ERLE)
- Adaptation time (a few seconds)

When ECCH_ADAPTMODE is set to 0, both factors are used. When ECCH_ADAPTMODE is set to 1, the second factor only (adaptation time) is used.

1.4 Voice Activity Detector (VAD)

When a caller begins to speak over a prompt (also known as barge-in), the application typically stops the playing of the prompt so that it isn't distracting to the caller.

A voice activity detector (VAD) is a component in the CSP software that examines the caller's incoming signal and determines if the signal contains significant energy and is likely to be speech rather than a click, for example. The significance is determined by configurable parameters.

The VAD has several configurable parameters such as the threshold of energy that is considered significant during prompt play and after the prompt has completed play. For more information, see parameter descriptions in **ec_setparm( )** in the *Continuous Speech Processing API Library Reference*.

For information on the VAD, see Chapter 5, "Using the Voice Activity Detector". For information on choices of operating modes for the VAD, see Section 5.1, "Voice Activity Detector Operating Modes", on page 37.

1.5 Pre-Speech Buffer

The VAD does not usually detect an utterance just as it arrives; instead, the energy of the utterance builds until the utterance triggers the VAD. For example, the name "Steve", when pronounced, begins with a low-energy hiss.

If the VAD is monitoring the incoming energy, and only sends the signal to the application after the beginning of an utterance is detected, then most likely the low-energy start of the utterance will be missing. The ASR engine requires the complete speech utterance to correctly process the signal to fulfill the caller's request.

To avoid this problem, the CSP software stores a **pre-speech buffer**; that is, a recording of the echo-cancelled incoming energy prior to the VAD trigger. The data in the pre-speech buffer is sent to the application along with all subsequent energy signals. Pre-speech buffers are an integral part of VAD. See Figure 6, "Data Flow from Application to Firmware (Springware Boards)", on page 50 and Figure 7, "Data Flow from Application to Firmware (DM3 Boards)", on page 51 for an illustration of the pre-speech buffer.

There is one pre-speech buffer per voice channel. A pre-speech buffer can hold 250 milliseconds of speech when a sampling rate of 8000 samples per second and a sampling size of 8 bits per sample are used.

1.6 Barge-In and Voice Event Signaling

The combination of echo cancellation (EC) and voice activity detector (VAD) can be used to effect **barge-in**, the act of speaking over a prompt. EC significantly reduces the echo of the prompt from the incoming speech signal. The VAD detects the beginning of an utterance and optionally can send a VAD event to the host application.

The barge-in feature stops the playing of the prompt upon detection of audio energy exceeding the threshold. For some applications, you may choose to halt the prompt based on other criteria; for example, only after the caller utters a valid vocabulary word. In this case, CSP can be set to inform the application that energy has been detected without terminating the prompt playback. This is called **voice event signaling**. CSP then sends the pre-speech buffer followed by the speech buffers in real time to the application.

For more information on setting barge-in and sending notification to the host application, see Section 5.1.3, "Stopping Play When Speech is Detected (Barge-In)", on page 39 and Section 5.1.2, "Sending a VAD Event to the Host Application", on page 38. For detailed function and parameter descriptions, see **ec_setparm( )** in the *Continuous Speech Processing API Library Reference*.

1.7 Streaming to the Host

You can choose to stream echo-cancelled data at all times or only after the VAD has detected energy. Voice-activated recording refers to the process of recording or streaming data to the host application only after the VAD has detected energy. Data in the pre-speech buffer is also sent to the host application as part of this process.

Note: In the industry, the terms recording and streaming are often used interchangeably. In this document, the terms have slight differences in meaning. Streaming refers to the transfer of data from the firmware to the application on the host. Recording refers to the transfer of data from the firmware to a file or memory buffer on the host.

The ECCH_VADINITIATED parameter in **ec_setparm( )** controls whether speech data is transmitted constantly or only after the VAD detects an utterance. Data is transmitted using the **ec_reciottdata( )** or **ec_stream( )** functions. For more information, see Section 5.1.6, "Sample VAD Scenarios", on page 40 and the function descriptions in the *Continuous Speech Processing API Library Reference*.

1.8 Silence Compressed Streaming

On Springware boards, the silence compressed streaming feature is not supported.

Silence compressed streaming (SCS) refers to the process of streaming audio energy to the host application with silence periods significantly reduced.

Silence compressed streaming provides the following benefits:

- allows for more efficient use of available bandwidth by reducing the amount of data streamed from the board to the host application
- provides a useful mechanism for identifying and handling non-speech such as a cough
- acts as a preliminary energy detector

To enable silence compressed streaming, set the ECCH_SILENCECOMPRESS parameter in **ec_setparm( )**. Data is transmitted using the **ec_stream( )** and **ec_reciottdata( )** functions. For more information, see Section 4.1.10, "Setting Up Silence Compressed Streaming", on page 34 and Section 5.1.5, "Silence Compressed Streaming", on page 39.

1.9 Streaming to the TDM Bus

You can choose to stream echo-cancelled data over the TDM bus to another device, regardless of whether that device supports CSP. This feature allows you to share the CSP resource among several devices.

This feature is disabled by default. To enable this feature, you set specific parameters in a supported media load file (DM3 boards) or configuration file (Springware boards). An extra time slot is then assigned to the channel.

For more information, see Section 4.1.2, "Reserving Extra Time Slots for Streaming to TDM Bus", on page 32.

Streaming to the TDM bus is supported on select DM3 boards only, at reduced density. The media load which is downloaded when you start the board determines whether this feature is available. For information on media loads, see the appropriate Configuration Guide. For information on board support for this feature, see the Release Guide for your system release.

1.10 Supported Data Formats

Information on supported data formats is provided in the following topics:

- Supported Data Formats on DM3 Boards
- Supported Data Formats on Springware Boards

1.10.1 Supported Data Formats on DM3 Boards

The CSP software supports the following encoding algorithms, sampling rates and sampling sizes for *streaming* on CSP channels using **ec_stream( )** or **ec_reciottdata( )**:

- G.711 mu-law PCM, 8 kHz sampling rate, 8-bit resolution (64 Kbps)
- G.711 A-law PCM, 8 kHz sampling rate, 8-bit resolution (64 Kbps)
- Linear PCM, 8 kHz sampling rate, 16-bit resolution little Endian and big Endian format (128 kbps)

Notes: ***1.*** If you stream a file using an unsupported data format, CSP features are not available.

2. When using **ec_stream( )**, VOX file format is supported only. Both WAVE and VOX file formats are supported for **ec_reciottdata( )** except in the case of the linear PCM coder which supports VOX file format only.

The CSP software supports the following encoding algorithms, sampling rates and sampling sizes for *playing* files during a CSP streaming session (WAVE or VOX file format):

- G.711 mu-law PCM, 8 kHz sampling rate, 8-bit resolution (64 kbps)
- G.711 A-law PCM, 8 kHz sampling rate, 8-bit resolution (64 kbps)

Notes: ***1.*** If you play a file using an unsupported data format while CSP streaming is occurring, CSP features are not available.

2. While CSP streaming is occurring on a channel, recording on this same voice channel is not supported.

1.10.2 Supported Data Formats on Springware Boards

The CSP software supports the following encoding algorithms, sampling rates and sampling sizes for play and recording/streaming on CSP channels:

- G.711 mu-law PCM, 8 kHz sampling rate, 8-bit resolution (64 kbps)
- G.711 A-law PCM, 8 kHz sampling rate, 8-bit resolution (64 kbps)
- Linear PCM, 8 kHz sampling rate, 8-bit resolution (64 kbps)
- OKI ADPCM, 8 kHz sampling rate, 4-bit resolution (32 kbps)

Notes: ***1.*** If you play or record a file using an unsupported data format, CSP features are not available.

2. When using **ec_stream( )**, VOX file format is supported only. Both WAVE and VOX file formats are supported for **ec_reciottdata( )**.

3. On Springware boards, we recommend that you use the same data format for play and recording/streaming.

4. On Springware boards, after completion of the CSP section of your application, set DXCH_EC_TAP_LENGTH to the default value of 48. This allows you to use other data formats that are not supported by CSP.

1.11 Comparison with Existing Features

Table 1 shows a brief comparison of the key differences between the ECR software (also known as HDEC, High-Density Echo Cancellation), the Barge-In package and the CSP software.

Table 1. Feature Comparison

Feature	Available in Barge-in package	Available in ECR	Available in CSP
Full-duplex channel (simultaneous play and record)	No	No	Yes
Hardware support	D/21H, D/41H and D/41ESC	high-density boards only	JCT-series boards and DM3 boards
Data format for play and record/stream	PCM 8 kHz (no AGC)	PCM 8 kHz (no AGC)	Several data formats available. For details, see Section 1.10, "Supported Data Formats", on page 21.
Echo cancellation tap length	6 ms (48 taps) used only for signal detection	Up to 16 ms (128 taps)	Up to 16 ms (128 taps) (Springware) Up to 64 ms (512 taps) (select DM3 boards)
Integrated barge-in and echo-cancelled streaming	No	No	Yes
Pre-speech buffers	256 bytes (can store up to 125 ms)	No	Can store up to 250 ms of 8 kHz PCM data
Echo-cancelled record initiated by voice activity detector	Yes	No	Yes
API	Part of voice library	Part of voice library	Separate CSP library

1.12 CSP Support on Springware Versus DM3 Boards

Table 2 summarizes the differences that exist when running CSP on DM3 boards versus on Springware boards. For a complete list of hardware that supports CSP, see the Release Guide and Release Update for your system release.

For more information on parameters and functions mentioned in Table 2, see the **ec_setparm( )** function description and other function descriptions in the *Continuous Speech Processing API Library Reference*.

Table 2. CSP Support on Springware Boards and DM3 Boards

Feature/Functionality	Springware Boards	DM3 Boards	Notes
Firmware buffer size (DXBD_RXBUFSIZE and DXBD_TXBUFSIZE)	Supported	Not supported	
Driver or transfer buffer size (ECCH_XFERBUFFERSIZE)	Supported	Supported	On DM3 boards, this parameter handles both firmware and driver buffers.
Echo cancellation tap length (DXCH_EC_TAP_LENGTH)	48 to 128 taps	Varies	On some DM3 boards, the tap length is fixed at 128. On other DM3 boards, a tap length up to 512 is supported. The media load downloaded to the board determines the default and available tap lengths. For more information on media loads, see the appropriate Configuration Guide. For information on tap length support on DM3 boards, see the Release Guide for this system release.
Non-play speech threshold, trigger and window parameters (DXCH_SPEECHNONPLAYTHRESH, DXCH_SPEECHNONPLAYTRIGG, DXCH_SPEECHNONPLAYWINDOW)	Supported	Not supported	Non-play parameters are not used on DM3 boards.
Adaptation mode (ECCH_ADAPTMODE)	Supported	Not supported	
Zero-crossing mode (ECCH_SVAD)	Not supported	Supported	
Automatic gain control (AGC)	Supported	Not supported	AGC must be turned off in all ASR applications.
Echo canceller convergence notification	Not supported	Supported	
Termination conditions in DV_TPT data structure	The following conditions are supported only: DX_MAXTIME, DX_MAXSIL, DX_MAXNOSIL	All conditions are supported EXCEPT for the following: DX_LCOFF, DX_PMON and DX_PMOFF	Applies when using ec_ functions. In CSP, DV_TPT terminating conditions are edge-sensitive.
Silence compressed streaming	Not supported	Supported	
Call control through Global Call library	Supported	Supported	
Call control through the Voice library	Supported	Not supported	

Table 2. CSP Support on Springware Boards and DM3 Boards (Continued)

Feature/Functionality	Springware Boards	DM3 Boards	Notes
Flexible routing configuration	Not relevant	Supported	For a definition of this term, see the Glossary.
Fixed routing configuration	Not relevant	Not supported	For a definition of this term, see the Glossary.

Event Handling

2

This chapter briefly describes events and event handling in Continuous Speech Processing (CSP) software.

If your application is running in asynchronous mode, you may need to use a Standard Runtime Library (SRL) function to collect events being sent from the firmware, depending on the programming model in use. For more information, see the *Standard Runtime Library API Programming Guide*.

For a list of events that can be returned by the CSP software, see the *Continuous Speech Processing API Library Reference*.

Error Handling 3

This chapter briefly describes how to handle errors that occur when running Continuous Speech Processing (CSP) applications.

CSP library functions return a value to indicate success or failure of the function.

- To indicate success, the library returns a value of zero or a non-negative number.
- To indicate failure, the library returns a value of -1.

Error codes that may be returned by a function are described with each function description in the *Continuous Speech Processing API Library Reference*.

If a library function fails, call the standard attribute function **ATDV_LASTERR( )** to return the error code and **ATDV_ERRMSGP( )** to return a string describing the error. These functions are described in the *Standard Runtime Library API Library Reference*.

If **ATDV_LASTERR( )** returns the error EDX_SYSTEM, a system error has occurred. In Linux*, check the global variable **errno** contained in *errno.h*. In Windows*, use **dx_fileerrno( )** to obtain the system error value. For a list of possible system error values, see the **dx_fileerrno( )** function description in the *Voice API Library Reference*.

Application Development Guidelines

4

This chapter provides programming guidelines when developing Continuous Speech Processing (CSP) applications. Topics include:

- Programming Considerations 31
- Interoperability Considerations for Springware Boards 35

4.1 Programming Considerations

The following topics discuss programming considerations for CSP:

- Introduction
- Reserving Extra Time Slots for Streaming to TDM Bus
- Opening a Voice Channel
- Assigning Time Slots
- Configuring the CSP Device Channel Using ec_setparm( )
- Configuring the CSP Device Channel Using dx_setparm( )
- Setting Up VAD Event Notification
- Setting Up EC Convergence Event Notification
- Setting Up Streaming or Recording
- Setting Up Silence Compressed Streaming
- Playing a Prompt
- Collecting Events
- Performing Voice Processing
- Cleaning Up

4.1.1 Introduction

The topics for programming considerations are listed in the order in which you would perform the activities.

Details on CSP functions mentioned in this section can be found in the *Continuous Speech Processing API Library Reference*.

By convention, CSP-specific functions begin with **ec_**, such as **ec_stream( )**. Voice-specific functions begin with **dx_**, such as **dx_play( )**. Functions that are part of the Standard Runtime Library begin with **sr_**, such as **sr_waitevt( )**.

4.1.2 Reserving Extra Time Slots for Streaming to TDM Bus

If you wish to send echo-cancelled data to another device in your system over a time division multiplexing bus (TDM bus), you must reserve an extra time slot for your CSP-capable channel.

On DM3 boards, you configure this time slot at initialization time by modifying parameters in a supported media load file (firmware file). The parameter number is 0x2b12 and is called EC Streaming to TDM Bus. Once the media load file is configured to support streaming to the TDM bus, a time slot is set aside for this activity. The **ec_getxmitslot( )** becomes valid for the channel, and streaming to the TDM bus will be supported by **ec_stream( )** and **ec_reciottdata( )**. Use **ec_getxmitslot( )** to retrieve the number of the time slot which will transmit echo-cancelled data over the TDM bus.

Streaming to the TDM bus is available on select DM3 boards only, with reduced density. For a list of boards that support this feature, see the Release Guide for the system release you are using. For information on media loads and parameters for streaming to the TDM bus, see the appropriate Configuration Guide.

On Springware boards in Linux*, you configure this time slot at initialization time in *dialogic.cfg*. On Springware boards in Windows*, you configure this time slot at initialization time in the configuration manager (DCM), using the CSPExtraTimeSlot parameter under the **Misc** tab. See the appropriate Configuration Guide and DCM online help for more information.

To disable a previously enabled channel, you must turn the feature off by editing the media load file or configuration file, and redownload the board.

By default, no extra time slots are configured. It is assumed that the echo-cancelled data is sent to the host application, either recorded to a file for storage or streamed to memory for use by the application.

4.1.3 Opening a Voice Channel

You must first open a voice device to obtain the channel device handle before you can perform any operation on the device. If you intend to modify board-level parameters, open a voice board. Open a network channel if applicable. See the function reference in the *Voice API Library Reference* for more information.

On DM3 boards, the flexible routing configuration is supported (for a definition, see the Glossary). On DM3 boards, you must use Global Call functions for call control. On DM3 network devices, use Global Call functions to open and close device handles. For more information, see the Global Call documentation set.

4.1.4 Assigning Time Slots

If your system configuration uses the SCbus or CT Bus, perform time slot routing for SCbus or CT Bus. See the *Voice API Library Reference* for more information.

4.1.5 Configuring the CSP Device Channel Using ec_setparm( )

Configure your CSP device channel to suit your specific purpose using **ec_setparm( )**.

Several parameters are available to define values such as:

- tap length of echo canceller (48 taps by default for Springware boards; 128 or 512 taps by default for DM3, depending on the media load on the DM3 board)

  For information about DM3 board support for tap length, see the Release Guide. For information on media loads, see the appropriate Configuration Guide.
- voice-activated record (on by default)
- barge-in (off by default)
- non-linear processing (NLP, on by default; must be turned off for ASR applications)
- silence compressed streaming (off by default)
- other VAD parameters such as the threshold of energy that is considered significant during prompt play and after the prompt has terminated

For more information, see Section 5.1, "Voice Activity Detector Operating Modes", on page 37, Chapter 6, "Buffers and Data Flow", Section 1.12, "CSP Support on Springware Versus DM3 Boards", on page 23, and the *Continuous Speech Processing API Library Reference*.

Note: On Springware boards, after completion of the CSP section of your application, set DXCH_EC_TAP_LENGTH to the default value of 48. This allows you to use other data formats that are not supported by CSP.

4.1.6 Configuring the CSP Device Channel Using dx_setparm( )

If desired, configure your CSP device channel using **dx_setparm( )**. See the function reference description in the *Voice API Library Reference* for more information on this function.

4.1.7 Setting Up VAD Event Notification

If desired, specify that the voice activity detector (VAD) send notification when audio energy is detected. To do so, set up a VAD event mask, DM_VADEVTS, to be passed from the firmware to the application using **dx_setevtmsk( )**. See Section 5.1.2, "Sending a VAD Event to the Host Application", on page 38 for more information.

4.1.8 Setting Up EC Convergence Event Notification

On Springware boards, this feature is not supported.

If desired, specify that the board send notification to the host when the incoming signal is converged (that is, echo-cancelled). To do so, set the DM_CONVERGED event mask for the CSP device channel using **dx_setevtmsk( )**. The notification is passed from the firmware to the application as a TEC_CONVERGED event. See Chapter 7, "Echo Canceller Convergence Notification" for more information.

4.1.9 Setting Up Streaming or Recording

If desired, set up the application to process the incoming signal or speech utterance.

Call **ec_reciottdata( )** to stream echo-cancelled data to a file or to a memory buffer (data is available to the application at the completion of the record activity) or **ec_stream( )** to stream echo-cancelled data to the application as it is received. Note that if voice-activated record is turned on (ECCH_VADINITIATED in **ec_setparm( )**), streaming only occurs after the voice activity detector detects significant energy. If voice-activated record is turned off, streaming occurs at all times.

For more information on the voice activity detector, see Chapter 5, "Using the Voice Activity Detector". For more information on functions and parameters, see the function reference information in the *Continuous Speech Processing API Library Reference*.

For Springware boards, you must turn off Automatic Gain Control (AGC) in your ASR application using MD_NOGAIN in the mode parameter. For DM3 boards, AGC is not available during an echo-cancelled recording or streaming.

The TEC_STREAM event is the completion event returned.

Note: Voice library record functions such as **dx_reciottdata( ) cannot** be used simultaneously with **ec_reciottdata( )** or **ec_stream( )** on a CSP channel.

4.1.10 Setting Up Silence Compressed Streaming

If desired, set up the application to perform silence compressed streaming. If this feature is turned on (ECCH_SILENCECOMPRESS in **ec_setparm( )**), silence periods in the speech utterance are significantly reduced before the utterance is streamed. ECCH_SILENCECOMPRESS is a channel-level parameter and must be set channel by channel.

An application can retrieve time synchronous information by calling the **ec_getblkinfo( )** function from within its write callback. The application passes the EC_BLK_INFO structure, which is filled with details regarding the block for which the callback is executed. These details include a time stamp, size of the data buffer, block type (initial, middle or trailing block), and a flag to indicate the last block sent in the stream. For more information on functions, parameters, and data structures, see the *Continuous Speech Processing API Library Reference*.

4.1.11 Playing a Prompt

Set up your application to play a prompt. For more information on functions used to play a prompt, such as **dx_play( )** and **dx_playiottdata( )**, see the function reference section in the *Voice API Library Reference*. For a list of data formats supported for CSP, see Section 1.10, "Supported Data Formats", on page 21.

4.1.12 Collecting Events

If your application is running in asynchronous mode, you may need to use **sr_waitevt( )**, **sr_enbhdlr( )** or other SRL function to collect an event code being sent from the firmware, depending on the programming model in use. For more information, see the *Standard Runtime Library API Library Reference*.

Depending on how you configure the CSP device channel, you will get different events. Handle each event appropriately. For example, the TDX_BARGEIN event indicates that play was halted by the VAD. The TEC_VAD event indicates that speech activity was detected. For more information on event codes, see the *Continuous Speech Processing API Library Reference*.

4.1.13 Performing Voice Processing

Continue with voice processing in your application. For example, you will want to act on the data that has been recognized, such as routing the call, prompting for another response, or sending the caller to voice-mail.

4.1.14 Cleaning Up

At the end of the application, unroute time slots and close the open channels.

4.2 Interoperability Considerations for Springware Boards

In most cases, the CSP software can operate with other existing features. To ensure proper operation on Springware boards, however, review compatibility issues for the following areas:

- Transaction Record
- DSP-Based Fax
- ISDN

4.2.1 Transaction Record

Transaction record enables an application to record a two-way transaction or conversation. It takes two inputs from the TDM bus to perform this record. For more information on this feature, see the *Voice API Library Reference*.

On Springware boards, at any given time, you can use a channel for either a CSP operation or a transaction record operation; both features cannot be used simultaneously on a channel.

4.2.2 DSP-Based Fax

On Springware boards, DSP-based fax cannot be used together with CSP on the same board.

4.2.3 ISDN

On Springware boards, ISDN cannot be used together with CSP on the same span.

Certain Dual Span products allow for ISDN on the first span and CSP on the second span. See the Release Guide and Release Update for the system release you are using for more information on hardware support. See the Technical Support Web site at *http://resource.intel.com/telecom/support/tnotes/tnbyos/unix/tn212.htm* for a Tech Note on how to route a CSP voice resource to an ISDN network resource.

Alternatively, you can use ISDN and CSP together in a system by installing a network interface product (DTI/SC) in your system in addition to the voice product (such as a D/240JCT-T1).

On DM3 boards, ISDN can be used together with CSP on the same span.

Using the Voice Activity Detector 5

This chapter provides details on the voice activity detector (VAD). The following topics are discussed:

- Voice Activity Detector Operating Modes. 37
- VAD Operation 43
- Fine-Tuning VAD Performance. 44

5.1 Voice Activity Detector Operating Modes

The voice activity detector operating modes are described in the following topics:

- Overview
- Sending a VAD Event to the Host Application
- Stopping Play When Speech is Detected (Barge-In)
- Voice-Activated or Constant Recording
- Silence Compressed Streaming
- Sample VAD Scenarios

5.1.1 Overview

The voice activity detector (VAD) is a component in the CSP software that examines a caller's incoming signal and determines if the signal contains significant energy and is likely to be speech rather than a click, for example.

When the voice activity detector (VAD) detects audio energy, you can specify that it act on this energy in one or more of the following ways:

- send a VAD event to the host application when speech is detected
- stop play when speech is detected (barge-in) or allow play to continue
- record/stream data to the host application only after energy is detected (voice-activated record/stream) or constantly record/stream
- record/stream data to the host application with silence periods removed (silence compressed streaming)

Table 3 summarizes the types of operation, the modes and the settings required to enable them. These operations are described in more detail in the subsections that follow.

Table 3. VAD and Barge-In Operating Modes

Operation	Modes	Required Setting
send VAD event to host when speech is detected	on	DM_VADEVTS event bit mask set in **dx_setevtmsk( )**
	off (default)	DM_VADEVTS bit mask not set
stop play when speech is detected	on	DXCH_BARGEIN = 1 in **ec_setparm( )** On Springware boards, to use barge-in, you must also set ECCH_VADINITIATED to 1.
	off (default)	DXCH_BARGEIN = 0
record/stream data	constant	ECCH_VADINITIATED = 0 in **ec_setparm( )** Use **ec_reciottdata( )** or **ec_stream( )** to record echo-cancelled data.
	voice activated (default)	ECCH_VADINITIATED = 1 in **ec_setparm( )** Use **ec_reciottdata( )** or **ec_stream( )** to record echo-cancelled data.
silence compressed streaming	on	ECCH_SILENCECOMPRESS = 1 in **ec_setparm( )** You must also set ECCH_VADINITIATED to 1. Use **ec_reciottdata( )** or **ec_stream( )** to record echo-cancelled data.
	off (default)	ECCH_SILENCECOMPRESS = 0 in **ec_setparm( )** Use **ec_reciottdata( )** or **ec_stream( )** to record echo-cancelled data.

5.1.2 Sending a VAD Event to the Host Application

The DM_VADEVTS parameter in **dx_setevtmsk**( ) controls whether the firmware sends a VAD event to the host application every time energy is detected. (This is also known as voice event signaling.) This may be useful when you want to halt the playing of the prompt based on other criteria, such as after the caller utters a valid vocabulary word. CSP then sends the pre-speech buffers followed by the speech buffers in real time to the application.

The event sent to the host application is called TEC_VAD. Use **sr_waitevt**( ), **sr_enbhdlr**( ), or other SRL function to collect an event code, depending on the programming model in use. For more information on SRL functions, see the *Standard Runtime Library API Library Reference*.

This event is only generated when data is being recorded or streamed. If VAD detects energy and a record or stream activity is not occurring, a VAD event is not sent to the host application.

By default, no event is sent to the host application.

For more information on using DM_VADEVTS, see Section 5.1.6, “Sample VAD Scenarios”, on page 40.

Note: The DM_VADEVTS parameter does not need to be set in order for barge-in and recording/streaming capability to be available.

5.1.3 Stopping Play When Speech is Detected (Barge-In)

The DXCH_BARGEIN parameter in **ec_setparm( )** controls whether barge-in is executed in the application. This means that when speech is detected, the prompt play is automatically halted. This allows the caller to speak without distraction.

The event sent to the host application is called TDX_BARGEIN and can be returned by calling **sr_waitevt( )**.

The default value is barge-in disabled.

For more information on using DXCH_BARGEIN, see Section 5.1.6, “Sample VAD Scenarios”, on page 40.

Notes:
1. If desired, you can use the DXCH_BARGEINONLY parameter in **ec_setparm( )** to generate the TDX_PLAY event in addition to the TDX_BARGEIN event when a barge-in condition occurs.

2. Streaming or recording starts on voice detection or on DTMF detection.

5.1.4 Voice-Activated or Constant Recording

The ECCH_VADINITIATED parameter in **ec_setparm( )** controls whether audio energy is transmitted only after the VAD detects speech energy or at all times.

To implement voice-activated recording or streaming, turn this parameter on. This is the default setting. To implement constant recording or streaming, turn this parameter off. Speech is transmitted using the **ec_reciottdata( )** or **ec_stream( )** functions.

Voice-activated recording is particularly useful for reducing CPU loading by only streaming data when audio energy is present.

For more information on using ECCH_VADINITIATED, see Section 5.1.6, “Sample VAD Scenarios”, on page 40.

5.1.5 Silence Compressed Streaming

On Springware boards, silence compressed streaming is not supported.

The ECCH_SILENCECOMPRESS parameter in **ec_setparm( )** controls whether audio energy is transmitted with silence periods removed or not. By default, this feature is disabled.

Silence compressed streaming uses default parameter values that are set in a configuration file (CONFIG, PCD, FCD files). These values are downloaded to the board when it is started. SCS parameters in the configuration file include:

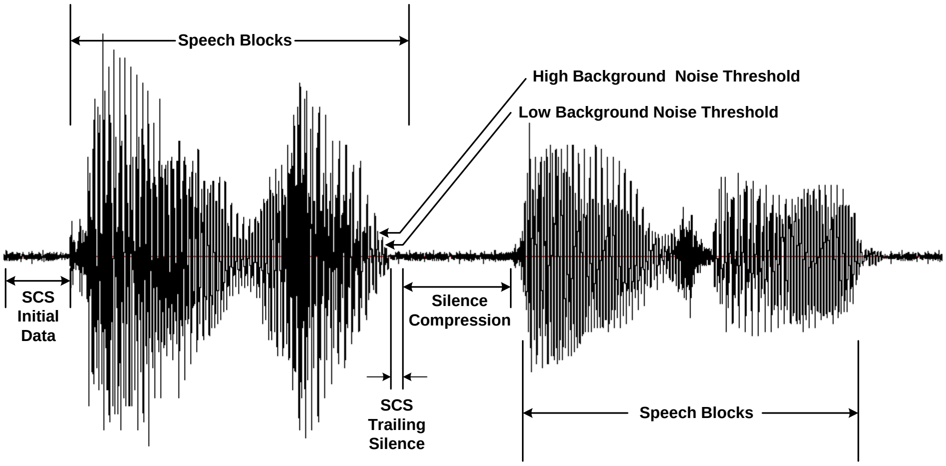
- initial data (default value is 0)
- trailing silence (default value is 200 in 10 msec units)
- speech probability threshold
- silence probability threshold

- low background noise threshold
- high background noise threshold

If the default settings in the FCD files are not appropriate for your configuration, you can modify the FCD file parameters using the CONFIG file and the *fcdgen* utility. Modifications can be made at any time prior to starting the system. For each FCD file to be modified, the procedure includes editing the CONFIG file and generating the FCD file. After a new FCD is generated, you must redownload the board for the new values to take effect. For configuration details, see the Configuration Guide for your product or product family.

Figure 4 illustrates a sample speech pattern and how the silence compressed streaming parameters (which reside in the configuration file) are used.

Figure 4. Example of Silence Compressed Streaming Operation



The algorithm used to determine the probability of speech in silence compressed streaming (SCS) is based on the voice activity detector (VAD) algorithm.

5.1.6 Sample VAD Scenarios

You can set VAD parameters and use the record/stream functions in different ways to achieve a specific purpose. The scenarios in Table 4 are a few ways in which you might set certain parameters and record/stream functions for use in your application.

Note: This section does not discuss all available parameters for use with Continuous Speech Processing (CSP). Other parameters that may need to be set include DXCH_EC_TAP_LENGTH, ECCH_ECHOCANCELLER (on by default), and ECCH_XFERBUFFERSIZE, among others. For information on all parameters, see the **ec_setparm( )** function description in the *Continuous Speech Processing API Library Reference*.

Table 4. Sample VAD Scenarios

Function/Parameter	Scenario				
	A	B	C	D	E
ec_stream( ) or **ec_reciottdata( )**	✓	✓	✓	✓	✓
ECCH_VADINITIATED	✓	✓	×	×	✓
DXCH_BARGEIN	✓	✓	×	×	†
DM_VADEVTS	✓	×	×	✓	‡
ECCH_SILENCECOMPRESS	×	×	×	×	✓

✓ Means that the function is initiated or the parameter is turned on.
× Means that the parameter is turned off.
† You can set DXCH_BARGEIN to on or off in scenario E.
‡ It is recommended that you set DM_VADEVTS off in scenario E.

The following topics provide more information on the sample scenarios:

- Scenario A: Streaming with VAD
- Scenario B: Streaming with VAD and without VAD Event
- Scenario C: Streaming without VAD (Constant Streaming)
- Scenario D: Constant Streaming with VAD Event
- Scenario E: Silence Compressed Streaming

Scenario A: Streaming with VAD

In scenario A shown in Table 4, “Sample VAD Scenarios”, on page 41, data is transmitted only after the VAD detects speech energy. The following takes place:

- The **ec_stream( )** or **ec_reciottdata( )** function is initiated in the application.
- After the VAD detects energy that meets certain predefined criteria (ECCH_VADINITIATED = 1):
  - the firmware sends a VAD event notification to the host application (DM_VADEVTS bit is turned on).
  - the prompt is terminated (DXCH_BARGEIN = 1).
  - data streaming or recording begins.

Note: On Springware boards, be aware that the timer on the DX_MAXTIME terminating condition (DV_TPT structure) begins when you initiate the **ec_stream( )** or **ec_reciottdata( )** function.

Scenario B: Streaming with VAD and without VAD Event

Scenario B, shown in Table 4, “Sample VAD Scenarios”, on page 41, is a variation of Scenario A. In this scenario, the following takes place:

- The **ec_stream( )** or **ec_reciottdata( )** function is initiated in the application.
- After the VAD detects energy that meets certain predefined criteria (ECCH_VADINITIATED = 1):

- the prompt is terminated (DXCH_BARGEIN = 1).
- data streaming or recording begins.

The difference between Scenario A and B is that in Scenario B, the VAD event bit (DM_VADEVTS) is turned off. The host application does not receive notification when the VAD detects energy. Note that the VAD event bit does not need to be set in order for barge-in and recording/streaming capability to be available.

Note: On Springware boards, be aware that the timer on the DX_MAXTIME terminating condition (DV_TPT structure) begins when you initiate the **ec_stream( )** or **ec_reciottdata( )** function.

Scenario C: Streaming without VAD (Constant Streaming)

In scenario C shown in Table 4, "Sample VAD Scenarios", on page 41, data is streamed at all times. The VAD, barge-in, and VAD event bit (voice event signaling) are turned off. The settings are as follows:

- The **ec_stream( )** or **ec_reciottdata( )** function is initiated in the application, and data streaming or recording begins.
- Voice-activated streaming/recording is not turned on (ECCH_VADINITIATED = 0).
- The firmware does not send a VAD event notification to the host application (DM_VADEVTS bit turned off).
- The prompt is not terminated when energy is detected (DXCH_BARGEIN = 0).

Scenario D: Constant Streaming with VAD Event

In scenario D shown in Table 4, "Sample VAD Scenarios", on page 41, data is streamed at all times. When energy is detected, the firmware sends a VAD event notification to the host application. The VAD and barge-in are turned off. The settings are as follows:

- The **ec_stream( )** or **ec_reciottdata( )** function is initiated in the application, and data streaming or recording begins.
- Voice-activated streaming/recording is not turned on (ECCH_VADINITIATED = 0).
- The firmware sends a VAD event notification to the host application (DM_VADEVTS bit turned on).
- The prompt is not terminated when energy is detected (DXCH_BARGEIN = 0).

Scenario E: Silence Compressed Streaming

In scenario E shown in Table 4, "Sample VAD Scenarios", on page 41, data is transmitted only after the VAD detects speech energy. In addition, silence periods are removed from this data. The following takes place:

- The **ec_stream( )** or **ec_reciottdata( )** function is initiated in the application.
- After the VAD detects energy that meets certain predefined criteria (ECCH_VADINITIATED = 1), data streaming or recording begins.
- With silence compressed streaming enabled (ECCH_SILENCECOMPRESS = 1), periods of silence that meet predefined criteria are compressed and not streamed to the host application.

Notes: *1.* You can set DXCH_BARGEIN on or off in this scenario. If turned on, the prompt is terminated before streaming begins.

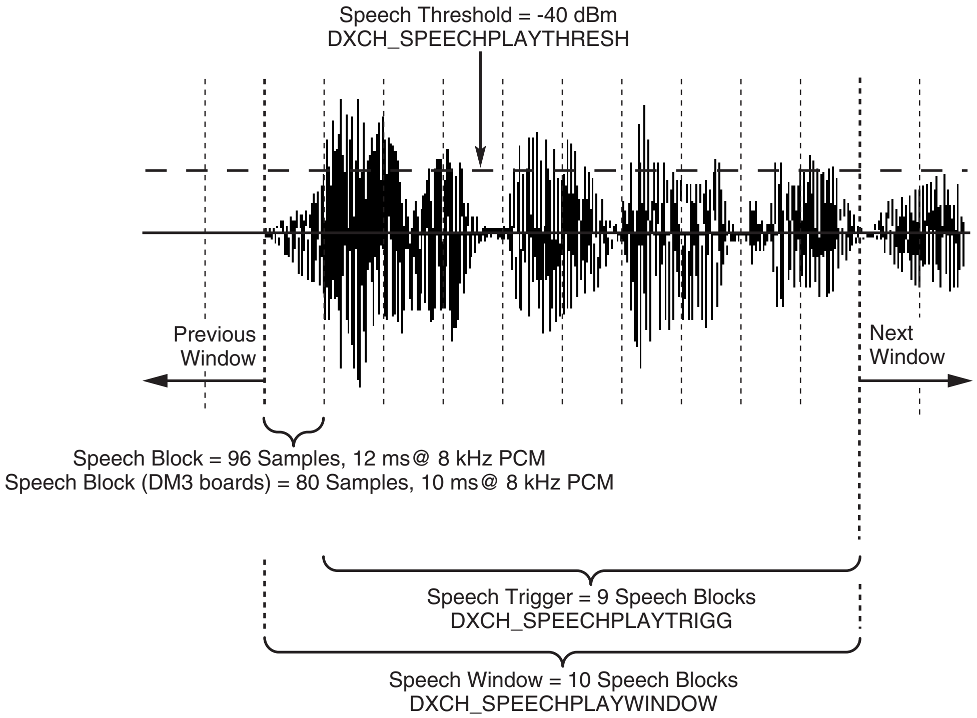
2. It is recommended that you set DM_VADEVTS off when using silence compressed streaming to reduce the number of TEC_VAD events to be handled by the application. The callback function being called is your indicator that the VAD has triggered.

5.2 VAD Operation

The best way to describe the VAD operation is by example. Figure 5 shows sample speech patterns and the voice activity detector's operation on this speech.

For a description of the parameters, see the **ec_setparm( )** function description in the *Continuous Speech Processing API Library Reference*.

Figure 5. Example of Voice Activity Detector (VAD) Operation


Speech Threshold = -40 dBm
DXCH_SPEECHPLAYTHRESH
Previous Window
Next Window
Speech Block = 96 Samples, 12 ms@ 8 kHz PCM
Speech Block (DM3 boards) = 80 Samples, 10 ms@ 8 kHz PCM
Speech Trigger = 9 Speech Blocks
DXCH_SPEECHPLAYTRIGG
Speech Window = 10 Speech Blocks
DXCH_SPEECHPLAYWINDOW

This example illustrates the following:

- The **speech window** consists of 10 speech blocks (the default value). You can adjust the size of the speech window as needed for play and non-play situations.
  DXCH_SPEECHPLAYWINDOW = 10

- On DM3 boards, each **speech block** in the speech window consists of 80 samples; each block is 10 milliseconds in length at 8 kHz PCM. This value is fixed and cannot be modified.

  On Springware boards, each **speech block** in the speech window consists of 96 samples; each block is 12 milliseconds in length at 8 kHz PCM. This value is fixed and cannot be modified.
- The **speech threshold** is -40 dBm and the **speech trigger** is 9 speech blocks.
  DXCH_SPEECHPLAYTHRESH = -40 dBm
  DXCH_SPEECHPLAYTRIGG = 9 (speech blocks)
- Each speech block is examined by the VAD to see whether the speech energy exceeds the speech threshold value of -40 dBm. If the speech energy exceeds or is equal to -40 dBm, then that speech block is assigned the value 1. If the speech energy is less than -40 dBm, the speech block is assigned the value 0.
- In this example, 9 out of 10 speech blocks in the speech window register speech energy greater than the speech threshold (as indicated by the value 1). Thus, barge-in occurs.
- When barge-in occurs, the VAD sends the voice data on to the host application or speech recognition engine for further voice analysis.

5.3 Fine-Tuning VAD Performance

The following topics discuss how to fine-tune VAD performance depending on the category of boards:

- Overview
- Fine-Tuning VAD Performance on Springware Boards
- Fine-Tuning VAD Performance on DM3 Boards

5.3.1 Overview

The voice activity detector (VAD) uses a different set of algorithms on Springware boards versus DM3 boards. On Springware boards, energy threshold is used to perform voice activity detection, while on DM3 boards, speech statistics can be used in addition to energy threshold.

5.3.2 Fine-Tuning VAD Performance on Springware Boards

On Springware boards, several of the defines used for VAD can be divided into two groups. You can adjust the values of these two sets of defines to fine-tune the performance of the VAD.

- Defines that take effect during the playing of a prompt:
  - DXCH_SPEECHPLAYTHRESH
  - DXCH_SPEECHPLAYTRIGG
  - DXCH_SPEECHPLAYWINDOW
- Defines that take effect after play has completed:
  - DXCH_SPEECHNONPLAYTHRESH
  - DXCH_SPEECHNONPLAYTRIGG
  - DXCH_SPEECHNONPLAYWINDOW

During play, echo or noise often exists on the channel. If you find that your application triggers on echo or background noise, you may want to:

- *increase* speech trigger (DXCH_SPEECHPLAYTRIGG) so that the incoming speech energy is present for a greater duration of the speech window.
- *increase* speech window (DXCH_SPEECHPLAYWINDOW) thereby requiring the incoming speech energy to be present for a longer time period. This should protect against false triggers due to noise spikes or other short duration (non-speech) noises.
- *increase* speech threshold (DXCH_SPEECHPLAYTHRESH) so that the energy level of incoming speech required to trigger the VAD is relatively higher during play and relatively lower when the prompt play completes (DXCH_SPEECHNONPLAYTHRESH).

Note: To reduce sensitivity to background noise, you must perform these actions in certain combinations only: increase speech trigger alone; increase speech threshold alone; increase speech trigger and speech window together; increase speech trigger, speech window, and speech threshold together. Increasing speech window alone will not help reduce sensitivity to background noise.

After the prompt completes, residual echo and VAD sensitivity to prompt-related false triggers should be reduced. Hence, you may set speech trigger (DXCH_SPEECHNONPLAYTRIGG), speech window (DXCH_SPEECHNONPLAYWINDOW) and speech threshold (DXCH_SPEECHNONPLAYTHRESH) to lower values allowing for easier speech detection. Doing so improves rejection of false triggers and VAD sensitivity.

For more information on parameters, see the **ec_setparm()** function description in the *Continuous Speech Processing API Library Reference*.

5.3.3 Fine-Tuning VAD Performance on DM3 Boards

The VAD algorithm on DM3 boards performs sophisticated calculations on each input signal to determine whether the signal is speech or not. The probability of speech is computed based on the current energy level estimate and zero-crossing frequency. The calculations include a combination of long-term and short-term energy and zero-crossing based probabilities.

Thus, the adaptive nature of the VAD on DM3 boards reduces the need to fine-tune VAD parameters.

When using the **SVAD mode** on DM3 boards (ECCH_SVAD = 0), which is a combination of energy and zero-crossing based probability calculations, you can adjust the DXCH_SPEECHPLAYWINDOW and DXCH_SPEECHPLAYTRIGG parameters to fine-tune the performance of VAD. If you find that your application triggers on echo or background noise, you may want to:

- *increase* speech trigger (DXCH_SPEECHPLAYTRIGG) so that the incoming speech energy is present for a greater duration of the speech window.

- *increase* speech window (DXCH_SPEECHPLAYWINDOW) thereby requiring the incoming speech energy be present for a longer time period. This should protect against false triggers due to noise spikes or other short duration (non-speech) noises.

Note: To reduce sensitivity to background noise, you must perform these actions in certain combinations only: increase speech trigger alone, or increase speech trigger and speech window together to reduce sensitivity to background noise. Increasing speech window alone will not help reduce sensitivity to background noise.

When using the **energy-only mode** on DM3 boards (ECCH_SVAD = 1), you can adjust the DXCH_SPEECHPLAYTRIGG, DXCH_SPEECHPLAYWINDOW and DXCH_SPEECHPLAYTHRESH parameters to fine-tune the performance of VAD.

On DM3 boards, non-play parameters, DXCH_SPEECHNONPLAYTRIGG, DXCH_SPEECHNONPLAYWINDOW and DXCH_SPEECHNONPLAYTHRESH, are not supported.

For more information on all VAD parameters, see the **ec_setparm( )** function description in the *Continuous Speech Processing API Library Reference*.

Buffers and Data Flow 6

This chapter defines the various types of buffers and describes the data flow between the application and the firmware. The following topics are discussed:

- Types of Buffers. 47
- Data Flow. 49
- Buffer Usage Tips 52

6.1 Types of Buffers

A buffer is a temporary storage area for data transfer. The CSP software uses buffers to transfer data from the application to the firmware where echo cancellation is performed.

The size of a buffer affects real-time processing and latency as well as system performance. You must choose a buffer size carefully to maximize throughput and minimize system load.

For Springware boards, the CSP software uses the buffers shown in Table 5, “Types of Buffers Used in CSP (Springware Boards)”, on page 48. For DM3 boards, the CSP software uses the buffers shown in Figure 6, “Types of Buffers Used in CSP (DM3 Boards)”, on page 49.

See Figure 6, “Data Flow from Application to Firmware (Springware Boards)”, on page 50 and Figure 7, “Data Flow from Application to Firmware (DM3 Boards)”, on page 51 for an illustration of these buffers.

Table 5. Types of Buffers Used in CSP (Springware Boards)

Buffer name	Configurable	Parameter/Description
Driver buffers (**in Windows***)	Yes: from 128 bytes to 16 kbytes (in multiples of 128 bytes)	ECCH_XFERBUFFERSIZE Specifies the size of the host application buffers on the receive side of a CSP-capable channel. These buffers are allocated and tracked by the libraries. The default buffer size is 16 kbytes. The content of these buffers is sent to the user-defined callback function in **ec_stream( )**. The content is sent to a file or memory buffer when using **ec_reciottdata( )**.

Driver buffers
(**in Linux***)
	Yes (when used as described only):
	1 kbytes,
	2 kbytes,
	4 kbytes,
	8 kbytes,
	16 kbytes
		ECCH_XFERBUFFERSIZE
		Same description as driver buffers in Windows with the following limitation.
		By default, the amount of data passed to the user-defined callback function is fixed at 16 kbytes. You can only override this default per process by calling **ec_setparm( )** BEFORE opening a channel:
		`int size = 1024; /* or 2, 4, 8, 16 kbytes */`
		`…`
		`ec_setparm(SRL_DEVICE, ECCH_XFERBUFFERSIZE, &size)`
		Note: You must use SRL_DEVICE as the device name.

Firmware buffers (**in Windows**)	Yes: from 128 bytes to 512 bytes	DXBD_TXBUFSIZE and DXBD_RXBUFSIZE Specifies the size of the transmit (play) and receive (record) buffers in shared RAM. These buffers are used to transfer data between the firmware and the driver. To change firmware buffers from the default of 512 bytes, you must modify the *voice.prm* file. For more information, see the installation and configuration guide.
Firmware buffers (**in Linux**)	No: fixed at 512 bytes	DXBD_TXBUFSIZE and DXBD_RXBUFSIZE Specifies the size of the transmit (play) and receive (record) buffers in shared RAM. These buffers are used to transfer data between the firmware and the driver.
Pre-speech buffer	No: fixed at 250 ms	No parameter available.

For more information on these parameters, see DXBD_RXBUFSIZE, DXBD_TXBUFSIZE and ECCH_XFERBUFFERSIZE descriptions in **ec_setparm**( ) in the *Continuous Speech Processing API Library Reference*.

Table 6. Types of Buffers Used in CSP (DM3 Boards)

Buffer name	Configurable	Parameter/Description
Transfer buffers	Yes: from 320 bytes to 16 kbytes	ECCH_XFERBUFFERSIZE Specifies the size of the host application buffers on the receive side of a CSP-capable channel. These buffers are allocated and tracked by the libraries. The content of these buffers is sent to the user-defined callback function in **ec_stream( )**. The content is sent to a file or memory buffer when using **ec_reciottdata( )**. On DM3 boards, the size of the buffers sent from the firmware to the host is derived from the size of the transfer buffers. If the transfer buffer is less than or equal to 2 kbytes, then the firmware buffer is set to the same size as the transfer buffer. If the transfer buffer is greater than 2 kbytes, then the firmware buffer is set to 2 kbytes. The content of multiple firmware buffers is accumulated in the transfer buffer before being written to file or provided to the application callback function. The firmware buffer size cannot be greater than 2 kbytes.
Pre-speech buffer	No: 250 ms	Not configurable.

6.2 Data Flow

For Springware boards, Figure 6 depicts the data flow as data travels from the application to the firmware level. Buffers are also identified. For DM3 boards, see Figure 7.

These diagrams are intended to illustrate the concepts rather than the actual physical location of buffers. For more information on buffers, see Table 5, “Types of Buffers Used in CSP (Springware Boards)”, on page 48 and Table 6, “Types of Buffers Used in CSP (DM3 Boards)”, on page 49.

Figure 6. Data Flow from Application to Firmware (Springware Boards)

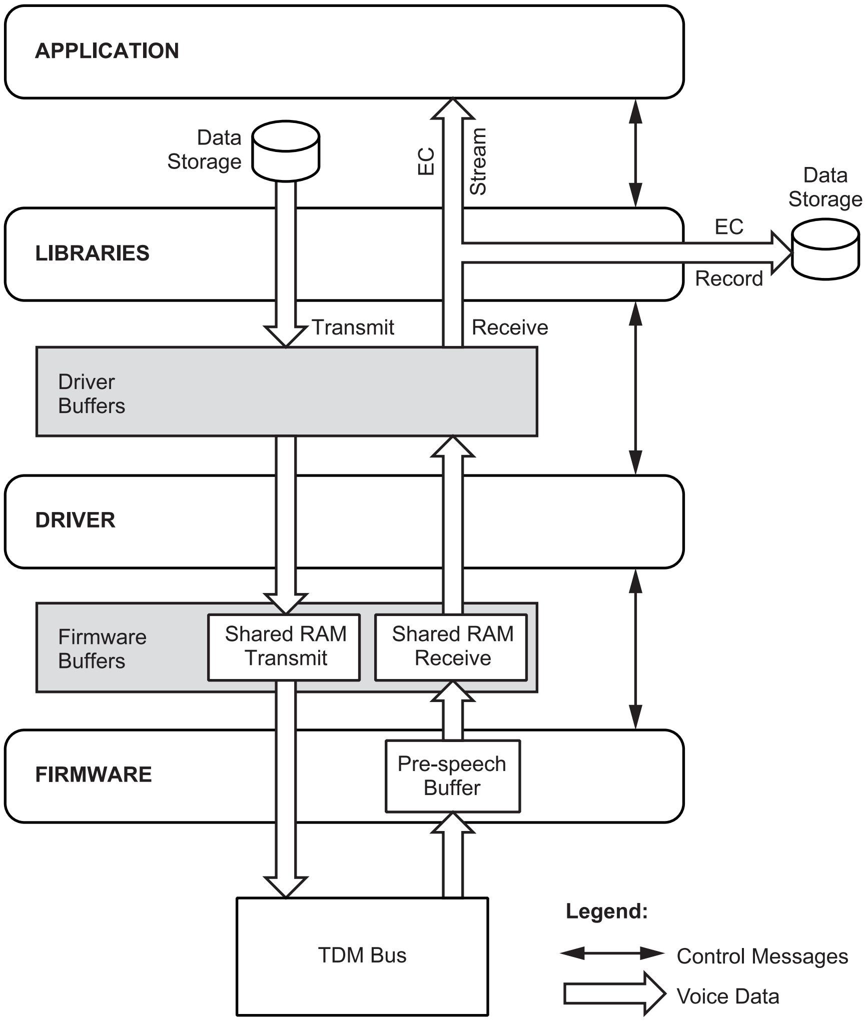
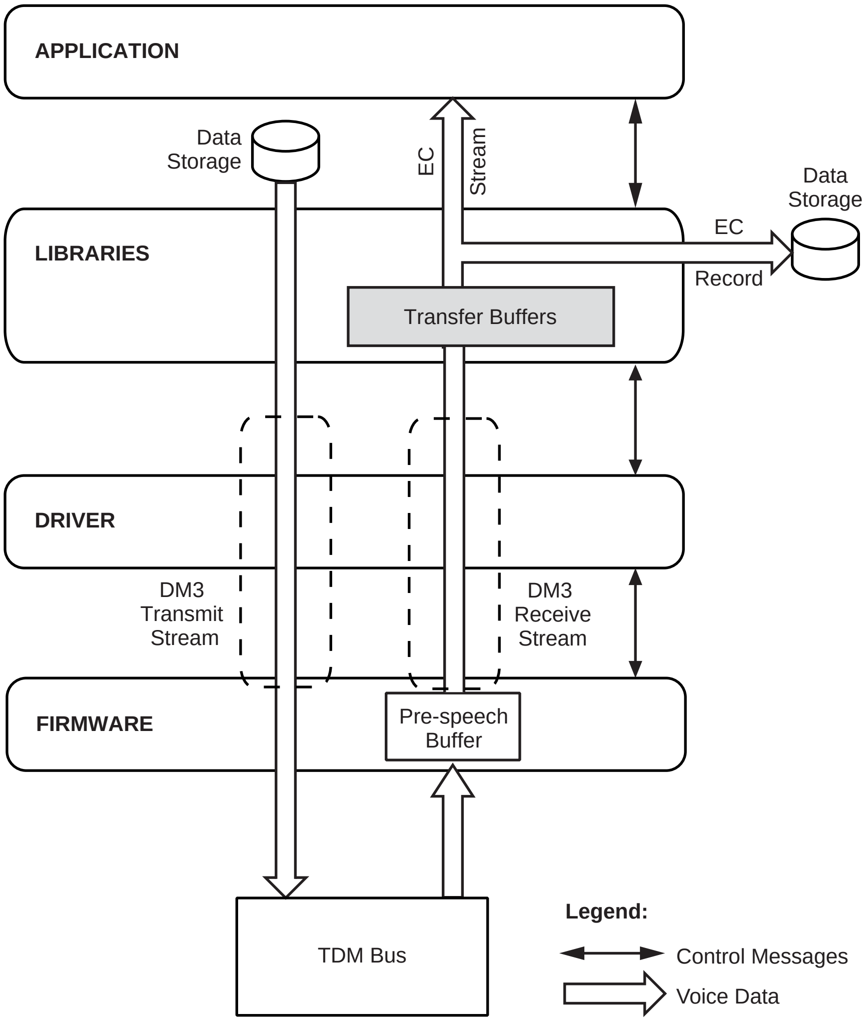

APPLICATION
Data Storage
EC
Stream
Data Storage
LIBRARIES
EC
Record
Transmit
Receive
Driver Buffers
DRIVER
Firmware Buffers
Shared RAM Transmit
Shared RAM Receive
FIRMWARE
Pre-speech Buffer
TDM Bus
Legend:
Control Messages
Voice Data

Figure 7. Data Flow from Application to Firmware (DM3 Boards)


APPLICATION
Data Storage
EC
Stream
Data Storage
EC
Record
LIBRARIES
Transfer Buffers
DRIVER
DM3 Transmit Stream
DM3 Receive Stream
FIRMWARE
Pre-speech Buffer
TDM Bus
Legend:
Control Messages
Voice Data

6.3 Buffer Usage Tips

The following guidelines apply to both Springware boards and DM3 boards:

- The smaller you make the driver buffer size or transfer buffer size, the more interrupts are generated to handle the buffers, and consequently, there is an associated increase in CPU loading. Therefore, you must choose this value carefully to maximize throughput while minimizing system load.
- The speed of the host processor, as well as other concurrent processing, has an impact on how low the buffers can be set.

When adjusting buffer sizes on Springware boards, keep the following guidelines in mind:

- In general, the driver buffer size should be at least **two** times the size of the firmware buffer. For ASR applications, the driver buffer size should be at least **three** times the size of the firmware buffer. If it isn't, play/record may terminate abruptly and data loss may occur.

  For example, if the firmware buffer size is 512 bytes, then the driver buffer size in ASR applications should be at least 1536 bytes. See Table 5, "Types of Buffers Used in CSP (Springware Boards)", on page 48 for driver buffer size limitations in Linux.
- Simply reducing the driver buffer size does **not** guarantee better performance. In fact, if the value is poorly chosen, the exact opposite may result.

When adjusting buffer sizes on DM3 boards, keep the following guideline in mind:

- For ASR applications, set the transfer buffer size to a value less than or equal to 2 kbytes so that no host buffering is performed, for minimal latency.

Echo Canceller Convergence Notification

7

This chapter describes how to enable the echo canceller convergence notification feature.

The echo canceller convergence notification feature is not supported on Springware boards.

In your Continuous Speech Processing (CSP) application, you can specify whether an echo canceller convergence event is sent to the host application.

Convergence refers to the point at which the echo canceller has processed enough data to be able to identify the echo component in the incoming signal, and thereby reduce the echo to provide echo-cancelled data to the host.

Use the DM_CONVERGED parameter in **dx_setevtmsk( )** to have the firmware send an echo canceller convergence event to the host application. For function reference information, see the *Voice API Library Reference*.

Using this parameter provides an extra safeguard to help determine when the echo canceller has converged and when the echo-cancelled data is ready for further processing.

The event sent to the host application is called TEC_CONVERGED. Use **sr_waitevt( )**, **sr_enbhdlr**( ), or other SRL function to collect an event code, depending on the programming model in use. For more information about programming models and these functions, see the *Standard Runtime Library API Library Reference*.

The TEC_CONVERGED event is only reported if the echo is present for at least 1.5 seconds (and the echo canceller has converged). This is done to ensure reliability of the converged event, as it takes approximately 1-2 seconds for the echo canceller to converge on speech signals. While convergence is much quicker on tones, almost all use cases of echo cancellation in CSP involve speech and not tones; hence the confidence factor of a minimum of 1.5 seconds.

If the 1.5 seconds has expired, there is a further criteria whereby the TEC_CONVERGED event is only reported if the echo canceller provides a rejection of more than 20 dB from the incoming echo.

By default, no event is sent to the host application.

Building Applications 8

This chapter provides general information on building applications using the Continuous Speech Processing (CSP) software. The following topics are covered:

- CSP Library Integration with Voice Libraries 55
- Compiling and Linking 56

8.1 CSP Library Integration with Voice Libraries

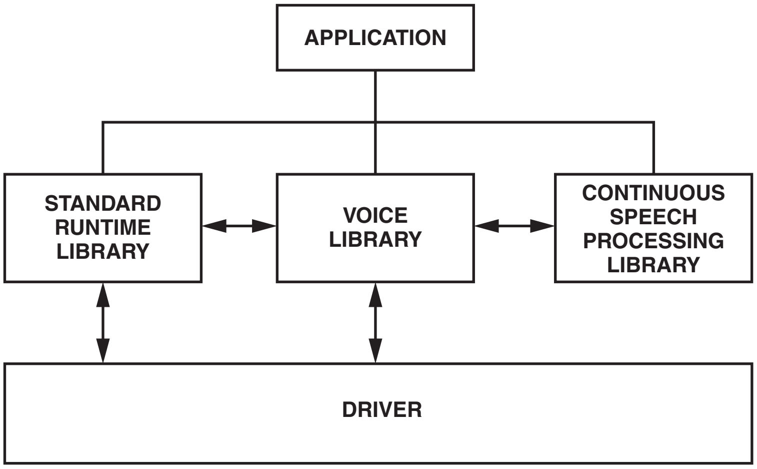
The C-language application programming interface (API) included with the Continuous Speech Processing (CSP) software provides a library of functions used to create CSP-enabled applications. These functions are integrated with the voice library. This architecture in turn enables you to add CSP capability to an existing voice application.

The voice library and Standard Runtime Library files are part of the system voice software. The CSP library, together with the voice libraries, provide the interface to the voice driver.

Figure 8 illustrates the relationship among the three libraries.

Note: On DM3 boards, the CSP library communicates directly with the driver.

Figure 8. CSP, SRL and Voice Libraries



For more information on voice and SRL libraries, see the *Voice API Library Reference* and the *Standard Runtime Library API Library Reference*, respectively.

8.2 Compiling and Linking

The following topics discuss compiling and linking requirements:

- Include Files
- Required Libraries
- Variables for Compiling and Linking

8.2.1 Include Files

To use Continuous Speech Processing (CSP) API functions in your application, certain include files (also known as header files) and library files are required.

Function prototypes and equates are defined in include files, also known as header files. Applications that use CSP library functions must contain statements for include files in this form, where filename represents the include file name:

```
include <filename.h>
```

You must include the following voice and CSP header files in your application **in the order shown** before calling any Intel® Dialogic® API functions:

srllib.h
Contains function prototypes and equates for the Standard Runtime Library (SRL). Used for all application development.

dxxxlib.h
Contains function prototypes and equates for the voice library. Used for voice processing.

eclib.h
Contains function prototypes and equates for the Continuous Speech Processing (CSP) library. Used for CSP.

If using Global Call and Network Interface libraries with CSP, include the following Intel Dialogic header files **in this order**:

srllib.h
Contains function prototypes and equates for the Standard Runtime Library (SRL). Used for all application development.

dxxxlib.h
Contains function prototypes and equates for the voice library. Used for voice processing.

dtilib.h
Contains function prototypes and equates for the digital network interface library. Used for digital network processing.

eclib.h
Contains function prototypes and equates for the Continuous Speech Processing (CSP) library. Used for CSP.

gclib.h
Contains function prototypes and equates for the Global Call library. Used for uniform call control.

8.2.2 Required Libraries

The required library files are listed separately for Linux and Windows.

Linux

When using CSP in Linux, you must link the following library files **in the order shown** when compiling your application:

libec.so
CSP library file. Specify `-lec` in makefile.

libdxxx.so
Main voice library file. Specify `-ldxxx` in makefile.

libsrl.so
Standard Runtime Library file. Specify `-lsrl` in makefile.

If you use `curses`, you must ensure that it is the last library to be linked.

When using Global Call with CSP in Linux, you must link the Global Call library (*libgc.so*) before linking the CSP library (*libec.so*). Additionally, you may need to link the digital network interface library (*libdti.so*). Thus, when using CSP and Global Call, you would link the following: *libgc.so*, *libec.so*, *libdxxx.so*, *libdti.so*, *libsrl.so*.

Windows

When using CSP in Windows, you must link the following library files when compiling your application:

libecmt.lib
CSP library file.

libdxxmt.lib
Voice library file is required.

libsrlmt.lib
Standard Runtime Library is required.

When using Global Call with CSP in Windows, you must link the Global Call library (*libgc.lib*) before linking the CSP library (*libecmt.lib*). Additionally, you may need to link the digital network interface library (*libdtimt.lib*). Thus, when using CSP and Global Call, you would link the following: *libgc.lib*, *libecmt.lib*, *libdxxmt.lib*, *libdtimt.lib*, *libsrlmt.lib.*

8.2.3 Variables for Compiling and Linking

In System Release 6.0, the following variables have been introduced to provide a standardized way of referencing the directories that contain header files and shared objects:

INTEL_DIALOGIC_INC
Variable that points to the directory where header files are stored.

INTEL_DIALOGIC_LIB
Variable that points to the directory where shared library files are stored.

These variables are automatically set at login and should be used in compiling and linking commands. The following is an example of a compiling and linking command that uses these variables:

```
cc -I${INTEL_DIALOGIC_INC} -o myapp myapp.c -L${INTEL_DIALOGIC_LIB} -lgc
```

Note: It is strongly recommended that developers begin using these variables when compiling and linking applications since they will be required in future releases. The name of the variables will remain constant, but the values may change in future releases.

Glossary

application programming interface (API): A set of standard software interrupts, calls, and data formats that application programs use to initiate contact with network services or other program-to-program communications.

asynchronous function: A function that allows program execution to continue without waiting for a task to complete. To implement an asynchronous function, an application-defined event handler must be enabled to trap and process the completed event. See synchronous function.

automatic speech recognition (ASR): A set of algorithms that processes speech utterances.

barge-in: The act of a party beginning to speak while a prompt is being played. When the VAD detects significant energy in the voice channel, CSP can optionally terminate prompts playing on that channel. Thus the party on the other end of the line is said to have “barged in” on the prompt.

buffer: A block of memory or temporary storage device that holds data until it can be processed. It is used to compensate for the difference in the rate of the flow of information when transmitting data from one device to another.

comfort noise generation (CNG): The ability to produce a background noise when there is no speech on the telephone line.

convergence: The point at which the echo canceller processes enough data to be able to identify the echo component in the incoming signal and thereby reduce it to provide echo-cancelled data to the host.

device: A computer peripheral or component controlled through a software device driver. A voice and/or network interface expansion board is considered a physical board containing one or more logical board devices, where each channel or time slot on the board is a device.

DM3: Mediastream processing architecture from Intel. It is open, layered, and flexible, encompassing hardware as well as software components. A whole set of products are built on DM3 architecture.

driver: A software module that provides a defined interface between a program and the hardware.

echo: The component of an outgoing signal (that is, the play prompt) reflected in the incoming signal. The echo occurs when the signal passes through an analog device or other interface somewhere in the circuit.

echo-cancelled signal: The incoming signal whose echo component has been significantly reduced by the echo canceller.

echo cancellation (EC): A technique used to significantly reduce traces of an outgoing prompt in the incoming signal. These traces are referred to as echo. The **echo canceller** is the component in CSP responsible for performing echo cancellation.

firmware: A set of program instructions that are resident (usually in EPROM) on an expansion board.

fixed routing: In this configuration, the resource devices (voice/fax) and network interface devices are permanently coupled together in a fixed configuration. Only the network interface time slot device has access to the CT Bus.

flexible routing: In this configuration, the resource devices (voice/fax) and network interface devices are independent, which allows exporting and sharing of the resources. All resources have access to the CT Bus.

incoming signal or incoming speech signal: The speech uttered by the caller, or the DTMF tone entered by the caller. Also known as the **echo-carrying signal**. This signal contains an echo component only if an outgoing prompt is played while the incoming signal is generated.

latency: The lag time experienced as a result of audio energy traveling over the telephone or data network from the sender to the receiver.

library: A collection of precompiled routines that a program can use. The routines, sometimes called modules, are stored in object format. Libraries are particularly useful for storing frequently used routines because you do not need to explicitly link them to every program that uses them. The linker automatically looks in libraries for routines that it does not find elsewhere.

non-linear processing (NLP): A process used to block or suppress the residual (echo-cancelled) signal, when there is no near end speech. This process can be used with **comfort noise generation** (CNG) to produce background noise. Background noise energy estimation is used to adjust the level of comfort noise generated. This allows the speaker to listen to the same level of background noise when the non-linear processor is switched on and off due to double-talk situations or near end speech. A typical usage of this feature is background noise used in dictation applications to let the user know that the application is working.

outgoing prompt or outgoing signal: The speech in a computer telephony application that is played to a caller. Also known as the **echo-generating signal**.

pre-speech buffer: A circular buffer that stores the incoming speech signal and is used to reduce the problem of clipped speech. This data, which includes the incoming speech signal prior to the VAD trigger, is then sent to the host application for processing. This action ensures that minimal incoming data is lost due to VAD latency.

reference signal or echo-reference signal: The outgoing signal that is free of echo before it is passed to the network device. This signal is used by the echo canceller to characterize the echo to be removed from the incoming signal.

Standard Attribute functions: Class of functions that take one input parameter (a valid device handle) and return generic information about the device. For instance, Standard Attribute functions return IRQ and error information for all device types. Standard Attribute function names are case-sensitive and must be in capital letters. Standard Attribute functions for all devices are contained in the SRL. See Standard Runtime Library.

Springware: Downloadable signal- and call-processing firmware from Intel. Also refers to boards whose device family is not DM3.

Standard Runtime Library (SRL): Software resource containing Event Management and Standard Attribute functions and data structures used by all devices, but which return data unique to the device.

synchronous function: A function that blocks program execution until a value is returned by the device. Also called a blocking function. See asynchronous function.

tap length: Also called tail length or length. Refers to the number of milliseconds of echo that is eliminated from the incoming signal. The length of an echo canceller is sometimes given as “taps,” where each tap is 125 microseconds long.

TDM bus: time division multiplexing bus. A resource sharing bus such as the SCbus or CT Bus that allows information to be transmitted and received among resources over multiple data lines.

utterance: Speech made by the user.

voice activity detector (VAD): Also called voice energy detector. This detector identifies the presence of speech energy and determines when significant energy is detected in the voice channel. It notifies the host application that speech energy is detected.

zero-crossing: Refers to energy that crosses over the zero mark many times over a short period of time and thus has a high probability of being speech.

Index

A

adaptation modes 18

A-law PCM 22

AT_FAILURE 29

AT_FAILUREP 29

ATDV_ERRMSGP( ) 29

ATDV_LASTERR( ) 29

automatic gain control (AGC) 34

automatic speech recognition applications, buffer usage tips 52

B

barge-in
  - details 20
  - enabling 39
  - play and non-play situations 44

barge-in package, compared 23

buffers
  - definition 47
  - DM3 49
  - illustrated 49
  - Springware 48
  - types of 47
  - usage tips 52

C

comparison
  - CSP and other features 23

compatibility considerations, Springware 35

compiling applications 57

compiling variables 58

CONFIG files 40

configuration manager (DCM) 32

convergence 18

CSP
  - components 16
  - data flow diagram 14
  - data formats supported 22
  - feature comparison 23
  - features 13
  - illustrated 16

CSP library
  - error handling 29
  - overview 57

D

data flow
  - illustrated 49

data formats
  - on DM3 boards 22
  - on Springware boards 22

DCM 32

DM_VADEVTS 38

driver buffer size, tips 52

driver buffers, limitation in Linux 48

DSP-based fax, compatibility issues 35

dx_setevtmsk( ) 38

DXBD_RXBUFSIZE 48

DXBD_TXBUFSIZE 48

DXCH_BARGEIN 39

DXCH_BARGEINONLY 39

DXCH_EC_TAP_LENGTH 18

dxxxlib.h 56

E

EC_BLK_INFO structure 34

ec_getblkinfo( ) 34

ec_setparm( )
  - DXCH_BARGEIN 39
  - ECCH_VADINITIATED 39

ECCH_ADAPTMODE 19

ECCH_ECHOCANCELLER 18

ECCH_SILENCECOMPRESS 34, 39

ECCH_VADINITIATED 20, 39

ECCH_XFERBUFFERSIZE 48, 49
  - limitation in Linux 48

echo cancellation resource (ECR), compared 23

echo canceller
  - adaptation modes 18
  - details 17
  - tap length 18

echo, illustrated 14

eclib.h 56

energy mode 45

error handling 29

event
  signaling 38
events
  TDX_BARGEIN 39
extended attribute functions
  error handling 29
external reference signal 16

F

fast mode, echo canceller 18
FCD files 40
fcdgen utility 40
feature comparison
  CSP and other features 23
  DM3 versus Springware boards 23
features 13
firmware buffers 48
  size tips 52
full-duplex operation 13

H

header files 56
high-density echo cancellation (HDEC), compared 23

I

include files 56
INTEL_DIALOGIC_INC 58
INTEL_DIALOGIC_LIB 58
interoperability considerations 35
ISDN, compatibility issues 36

L

libdxxmt.lib 57
libdxxx.so 57
libec.so 57
libecmt.lib 57
library files 57
library header files 56
libsrl.so 57
libsrlmt.lib 57
linear PCM 22
linking libraries 57
linking variables 58

M

modes
  adaptation 18
  echo canceller 18
  voice activity detector 37
mu-law PCM 22

O

OKI ADPCM 22

P

play and non-play situations 44
pre-speech buffer 19, 48, 49

R

recording data 20
  terminology 20
  voice-activated 20
  with silence removed 21

S

silence compressed streaming 21, 34
  configuration parameters 39
  details 39
  scenario 42
slow mode, echo canceller 18
speech window
  illustrated 43
sr_waitevt( ) 38
srllib.h 56
standard attribute functions
  error handling 29
Standard Runtime Library 57
streaming data 20
  constantly 20
  over TDM bus 21, 32
  terminology 20
  voice-activated 20
  with silence removed 21
SVAD 45

T

tap length 18
TDM bus, streaming data on 21, 32
TDX_BARGEIN event 39

TEC_VAD event 38
time slot, assigning for CSP 32
transaction record, compatibility issues 35
transfer buffers 49

U

usage tips, buffer sizes 52

V

variables, compiling and linking 58
voice activity detector (VAD)
    details 19
    fine-tuning performance guidelines 44
    illustrated 43
    operating modes 37
    sample operation 43
voice coders
    on DM3 boards 22
    on Springware boards 22
voice event signaling
    details 20
voice library 57
voice-activated recording
    definition 20
    enabling 39
voice-activated streaming
    enabling 39
    See voice-activated recording 20
VOX format 22

W

WAVE format 22

Z

zero-crossing mode 45