Continuous Speech Processing API for Linux and Windows

Programming Guide

December 2001

05-1699-001

COPYRIGHT NOTICE

Publication Date: December 2001

Document Number: 05-1699-001

Intel Corporation
Telecommunications and Embedded Group
1515 Route 10
Parsippany, NJ 07054

For **Technical Support**, visit the Dialogic support website at:
http://support.dialogic.com

For **Sales Offices** and other contact information, visit the main Dialogic website at:
http://www.dialogic.com

Table of Contents

Tables

Figures

About This Publication

The following topics provide information about this publication:

- Purpose
- 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 running on Dialogic supported boards, both SpringWare (also known as earlier-generation) and DM3 boards, 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.

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 Software Reference* 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:

- *Continuous Speech Processing API Library Reference*
- *Continuous Speech Processing Demo Guide*
- *System Release Guide*
- *System Release Update* (available on the Dialogic Technical Support Web site only)
- *Voice Software Reference* (includes the *Voice Features Guide, Standard Runtime Library Programmer's Guide* and *Voice Programmer's Guide*)
- *Compatibility Guide for the Dialogic R4 API on DM3 Products*
- *SCbus Routing Function Reference*
- *GlobalCall™ API Software Reference*
- *ISDN Software Reference*
- *DM3 Configuration File Reference*
- Dialogic FirstCall InfoServer™ Web site at *http://support.dialogic.com*

Product Description 1

The following topics provide information on Continuous Speech Processing:

1.1 Features

Continuous Speech Processing (CSP) is software that supports development of host-based automatic speech recognition (ASR) applications on specific Dialogic boards. It provides many features such as high-performance echo cancellation, voice energy detection, barge-in, voice event signaling, pre-speech buffering, full-duplex operation and more.

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 Dialogic 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 eliminates up to 16 milliseconds of echo in the incoming signal.
- 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 that can store up to 250 milliseconds of incoming speech in a circular buffer, reducing the problem of clipped speech and increasing recognition accuracy. This buffered data is passed on to the host application along with subsequent speech signals.
- 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 rearm or re-enable the VAD. This is useful when non-speech such as a cough is determined.

- 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.
- Demonstration program that illustrates the key features of CSP.

The following features are available on DM3 boards only:

- Echo canceller convergence event notification – ability to send an event to the host application when the echo canceller has converged; that is, the echo component has been significantly reduced.
- More powerful voice activity detector (VAD) – the VAD performs sophisticated calculations using a combination of energy and zero-crossing mode (where the energy level goes to zero for a time period) to accurately determine the start of speech.

For a description of technical terms, see the Glossary.

1.2 How CSP Works

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 prompt), 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 speech 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.

Several CSP components (listed below) make this connection possible. Many of these components reside in the firmware level of the CSP-capable board.

- Echo Canceller
- Voice Activity Detector (VAD)
- Pre-Speech Buffer
- Barge-In and Voice Event Signaling
- Recording or Streaming to the Host

Figure 1-1 depicts the data flow from the network to the CSP voice channel. It also shows how echo is introduced in the signal in the network and how it is cancelled. On DM3 boards, the option of sending the echo-cancelled signal over the TDM bus to another board is not available.

Additional functionality is provided that allows you to send a reference signal from another device over the TDM bus to the CSP voice channel. This external reference signal can be from a secondary network device. Figure 1-2 depicts this feature.

Figure 1-1. CSP Components and Data Flow

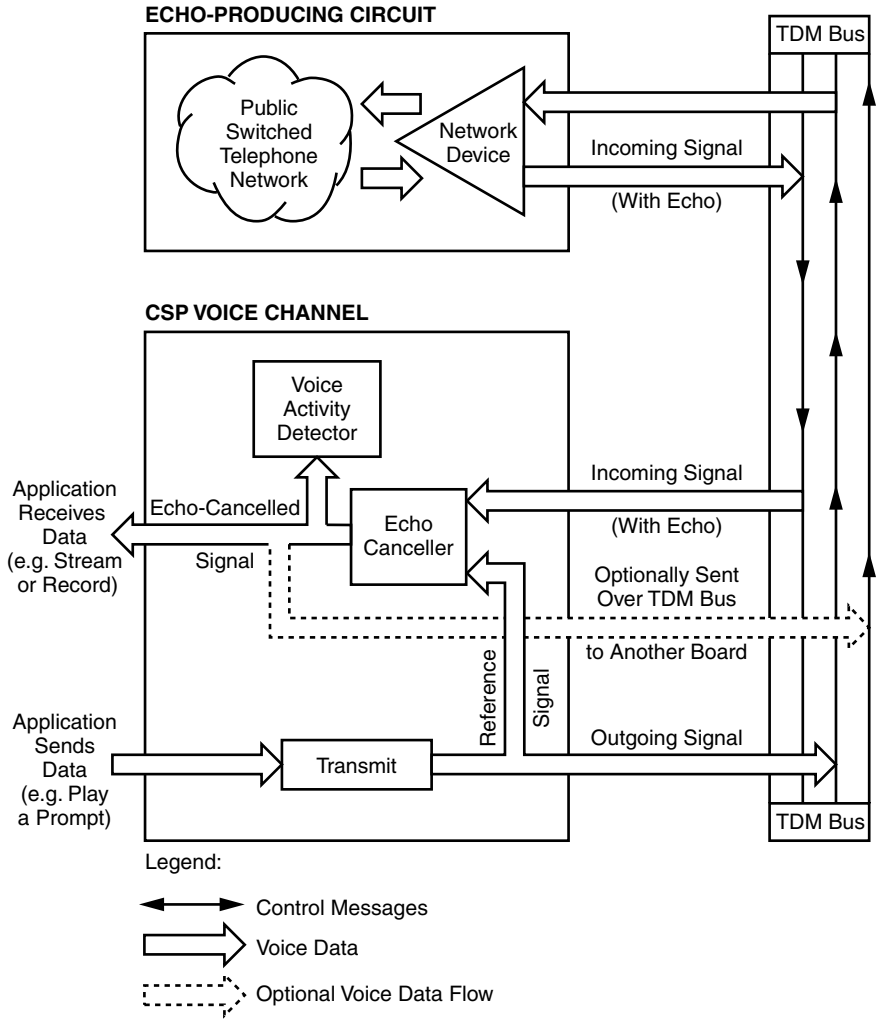
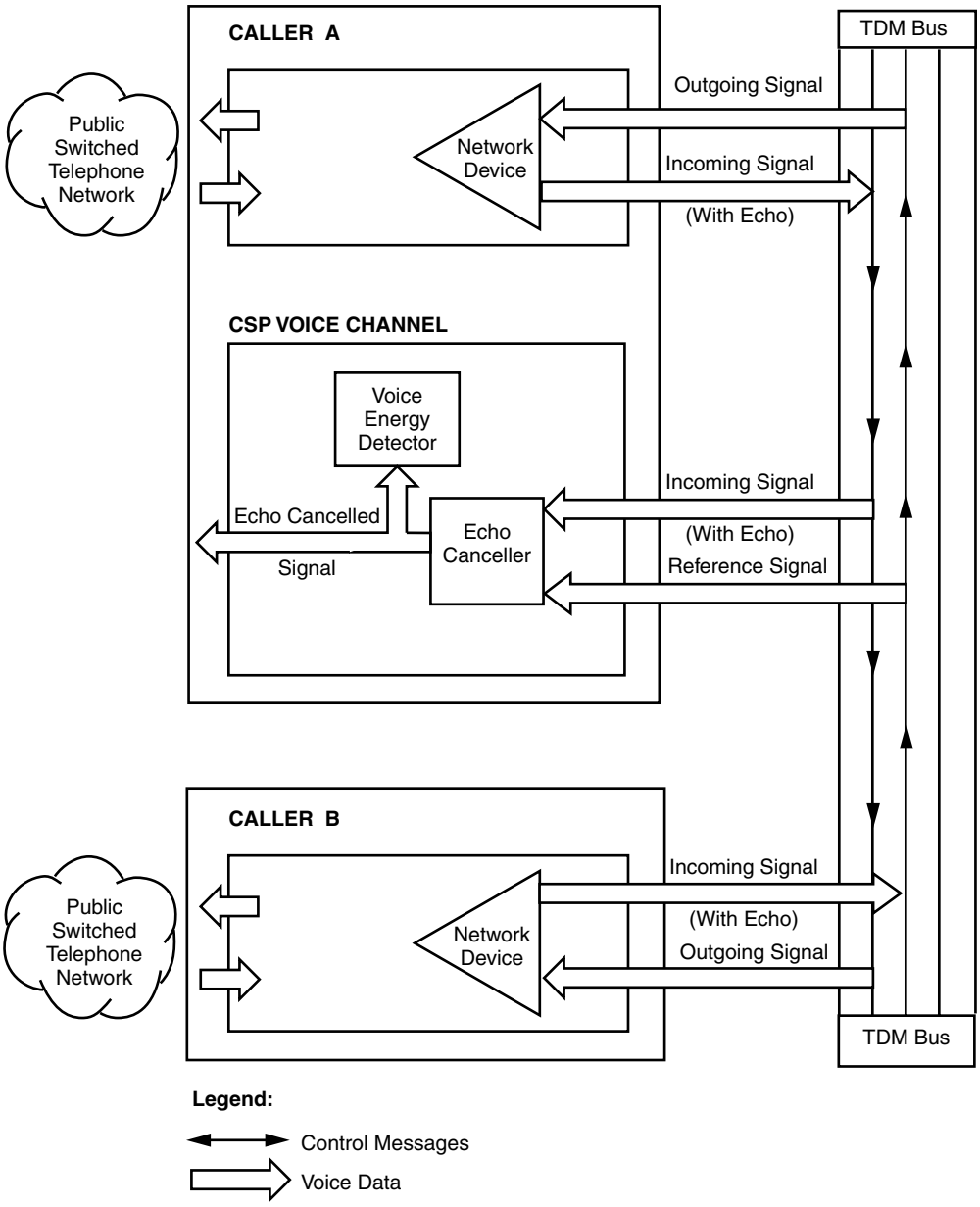


Figure 1-2. Echo Cancellation Using an External Reference Signal



In Figure 1-2, 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).

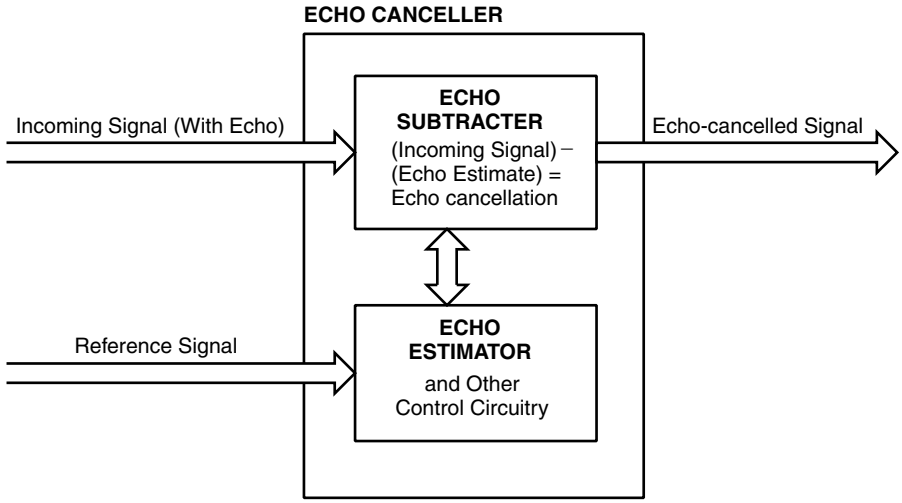
By using caller B's incoming signal as a reference signal, the echo canceller produces an echo-cancelled signal for caller A.

1.2.1 Echo Canceller

The echo canceller is a component in the CSP software that is used by applications to eliminate echoes in the incoming signal. In the scenario described in Section 1.2, "How CSP Works", on page 1-2, 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 1-3 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 1-3. Echo Canceller



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 speech 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*.

1.2.1.1 Tap Length

Note: On DM3 boards, tap length is not configurable. The tap length is fixed at 128 taps or 16 ms.

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.

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.*

1.2.1.2 Adaptation Modes

Note: 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.

1.2.2 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 5-1.

1.2.3 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 speech signal, 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 speech signal prior to the VAD trigger. The data in the pre-speech buffer is sent to the application along with all subsequent speech signals. Pre-speech buffers are an integral part of VAD. See Figure 6-1, "Data Flow from Application to Firmware", on page 6-4 and Figure 6-2, "Data Flow from Application to Firmware (DM3 Boards)", on page 6-5 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.

1.2.4 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 5.1.2, “Stopping Play When Speech is Detected (Barge-In)”, on page 5-2 and Section 5.1.1, “Sending a VAD Event to the Host Application”, on page 5-2. For detailed function and parameter descriptions, see **ec_setparm( )** in the *Continuous Speech Processing API Library Reference*.

1.2.5 Recording or Streaming to the Host

You can choose to record (stream) data at all times or only after the VAD has detected speech. Voice-activated recording refers to the process of **recording** or **streaming** data to the host application only after the VAD has detected speech. 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 7.1.4, “Sample VAD Scenarios”, on page 7-3 and the function descriptions in the *Continuous Speech Processing API Library Reference*.

1.3 Supported Data Formats

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

- Supported Data Formats on SpringWare Boards
- Supported Data Formats on DM3 Boards

1.3.1 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 (WAVE or VOX file format):

- Mu-law PCM, 8 kHz sampling rate, 8-bit resolution (64 kbps)
- 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. On SpringWare boards, we recommend that you use the same data format for play and recording/streaming.

1.3.2 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( )** functions (WAVE or VOX file format with some exceptions):

- Mu-law PCM, 8 kHz sampling rate, 8-bit resolution (64 kbps)
- 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) (VOX file format only)

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

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):

- Mu-law PCM, 6 and 8 kHz sampling rate, 8-bit resolution (48 and 64 kbps)
- A-law PCM, 6 and 8 kHz sampling rate, 8-bit resolution (48 and 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.

1.4 Comparison with Existing Features

Table 1-1 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. For a complete list of hardware that supports CSP, see the *Release Guide* and *Release Update*.

Table 1-1. 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	Dialogic 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 1.3, "Supported Data Formats", on page 1-8.
Echo cancellation tap length	6 milliseconds (48 taps) used only for signal detection	Up to 16 milliseconds (128 taps)	Up to 16 milliseconds (128 taps)
Integrated barge-in and echo-cancelled streaming	No	No	Yes
Pre-speech buffers	256 bytes (can store up to 125 milliseconds)	No	Can store up to 250 milliseconds 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

1.5 CSP Support on SpringWare Versus DM3 Boards

Table 1-2 summarizes the differences that exist when running CSP on DM3 boards versus on SpringWare boards.

For more information on parameters and functions mentioned in Table 1-2, see the **ec_setparm( )** function description and other function descriptions in the *Continuous Speech Processing API Library Reference*.

Table 1-2. 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 tap	128 tap (not modifiable)	
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	DX_MAXTIME, DX_MAXSIL, DX_MAXNOSIL supported only	All conditions supported except DX_MAXTIME, DX_LCOFF, DX_PMON and DX_PMOFF	Applies when using ec_ functions. In CSP, DV_TPT terminating conditions are edge-sensitive.
call control through GlobalCall	Supported	Supported	
call control through the Voice library	Supported	Not supported	
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

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 Programmer's Guide* (in the *Voice Software Reference*).

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

Error Handling 3

Continuous Speech Processing (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 Programmer's Guide* (in the *Voice Software 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 Software Reference*.

Application Development Guidelines

4

When developing your Continuous Speech Processing (CSP) application, refer to the following guidelines and interoperability considerations:

4.1 Guidelines for Developing CSP Applications

Guidelines for developing CSP applications are provided in the following topics:

- Reserving Extra Time Slots
- Including Header Files and Linking
- 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
- Playing a Prompt
- Collecting Events
- Performing Voice Processing
- Cleaning Up

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_**.

4.1.1 Reserving Extra Time Slots

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

Notes: ***1.*** On DM3 boards, this feature is not supported.

2. On D/120JCT-LS boards, you must turn the extra time slot parameter on to use CSP.

In Linux, you configure this time slot at initialization time in *dialogic.cfg*. In Windows, you configure this time slot at initialization time in the Dialogic Configuration Manager (DCM), under the **Misc** tab. See the *System Installation and Configuration Guide* and DCM online help for more information.

To retrieve the number of the time slot which transmitted the echo-cancelled data, use **ec_getxmitslot( )**. For details, see the *Continuous Speech Processing API Library Reference*.

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.

4.1.2 Including Header Files and Linking

In your application, include Dialogic header files *srllib.h*, *dxxxlib.h* and *eclib.h* in this order.

In Linux, link to the *libec.so*, *libdxxx.so*, and *libsrl.so* libraries. In Windows, link to the *libecmt.lib*, *libdxxmt.lib*, and *libsrlmt.lib* libraries. See Section 8.2, “Compiling and Linking”, on page 8-2 for more information.

4.1.3 Opening a Voice Channel

Open a voice channel to obtain the channel device handle. 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 Software 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 GlobalCall functions for call control. On DM3 network devices, use GlobalCall functions to open and close device handles. You can choose to open a voice device together with the network device using **gc_open( )**, or you can open the voice device separately using **dx_open( )**. For SCbus routing, use digital network interface (DTI) functions.

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 *SCbus Routing Function 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 taps for DM3)
- 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)
- 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 5-1, Chapter 6, "Buffers and Data Flow", Section 1.5, "CSP Support on SpringWare Versus DM3 Boards", on page 1-10, and the *Continuous Speech Processing API Library Reference*.

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 Software 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 5.1.1, "Sending a VAD Event to the Host Application", on page 5-2 for more information.

4.1.8 Setting Up EC Convergence Event Notification

Note: 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 record 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 send echo-

cancelled data to the application as it is received. For more information, 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 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 Software Reference*. For a list of data formats supported for CSP, see Section 1.3, "Supported Data Formats", on page 1-8.

4.1.11 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 Programmer's Guide (in the *Voice Software 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*.

4.1.12 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.

4.1.13 Cleaning Up

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

4.2 Interoperability Considerations for SpringWare Boards

For SpringWare boards only. In most cases, the CSP software can operate with other existing Dialogic features. The current ECR feature (also known as HDEC, High-Density Echo Cancellation) continues to be supported with CSP.

To ensure proper operation, however, review compatibility issues for the following areas:

- Transaction Record
- DSP-Based Fax
- ISDN

4.2.1 Transaction Record

For SpringWare boards only. Transaction record enables an application to record a two-way transaction or conversation. It takes two inputs from the SCbus to perform this record. For more information on this feature, see the *Voice Software Reference*.

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

For SpringWare boards only. Currently DSP-based fax cannot be used together with CSP on the same board.

4.2.3 ISDN

For SpringWare boards only. Currently ISDN cannot be used together with CSP on the same span on a board.

Certain DualSpan™ products allow for ISDN on the first span and CSP on the second span. See the *Release Guide* and *Release Update* for more information on hardware support. See the Dialogic Technical Support web site at *http://support.dialogic.com/tnotes/tnbyos/winnt/tn322.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 Dialogic network interface product (DTI/SC) in your system in addition to the Dialogic voice product (such as a D/240JCT-T1).

Using the Voice Activity Detector 5

The following topics provide more information on the voice activity detector (VAD):

5.1 Voice Activity Detector Operating Modes

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

Table 5-1 summarizes the types of operation, the modes and the settings required to enable them.

Table 5-1. 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

The following topics provide more information on the VAD operating modes:

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

5.1.1 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 Programmer's Guide (in the *Voice Software 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 5.1.4, "Sample VAD Scenarios", on page 5-3.

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

5.1.2 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 5.1.4, "Sample VAD Scenarios", on page 5-3.

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.

5.1.3 Voice-Activated or Constant Recording

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

To implement constant recording or streaming, turn this parameter OFF. To implement voice-activated recording or streaming, turn this parameter ON. Speech is transmitted using the **ec_reciottdata( )** or **ec_stream( )** functions.

The default value is speech utterance data is transmitted only after the VAD detects energy. This setting 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 5.1.4, "Sample VAD Scenarios", on page 5-3.

5.1.4 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 5-2 are a few ways in which you might set the DM_VADEVTS, ECCH_VADINITIATED, DXCH_BARGEIN 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 Reference.

Table 5-2. Sample VAD Scenarios

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

In the table, the check mark denotes that the function is initiated or that the parameter is turned on. The "x" denotes that the parameter is turned off.

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

5.1.4.1 Scenario A: Streaming with VAD

In scenario A shown in Table 5-2, "Sample VAD Scenarios", on page 5-3, 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.

5.1.4.2 Scenario B: Streaming with VAD and without VAD Event

Scenario B, shown in Table 5-2, "Sample VAD Scenarios", on page 5-3, 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.

5.1.4.3 Scenario C: Streaming without VAD (Constant Streaming)

In scenario C shown in Table 5-2, "Sample VAD Scenarios", on page 5-3, 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).

5.1.4.4 Scenario D: Constant Streaming with VAD Event

In scenario D shown in Table 5-2, “Sample VAD Scenarios”, on page 5-3, 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:

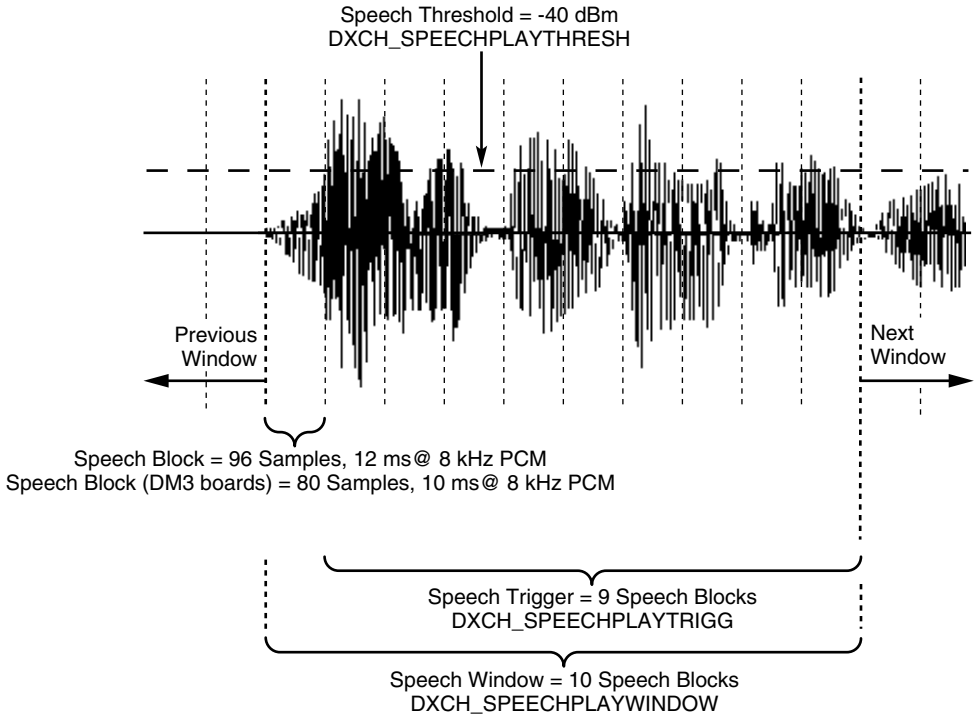
- The **ec_stream( )** or **ec_reciottdata( )** function is initiated in the application, and data streaming or recording begins.
- Voice-activated streaming/recording is not turned on (ECCH_VADINITIATED = 0).
- The firmware sends a VAD event notification to the host application (DM_VADEVTS bit turned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

5.2 VAD Operation

The best way to describe the VAD operation is by example. Figure 5-1 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 5-1. Example of Voice Activity Detector (VAD) Operation



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 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.

  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.
- 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 it 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 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 are used.

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

- Fine-Tuning VAD Performance on SpringWare Boards
- Fine-Tuning VAD Performance on DM3 Boards

5.3.1 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*.

5.3.2 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

The following topics provide information on buffers and data flow:

6.1 Types of Buffers

A buffer is a temporary storage area for data transfer. The CSP software uses buffers at different layers in the Dialogic software to transfer data between the layers, 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 6-1. **For DM3 boards**, the CSP software uses the buffers shown in Table 6-2.

See Figure 6-1, “Data Flow from Application to Firmware”, on page 6-4 and Figure 6-2, “Data Flow from Application to Firmware (DM3 Boards)”, on page 6-5 for an illustration of these buffers.

Table 6-1. 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	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.
Pre-speech buffer	No: 250 ms	Not configurable.

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 6-2. Types of Buffers Used in CSP (DM3 Boards)

Buffer name	Configurable	Parameter/Description
Transfer buffers	Yes: from 240 bytes (30 ms) 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 6-1 depicts the data flow as data travels through the layers in the Dialogic software, from the application to the firmware level. Buffers are also identified. For DM3 boards, see Figure 6-2.

These diagrams are intended to illustrate the concepts rather than the actual physical location of buffers. For more information on buffers, see Table 6-1, “Types of Buffers Used in CSP (SpringWare Boards)”, on page 6-2 and Table 6-2, “Types of Buffers Used in CSP (DM3 Boards)”, on page 6-3.

Figure 6-1. Data Flow from Application to Firmware

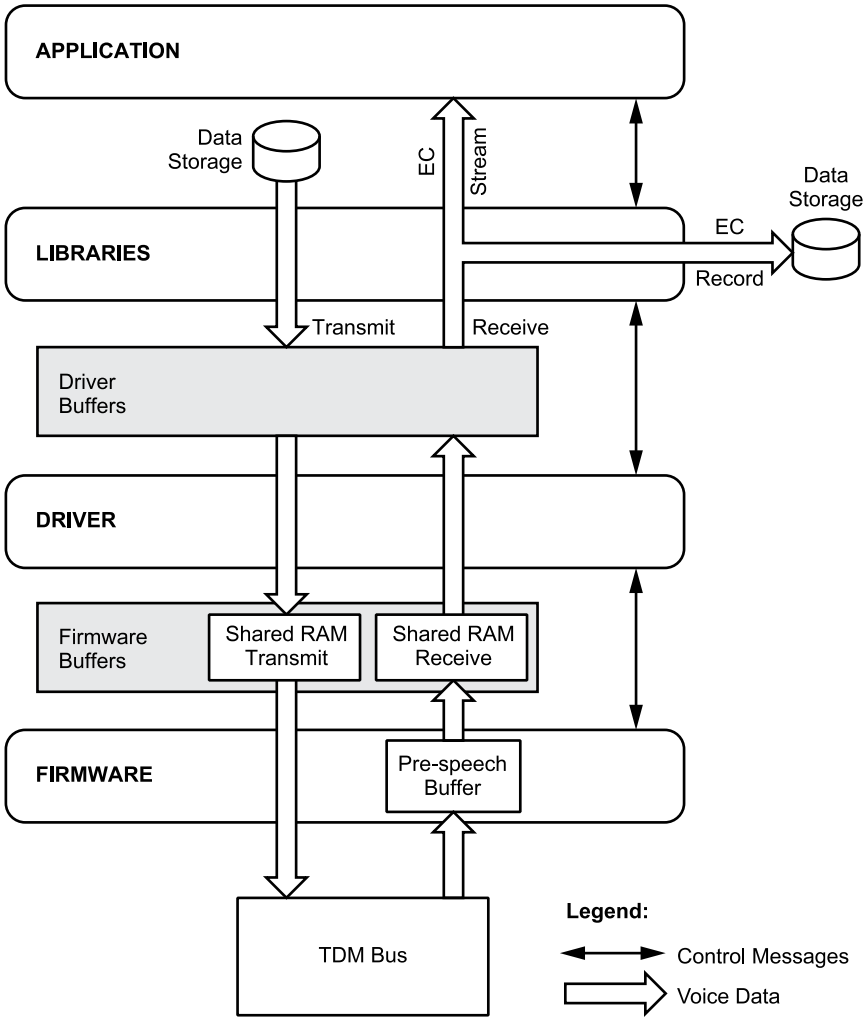
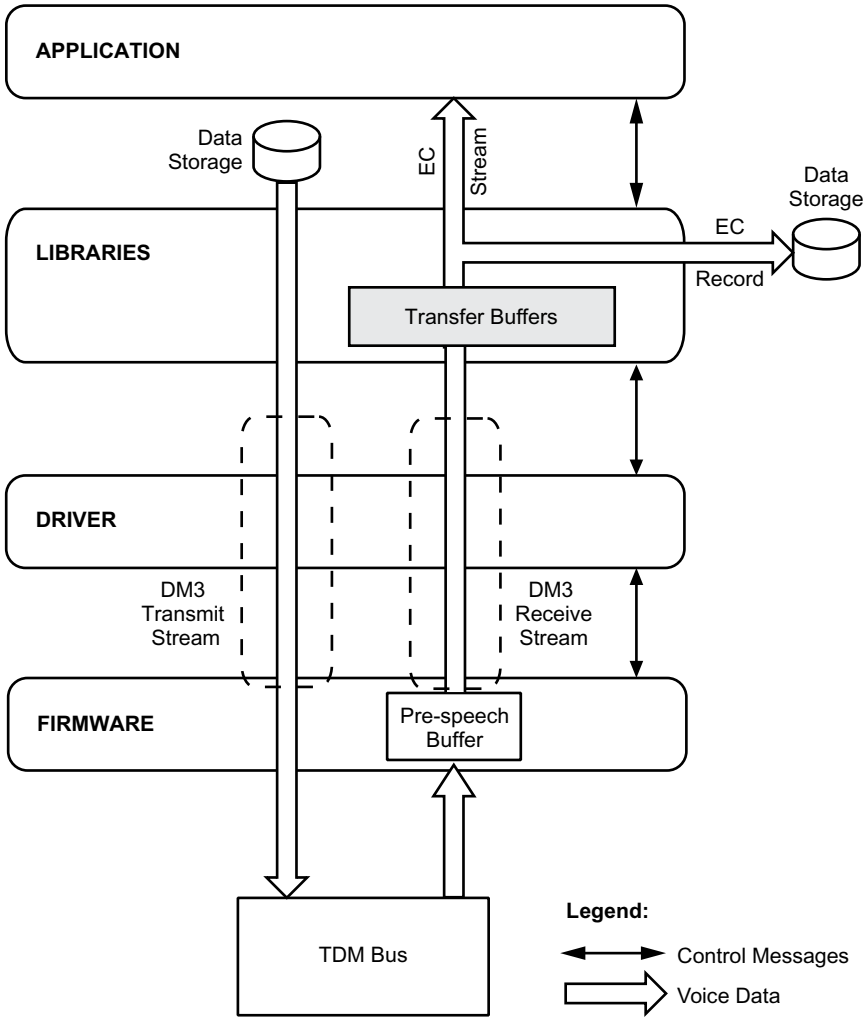


Figure 6-2. Data Flow from Application to Firmware (DM3 Boards)



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.

For SpringWare boards only. 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 6-1, "Types of Buffers Used in CSP (SpringWare Boards)", on page 6-2 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.

For DM3 boards only. 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

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

Note: This feature is not supported on SpringWare boards. It is supported on DM3 boards only.

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.

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, see the Standard Runtime Library Programmer's Guide (in the Voice Software Reference).

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

Building Applications 8

Information on building applications is provided in the following topics:

8.1 CSP Library Integration with Voice Libraries

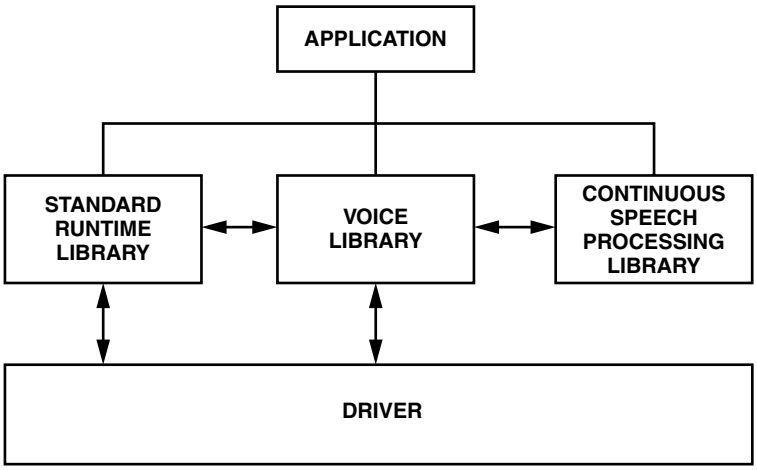
The C-language application programming interface (API) included with the Continuous Speech Processing (CSP) software provides a library of functions used to create CSP-enabled applications. These functions are integrated with the voice library. This architecture in turn enables you to add CSP capability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

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

Figure 8-1 illustrates the relationship among the three libraries.

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

Figure 8-1. CSP, SRL and Voice Libraries



For more information on voice libraries, see the *Voice Software Reference*.

8.2 Compiling and Linking

To use Continuous Speech Processing (CSP) API functions in your application, certain include files (also known as header files) and library files are required. You must add statements for these include files in your application, and you must then link these library files when compiling.

The following topics discuss compiling and linking requirements:

- Include Files
- Required Libraries

8.2.1 Include Files

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 Dialogic functions:

srllib.h
Contains function prototypes and equates for the Standard Runtime Library (SRL). Used for all Dialogic 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 GlobalCall and Network Interface libraries with CSP, include the following Dialogic header files **in this order**:

srllib.h
Contains function prototypes and equates for the Standard Runtime Library (SRL). Used for all Dialogic 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 GlobalCall library. Used for uniform call control.

8.2.2 Required Libraries

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.

libpthread.so
POSIX threads system library. Specify **-lpthread** in makefile.

libLiS.so
Linux Streams (LiS) library. Specify **-lLiS** in makefile.

libdl.so
Dynamic Loader system library. Specify **-ldl** in makefile.

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

When using GlobalCall with CSP **in Linux**, you must link the GlobalCall 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 GlobalCall, you would link the following: *libgc.so*, *libec.so*, *libdxxx.so*, *libdti.so*, *libsrl.so*.

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 GlobalCall with CSP **in Windows**, you must link the GlobalCall 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 GlobalCall, you would link the following: *libgc.lib*, *libecmt.lib*, *libdxxmt.lib*, *libdtimt.lib*, *libsrlmt.lib.*

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 speeech 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. An Intel Dialogic 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: Dialogic mediastream processing architecture. It is open, layered, and flexible, encompassing hardware as well as software components. A whole set of Dialogic 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 timeslot 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 Dialogic 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 Dialogic devices are contained in the Dialogic SRL. See Standard Runtime Library.

SpringWare: A Dialogic downloadable signal and call processing firmware. Also refers to boards whose device family is not DM3.

Standard Runtime Library: A Dialogic software resource containing Event Management and Standard Attribute functions and data structures used by all Dialogic 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-multiplexed 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.

Index

F

G

H

I

L

M

O

P

R