MOTOROLA

Embedded SDK (Software Development Kit)

Generic Echo Canceller Library

SDK136/D
Rev. 1, 07/22/2002

MOTOROLA intelligence everywhere

digital dna

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Contents

Chapter 7
License

List of Tables

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Figures

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Code Examples

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

About This Document

This manual describes the Generic Echo Canceller Library for use with Motorola's Embedded Software Development Kit (SDK).

Audience

This document targets software developers implementing communication features for analog telephone lines.

Organization

This manual is arranged in the following sections:

- **Chapter 1, Introduction,** provides a brief overview of this document
- **Chapter 2, Directory Structure,** provides a description of the required core directories
- **Chapter 3, Generic Echo Canceller Library Interfaces,** describes all of the Generic Echo Canceller Library functions
- **Chapter 4, Building the Generic Echo Canceller Library,** tells how to build a test application project with the pre-built Generic Echo Canceller Library
- **Chapter 5, Linking Applications with the Generic Echo Canceller Library,** describes linking projects with the Generic Echo Canceller Library
- **Chapter 6, Generic Echo Canceller Library Applications,** describes the use of the Generic Echo Canceller Library through test/demo applications
- **Chapter 7, License,** provides the license required to use this product

Suggested Reading

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

- *Motorola DSP56800E Reference Guide*, DSP56800ERM/D
- *Motorola DSP568xx User's Manual*, for the DSP device you're implementing
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Code examples	`//Process command for line flash`
Italic	Directory names Project names Calls Functions Statements Procedures Routines Arguments File names Applications Variables Directives Code snippets in text	...and contains these core directories: *applications* contains applications software... ...CodeWarrior project, *3des.mcp* is... ...the *pConfig* argument.... ...defined in the C header file, *aec.h*....
Bold	Reference sources Paths Emphasis	...refer to the **Targeting DSP5685x Platform** manual.... ...see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
ALL CAPITAL LETTERS	# defines/ Defined constants	# define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]
Quotation marks, “...”	Returned messages	...the message, “Test Passed” is displayed.... ...if unsuccessful for any reason, it will return “NULL”...
Blue Text	Linkable on-line	...refer to Chapter 7, License....
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}

Definitions, Acronyms, and Abbreviations

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

ADC Analog-to-Digital Converter

AGC	Automatic Gain Control
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BPF	Bandpass Filter
CID	Caller ID (Calling Party Name and/or Number Identification)
CPE	Customer Premises (telephony) Equipment
CQ	Call Qualifier
DAA	Data Access Arrangement
DB	Decibel
DSP	Digital Signal Processor or Digital Signal Processing
DTMF	Dual Tone Multiple Frequency
FIR	Finite Impulse Response digital filter
FSK	Frequency Shift Keying modulation
IDE	Integrated Development Environment
LPF	Low Pass Filter
MDMF	Multiple Data Message Format of GR-30-CORE
MIC	Microphone
MIPS	Million Instructions Per Second
OnCE™	On-Chip Emulation
PC	Personal Computer
PCM	Pulse Code Modulation
PSTN	Public Switched Telephone Network
SDK	Software Development Kit
SDMF	Single Data Message Format of GR-30-CORE
SRC	Source
VMWI	Visual Message Waiting Indicator

References

The following sources were referenced to produce this book:

1. *Motorola DSP56800E Reference Guide*, DSP56800ERM/D
2. *Motorola DSP568xx User's Manual*, for the DSP device you're implementing
3. *Targeting Motorola DSP568xx Platform*, for the DSP device you're implementing
4. *Motorola Embedded SDK Programmer's Guide*
5. SR-3004, *Testing Guidelines for Analog Type 1, 2, and 3 CPE as Described in SR-INS-002726 (a module of ADSI, FR-12)*, Telcordia Technologies, January 1995.
6. GR-30-CORE, *LSSGR: Voiceband Data Transmission Interface Section 6.6 (a module of LSSGR, FR-64)*, Telcordia Technologies, December 1998.
7. GR-31-CORE, *LSSGR CLASS^{SM} Feature: Calling Number Delivery (FSD 01-02-1051) (a module of LSSGR, FR-64)*, Telcordia Technologies, June 2000.

8. GR-1188-CORE, *LSSGR CLASS℠ Feature: Calling Name Delivery Generic Requirements (FSD 01-02-1070) (a module of LSSGR, FR-64)*, Telcordia Technologies, December 2000.
9. GR-1401-CORE, *LSSGR CLASS℠ Feature: Visual Message Waiting Indicator Generic Requirements (FSD 01-02-2000) (a module of LSSGR, FR-64)*, Telcordia Technologies, June 2000.
10. *ITU-T Recommendation V.23*, (11/88) - 600/1200-baud modem standardized for use in the general switched telephone network.

Chapter 1
Introduction

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

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

Motorola's Generic Echo Canceller Library is a licensed software library for use on Motorola DSP56800E series processors. Please refer to the Software License Agreement in **Chapter 7** for license terms and conditions.

1.1 Quick Start

Motorola's Embedded SDK is targeted to a large variety of hardware platforms. To take full advantage of a particular hardware platform, use **Quick Start** from the **Targeting Motorola DSP5685x Platform** documentation.

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

Note: "DSP568xx" refers to the specific device for which you're developing, as shown in the preceding example.

1.2 Telephony Features Libraries

The Generic Echo Canceller Library is one of a set of Motorola Embedded SDK modules and applications consisting of the following:

- Type 1 Telephony Features Library
- Type 1 and 2 Telephony Features Library
- Type 1 and 2 Telephony Parser Library
- Full Duplex Speakerphone Library

- Generic Echo Canceller Library
- Feature Phone Application Software

These modules are designed to interoperate to provide all of the software necessary to implement a feature phone with full duplex speakerphone and Type 1 and 2 Caller ID functionality. By using some or all of these modules, several other types of telephony applications are also possible. Each module may also be used independently.

1.3 Overview of the Generic Echo Canceller Library

This library implements the algorithm for a Generic Echo Canceller in Customer Premises Equipment (CPE). The Generic Echo Canceller, along with other modules like the Type 1 Telephony Features Library, the Type 1 and 2 Telephony Features Library, and the Full Duplex Speakerphone Library, all from Telcordia Technologies, provide for a complete set-up for telephony applications. The Generic Echo Canceller is used for Type 2 Caller ID and for speakerphone applications, as well as for other applications that may require echo cancellation.

1.3.1 Background

The source of echoes in voice transmission can be understood by considering a simplified connection between two subscribers. As shown in **Figure 1-1**, User 1 is on the left and User 2 is on the right.

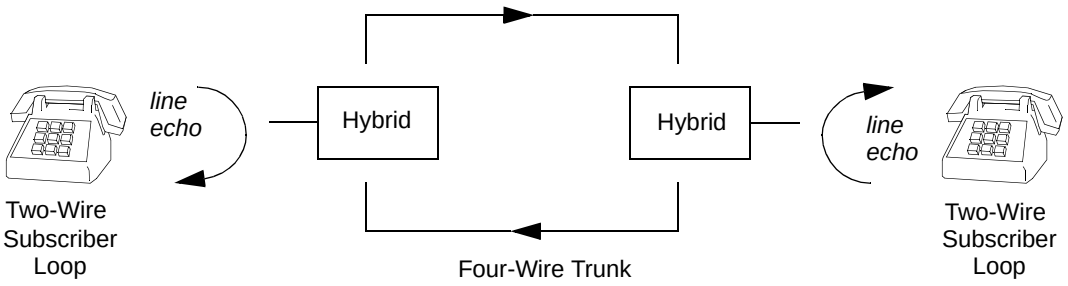

Figure 1-1. Simplied Telephone Connection

The role of the Hybrid is to convert a two-wire transmission system into a four-wire transmission system, or vice versa, without any impedance mismatches, to achieve complete transmission of signal in a single direction. However, in real practice, that cannot always be completely achieved. There is a fraction of the signal that returns to the sender, creating a line echo. It becomes imperative to cancel this echo for various purposes in telephony applications, such as detection of call progress tones or for speakerphones. As the exact nature of the echo is impossible to predetermine, an adaptive filter approach is adopted to achieve cancellation. The Generic Echo Canceller Library uses Finite Impulse Response (FIR) adaptive digital filters, in which the nature of the echo is assumed to be linear.

1.3.2 Features and Performance

The Generic Echo Canceller Library provides a flexible, complete, general-purpose echo cancellation system. Two streams of 8kHz samples are used as the input to the module. The first stream is the reference signal to be output to the echo interface. The second stream is an input from the echo interface, which contains both an echo of the reference and an additive-independent input signal. The echo canceller is capable of achieving an echo attenuation of greater than 40dB (under white noise input), while the independent input signal remains unchanged. It uses a normalized LMS algorithm to adaptively train coefficients of the echo canceller filter, which performs particularly well in situations where speech is used. The module contains a Voice Activity Detection algorithm to determine which side (line or audio) is talking. It also has a divergence detection algorithm to avoid unstable situations such as hunting. The Generic Echo Canceller Library works with the Full Duplex Speakerphone Library as a line echo canceller for achieving all functionalities of a natural-sounding speakerphone and can be used for echo cancellation in any other situation. The module provides for easy-to-use, variable tail length echo cancellation, ranging from 8ms to 64ms of tail length, in increments of 8ms.

Table 1-1 details the memory and MIPS requirements for the Generic Echo Canceller Library. There are two versions of code, depending on whether the data structure and the modulo buffer used by the library are placed in internal memory or external memory, and the MIPS changes between the two versions. The MIPS also depends on the tail length chosen for the generic echo canceller. For every extra 8ms of tail length, approximately 0.768 MIPS if in Internal Memory, or approximately 1.28 MIPS if in External Memory, should be added to the MIPS displayed in **Table 1-1**. The MIPS in this table have been estimated for a tail length of 8ms for the generic echo canceller. Depending on the tail length, the Data RAM per instance changes. The buffer size shown in **Table 3-1** should be added to the memory requirement stated in **Table 1-1**.

Table 1-1. Generic Echo Canceller Library Memory and MIPS Requirements

	Program Memory (ROM) (16 bit words)	Data Memory (ROM) (16 bit words)	Data RAM (16 bit words)	Data RAM Per Instance (16 bit words)	MIPS
Generic Echo Canceller Library (Internal Memory)	680	NA	NA	580 + buffer[1]	10.88
Generic Echo Canceller Library (External Memory)	680	NA	NA	580 + buffer[1]	13.00

1.See Section 1.3.2 for details

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 2
Directory Structure

Note: "DSP568xx" refers to the specific device for which you're developing, as shown in **Chapter 1, "Introduction."** .

2.1 Required Core Directories

Figure 2-1 details required platform directories:

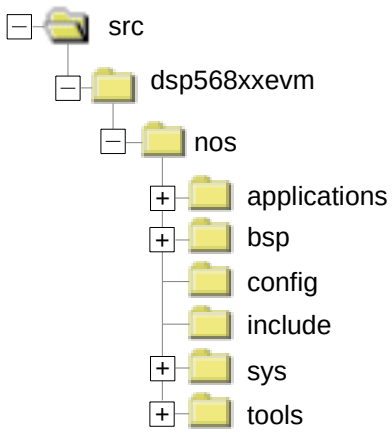

Figure 2-1. Core Directories

As shown in **Figure 2-1**, DSP568xxEVM includes a *no operating system support (nos)* directory, which includes the following core directories:

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

There are also optional directories that include domain-specific libraries.

2.2 Optional (Domain-Specific) Directories

Figure 2-2 demonstrates how the Generic Echo Canceller Library directory, *gec*, is encapsulated in the domain-specific directory, *telephony*.

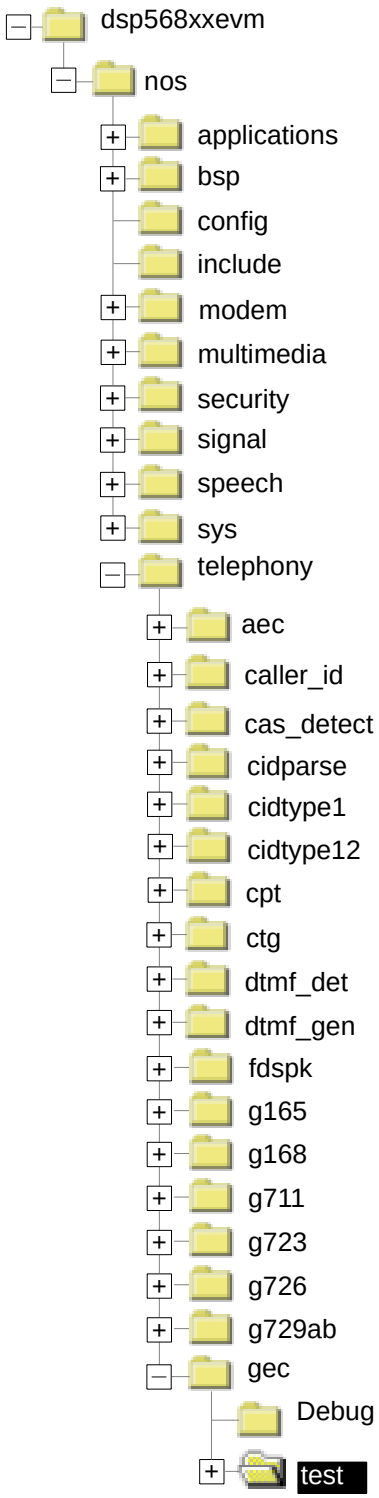

Figure 2-2. *telephony* Directory

The *telephony* directory contains specific telephony modules including vocoder algorithms such as G.728 or G.726, as well as the Generic Echo Canceller Library, the Caller ID Libraries, and others.

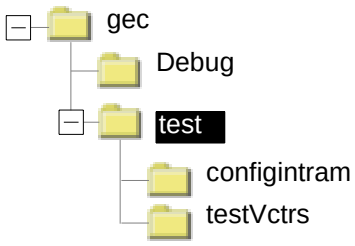

Figure 2-3. *gec* Directory Structure

The Generic Echo Canceller Library directory, *gec*, is shown in **Figure 2-3**. It includes:

- ***gec*** contains the Generic Echo Canceller Library test files and the pre-built library
  - ***Debug*** contains the *gec.lib*, which can be included in application projects that will use the procedures provided by this module
  - ***test*** contains a test project that includes test data, input data and results, and uses the library. This test project can be used to verify if the library is functioning correctly.
    - contains C source code for the text application, which is also useful as an example as to how the API is used, and can be modified to suit the application requirements
    - ***configintram*** contains the *linker.cmd* file using only internal memory for the entire project. It also contains *appconfig.h* and *appconfig.c*, which override the SDK's *config.h*.
    - ***testVctrs*** contains header files with test vectors for different tail lengths. They are used by the test application to validate the library code for different tail lengths.

Chapter 3
Generic Echo Canceller Library Interfaces

The Generic Echo Canceller Library is defined as:

gec.lib

The service, interface, and function calls included are described in this chapter.

3.1 Generic Echo Canceller Library Interface Services

The Generic Echo Canceller Library provides the procedures for the cancellation of echo in a system for applications; for example, the line echo on a telephone line for telephony applications. The output of the Generic Echo Canceller is typically used by a Caller ID module or a speakerphone module. Depending on the application, the output can also go unused. Because of this, the output of the module is given out seperately and the module itself does not overwrite the input samples that are echo-laden; i.e., that contain echoes.

3.2 Interface

All of the Generic Echo Canceller Library is written in the ASM assembly language. Before calling either of the Generic Echo Canceller functions, it is necessary to declare and fill required member variables of two structure types defined in *teldefs.h*, which are described below in more detail. The library requires declarations of the following structure *typedefs* for each instance, as well as a handle or pointer of type *gec_sData** defined in *gec.h*:

```
teldefs_sControl
teldefs_sSamples
```

The structures in *teldefs.h*, *teldefs_sControl* and *teldefs_sSamples*, will be accessed by the Generic Echo Canceller Library for necessary inputs. The *teldefs_sControl* structure contains all information provided by the application required by the Generic Echo Canceller Library; for example, whether the telephone application is using the echo canceller in a handset mode or in a speakerphone mode. The *teldefs_sSamples* structure contains samples from the line and audio interface and is to be filled by the application. The

output of the Generic Echo Canceller will be placed in a separate array, *gec[]*. It is left to the application or other modules using the same structures, as mentioned above, to determine what use is made of the echo-cancelled samples. A brief pertinent listing of *teldefs.h* is shown in **Code Example 3-1**.

Code Example 3-1. *teldefs.h* - Reference Definition for *gec*

```
#ifndef __TELDEFS_H
#define __TELDEFS_H

typedef struct teldefs_tsControl
{

      // phone state.
      int    hookSwitch;              //on hook or offhook
      int    handsFreeLayer1;         //indicates Speakerphone ON/OFF

      // ----------- Caller ID. ----------------------------
      // Caller ID related variables are declared here.

      // -----------  Generic Echo Canceller. ---------------
      int*   pgecCircularBuffer;      //pointer to modulo buffer
      int    gecLengthIndex;          //choice of tail length

      // -----------  FD Speakerphone -----------------------
      // FDSpeakerPhone related variables

      // -----------  Internal/Module use only ---------------
      // Variables that will be used to pass information between modules.


} teldefs_sControl;


typedef struct teldefs_tsSamples
{
      int    line[5];           //line input or audio output
      int    audio[5];          //audio input or line output
      int    gec[5];            //gec's output
      int    aec[5];            //fdspk's output
      int    voipinput[5];      //third port's input
      int    voipoutput[5];     //third port's output
      int    leccid[5];         //used by type2 to receive line
                                //echo cancelled samples.

} teldefs_sSamples;

#endif
```

The handle to the structure *gec_sData* is to be declared by the application. Memory space is allocated by the *gecEchoCancellerCreate()* function, which is used by the Generic Echo Canceller functions from that point on. It is esential that no other function, or the application itself, access the memory space allocated for this structure. The Generic Echo Canceller stores values that it will need to reuse, so tampering with the contents of the strucure will cause the function to report erroneous results.

The size of the circular buffer required by the Generic Echo Canceller depends on the length of the echo canceller; see **Table 3-1**. Before the module's initialization code is called, the pointer to this buffer must be given to the *teldefs_sControl* instance in the *pgecCircularBuffer* variable. All of this is done by the *gecEchoCancellerCreate()* function before the *gecEchoCancellerInit()* function is called. To ensure the data is only using internal memory to reduce the MIPS of this library, be sure the application has allocated sufficient dynamic internal memory through its *linker.cmd* file.

See header file *gec.h* in **Code Example 3-2** for a description of the function prototypes and structure instances which are used by the Generic Echo Canceller Library.

Code Example 3-2. *gec.h* - Reference Definition

```
#ifndef __GEC_H
#define __GEC_H


/**************************
 Foundational Include Files
**************************/
#include "gecf.h"

#ifndef __TELDEFS_H
#include "teldefs.h"
#endif


/**************************
 Function Prototypes
**************************/

extern void gecEchoCancellerInit(gec_sData*,teldefs_sControl*);
extern void gecEchoCanceller(gec_sData*,teldefs_sControl*,teldefs_sSamples*);
extern gec_sData* gecEchoCancellerCreate(teldefs_sControl*);
extern void gecEchoCancellerDestroy(gec_sData*,teldefs_sControl*);


/**************************
 Structures that must be
 defined before calling the
 functions. Examples shown.
**************************/
```

```
/*
teldefs_sControl    line1Control;       //their addresses sent
teldefs_sSamples    line1Samples;       //as arguments for the
gec_sData*          pgec1Data;          //functions listed above.
*/

/*The definition of gec_sData is listed in gecf.h which contains
various space allocators and housekeeping variables. However,
since these variables are not to be manipulated by an application
or other modules, the contents are not listed here. */

#endif
```

3.2.1 Variable Definition

This section offers a more detailed explanation of the variables in the structure *teldefs_tsControl* that an application using the Generic Echo Canceller Library must set. Some of the following are optional.

hookSwitch — Set by the application to indicate to the Generic Echo Canceller Library whether the phone is ON- or OFF-hook. For ON-hook it is set to 0; for OFF-hook, it is set to 1. This variable must be set each time the application detects or applies a physical change in or to the hookswitch state. The initial state also should be specified, implying there should never be an "unknown" state.

handsFreeLayer1 — Set by the application to indicate to the Generic Echo Canceller Library whether the speakerphone is currently ON or OFF. It is required so the Generic Echo Canceller Library can switch between the speakerphone mode and regular mode. When the speakerphone is ON this variable is set to 1; when the speakerphone is OFF, it is set to 0. This variable must be set each time the application detects or applies a physical change in or to the speakerphone state. The initial state also should be specified, implying there should never be an "unknown" state.

pgecCircularBuffer — This variable is set by the *gecEchoCancellerCreate()* function after dynamically allocating an appropriately-sized modulo buffer. The size of the buffer depends on the tail length that has been chosen for the echo canceller.

gecLengthIndex — Set by the application to indicate the desired tail length for the Generic Echo Canceller Library. The indices, and, correspondingly, the size of the buffer that is to be allocated, as well as the boundary on which it is to be aligned, are all listed in **Table 3-1**. This variable must be set before accessing any of the the module's API functions.

See **Table 3-1** for details on the possible Generic Echo Canceller filter and tail lengths, plus requirements for circular buffer size and the boundary on which they should be placed.

Table 3-1. Generic Echo Canceller Filter Lengths

Index	Filter Length	Tail Length	Circular Buffer Size	Boundary
0	64	8ms	72	0x80
1	128	16ms	144	0x100
2	192	24ms	216	0x100
3	256	32ms	288	0x200
4	320	40ms	340	0x200
5	384	48ms	408	0x200
6	448	56ms	476	0x200
7	512	64ms	544	0x400

3.3 Specifications

The following sections describe the Generic Echo Canceller Library functions.

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

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

3.3.1 *gecEchoCancellerCreate*

Call(s):

```
gec_sData* gecEchoCancellerCreate(teldefs_sControl* pControl);
```

Required Headers: *teldefs.h, gec.h*

Arguments:

Table 3-2. *gecEchoCancellerCreate* Arguments

pControl	*inout*	Points to the structure where input and output control information is stored

Description: This function allocates data memory for the data structure and modulo buffer for the Generic Echo Canceller Library, then calls the *gecEchoCancellerInit()* function. The prototype of this function is defined in *gec.h*. The return value of the function is a pointer to the data structure.

Before calling this function, the following variable in the control structure must be set by the application **according to requirements** for the length of filter for the echo canceller.

```
line1Control.gecLengthIndex
```

3.3.2 *gecEchoCancellerDestroy*

Call(s):

```
int gecEchoCancellerDestroy(gec_sData* pData,
                            teldefs_sControl* pControl);
```

Required Headers: *teldefs.h,gec.h*

Arguments:

Table 3-3. *gecEchoCancellerDestroy* Arguments

pData	*inout*	Points to the structure where the Generic Echo Canceller Library static data is stored
pControl	*inout*	Points to the structure where input and output control information is stored

Description: This function deallocates memory that was used by the current *gec* instance characterized by the *gec_sData* pData* pointer. The prototype of this function is defined in *gec.h*.

3.3.3 *gecEchoCancellerInit*

Call(s):

```
void gecEchoCancellerInit(gec_sData* pData, teldefs_sControl* pControl)
```

Required Headers: *teldefs.h, gec.h*

Arguments:

Table 3-4. *gecEchoCancellerInit* Arguments

pData	*inout*	Points to the structure where Generic Echo Canceller Library static data is stored
pControl	*inout*	Points to the structure containing control variables related to the telephony application that the Generic Echo Canceller Library module must know of; contains Input and Output Control Information

Description: The *gecEchoCancellerInit* function performs an initialization of all necessary variables in both the *teldefs_sControl* and *gec_sData* structures.

3.3.4 *gecEchoCanceller*

Call(s):

```
void gecEchoCanceller(gec_sData* pData, teldefs_sControl* pControl,
            teldefs_sSamples* pSamples)
```

Required Headers: *teldefs.h*, *gec.h*

Arguments:

Table 3-5. *gecEchoCanceller* Arguments

pData	*inout*	Points to the structure where Generic Echo Canceller Library static data is used and then returned for storage after computations
pControl	*inout*	Points to the structure where the Generic Echo Canceller Library can access input control variables and write to variables that are used by other modules and the application
pSamples	*inout*	Points to the structure where the voice samples are stored for the Generic Echo Canceller Library to access and where the library writes back the echo cancelled output

Description: This function takes in the audio (input reference stream) and line (echo and independent input stream) samples as input signals and cancels the echo present. It processes the samples coming at both ends, makes a binary decision of the side talking and, based on the decision, chooses to train the coefficients or to leave them unchanged. It also performs the divergence detection that is required for the echo canceller.

The echo cancelled samples are placed in the array *gec[]* in the structure of type *teldefs_sSamples*. A sample application exercising the Generic Echo Canceller Library, including the use of this library function, is found in **Code Example 3-3**.

Code Example: This section offers an example of how the Generic Echo Canceller Library is to be used in products.

Code Example 3-3. Use of Generic Echo Canceller Interface

```
#include "teldefs.h"
#include "gec.h"

struct gec_tsData* pgec1Data;
struct teldefs_tsControl line1Control;
struct teldefs_tsSamples  line1Samples;

void main(void)
{
      int i;

      // initialize values required before calling create() routine.
      line1Control.gecLengthIndex = 0;       // choice of tail length 8 ms.
      pgec1Data = gecEchoCancellerCreate (&line1Control);
```

```
        // specify initial state of phone device.
        line1Control.hookSwitch = 0;            // ON hook.
        line1Control.handsFreeLayer1 = 0;       // speakerphone OFF.

        while(1) {
              for(i=0;i<5;i++) {
                    line1Samples.audio[i] = stream1[i]; // reference signal
                    line1Samples.line[i] = stream2[i];  // echo+independentinput
              }

              for(i=0;i<5;i++) {
                    // call gecEchoCanceller only if off hook
                    if(line1Control.hookSwitch = 1)
                          gecEchoCanceller(&gec1Data,&line1Control,&line1Samples);


                    // output will be in line1Samples.gec[i] - to be used by
                    // application as required.
              }

              // monitor hook switch and speakerphone state change
              // and change line1Control.hookSwitch and
              // line1Control.speakerphone accordingly.

              // get 5 new samples in stream1[] and stream2[]
        }

}
```

Chapter 4
Building the Generic Echo Canceller Library

4.1 Building the Generic Echo Canceller Library

The Generic Echo Canceller Library combines all components described in the previous sections into one library: *gec.lib*. The library is pre-built for users, so no project is provided to be built. The library is located in the *lib* folder in the *.../telephony/gec* directory of the SDK directory structure.

Figure 4-1 shows how to build a project using the library. This project is located in the */nos/applications/telephony/* directory where other applications are also built in their respective folders under */applications*. The project is a speakerphone application that requires use of the Generic Echo Canceller Library as a Line Echo Canceller.

.

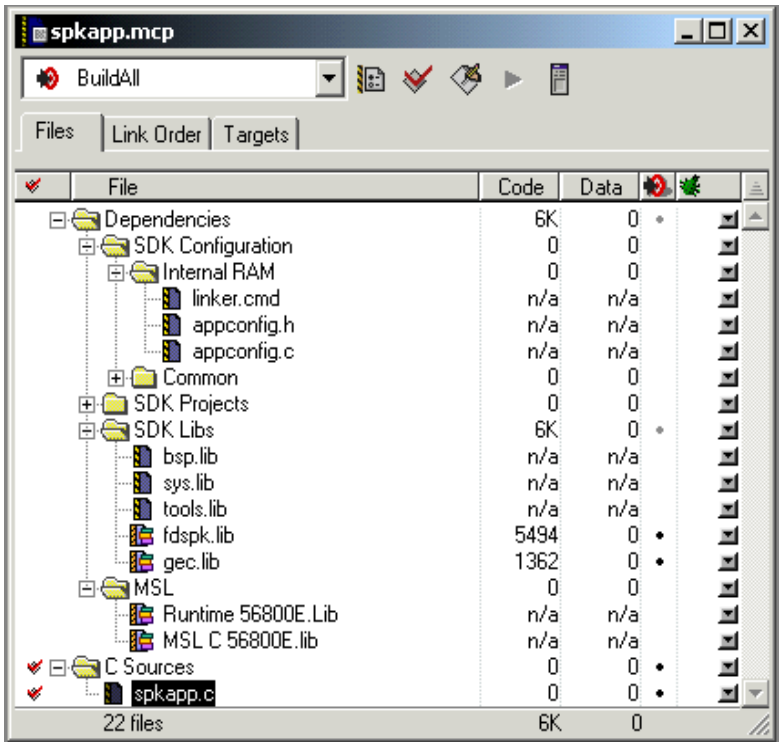

Figure 4-1. Example of a *gec* Library Link to a Project

Chapter 5
Linking Applications with the Generic Echo Canceller Library

5.1 Generic Echo Canceller Library

The Generic Echo Canceller library consists of a complete implementation of a flexible echo canceller with variable tail lengths and voice activity detection. It provides an API to use the echo canceller algorithm. The implementation is re-entrant and for every instance of run, the structures containing static variables for the generic echo canceller, *teldefs_sControl, gec_sData,* and *teldefs_sSamples,* must be instantiated. The following API's provide the interface between the user application and the Generic Echo Canceller Library:

- *gecEchoCancellerCreate(....)*
- *gecEchoCancellerDestroy(....)*
- *gecEchoCancellerInit(....)*
- *gecEchoCanceller(....)*

5.1.1 Library Sections

For a single instance, the data memory requirements for the Generic Echo Canceller Library are:

- 580 words for the *gec_sData* structure
- A circular buffer of a size that depends on the choice of filter length (**Table 3-1**)

The data could be placed dynamically in internal or external memory, affecting only the MIPS performance of the module and no other functionality.

For all instances, the program memory requirements for the Generic Echo Canceller Library are:

- 680 words for the entire module

To link in the Generic Echo Canceller Library's program space, *gec.text* must be included in the section that is decided by the application. **Code Example 5-1** contains an example linker file demonstrating the internal memory use and linking of *gec.text.*

Code Example 5-1. Example of a *linker.cmd* File for the Generic Echo Canceller Library

```
#*************************************************************************
#
# Linker.cmd file for DSP56858 External RAM
# using only external program and data memory.
#
#*************************************************************************

MEMORY {

            .pInterruptVector   (RWX):ORIGIN = 0x000000, LENGTH = 0x00008C
            .pIntRAM            (RWX):ORIGIN = 0x00008C, LENGTH = 0x009F74
            .pExtRAM            (RWX):ORIGIN = 0x00A000, LENGTH = 0x1E6000

            .pIntROM            (RX):ORIGIN = 0x1F0000, LENGTH = 0x000400
            .xIntRAM            (RW):ORIGIN = 0x000000, LENGTH = 0x005000
            .xIntRAM_DynamicMem (RW):ORIGIN = 0x005000, LENGTH = 0x001000

            .xStack             (RW):ORIGIN = 0x006000, LENGTH = 0x000800
            .xExtRAM_DynamicMem (RW):ORIGIN = 0x006800, LENGTH = 0x001000
            .xExtRAM            (RW):ORIGIN = 0x000000, LENGTH = 0x005000

            .xPeripherals       (RW):ORIGIN = 0x1FFC00, LENGTH = 0x000400
            .xExtRAM2           (RW):ORIGIN = 0x200000, LENGTH = 0xDFFF00

            .xCoreRegisters     (RW):ORIGIN = 0xFFFF00, LENGTH = 0x000100
}

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

FORCE_ACTIVE {FconfigInterruptVector}

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

SECTIONS {

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

      .ApplicationInterruptVector:
            {
                  vector.c (.text)

            }>.pInterruptVector

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

            .ApplicationCode:
            {
      # Place all code into Program RAM

                  *(.text)
                  *(rtlib.text)
                  *(fp_engine.text)
                  *(user.text)

                  # Place all data into Program RAM
```

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

        F_Pbss_start_addr = .;

                 _P_BSS_ADDR = .;
                 pramdata.c (.bss)

                F_Pbss_length = .-_P_BSS_ADDR;

        }>.pExtRAM

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

        .GECLibraryCode:
        {
# Place cid1 code into Program Internal RAM

                *(gec.text)

        }>.pIntRAM

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

        .ApplicationData:
        {
                # Define variables for C initialization code

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

                # Define variables for SDK mem library

                # Data (X) memory Layout

                      _EX_BIT = 0;

                 # Internal Memory Partitions (for mem.h partitions)

                      _NUM_IM_PARTITIONS = 0; # IM_ADDR_1 (no IM_ADDR_2)

                 # External Memory Partition (for mem.h partitions)

                      _NUM_EM_PARTITIONS = 1; # EM_ADDR_1

                 FmemEXbit =.;
                      WRITEH(_EX_BIT);
                 FmemNumIMpartitions =.;
                      WRITEH(_NUM_IM_PARTITIONS);
                 FmemNumEMpartitions =.;
                      WRITEH(_NUM_EM_PARTITIONS);
                 FmemIMpartitionList =.;
                      WRITEH(ADDR(.xIntRAM_DynamicMem)*1);
                      WRITEH(SIZEOF(.xIntRAM_DynamicMem)*1);
                 FmemEMpartitionList =.;
                      WRITEH(ADDR(.xExtRAM_DynamicMem)*1);
                      WRITEH(SIZEOF(.xExtRAM_DynamicMem)*1);

                # Add rest of the data into External RAM

                 *(.const.data)
                 *(.data)
```

```
                    *(fp_state.data)
                    *(rtlib.data)

                   F_Xdata_ROMtoRAM_length = 0;

                   F_Xbss_start_addr =.;
                   _X_BSS_ADDR =.;
                    *(rtlib.bss.lo)
                    *(rtlib.bss)
                    *(.bss)

                   F_Xbss_length =.-_X_BSS_ADDR; # Copy DATA

      }>.xExtRAM

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

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

}
```

As mentioned previously, the module requires a circular buffer (also called a modulo buffer) of a size that depends on the length of the filter chosen by the application (see **Table 3-1**). The buffer and the data structure (*gec_sData*) are dynamically allocated by the *gecEchoCancellerCreate()* function. The application should ensure sufficient data memory for dynamic allocation in either Internal Memory (preferred for lesser MIPS) or External Memory through the linker command file's *xIntRAM_DynamicMem* or *xExtRAM_DynamicMem*.

Chapter 6
Generic Echo Canceller Library Applications

Two example applications are discussed in this section; a speakerphone application and a test application.

6.1 Generic Echo Canceller Library Application

A Full Duplex Speakerphone application has been provided for use on a DSP56858 EVM board using the *gec.lib* library. Modifications can be made to suit the application. The application uses codec drivers and serial port drivers that are part of the SDK and is located in the */nos/applications/fdspk* directory.

6.2 Test Application

A test application has also been provided to verify the functionality of the module. To run it, the user can open the *testgec.mcp* project in the *nos/telephony/gec/test* directory. Example code for this application is given in **Code Example 6-1**.

The filter *gecLengthIndex* is set to 0 and the *correctCoeffs* is equated to *gecCoeffs0.h*. The user can change both the *gecLengthIndex* and *correctCoeffs* to the same number to test for different filter lengths; e.g., *gecLengthIndex = 4* and *correctCoeffs = #include gecCoeffs4.h*. The application is meant to run the module for 800 samples worth of white noise and return a “PASS” or “FAIL” result, depending on the run. The result is printed on the console screen. This application runs digitally, implying it requires no other modules.

Code Example 6-1. testapp.c

```
/* File: main.c */
#include <stdio.h>
#include “port.h”
#include “teldefs.h”
#include “gec.h”

int lineSamples[800]={
#include “gecLine.h”
};
```

```
int audioSamples[800]={
#include “gecCoeffs0.h”
};

int compare;
int done;
int cntr;
int index;

/* module related declarations */
teldefs_sControl          line1Control;
teldefs_sSamples          line1Samples;
gec_sData*                pgec1Data;

int main(void)
{
      int i;
      int fail = 0;

      printf(“Generic Echo Canceller Test...\n”);
      cntr = 0;
      compare = 0;
      done = 0;
      index = 0;

      line1Control.gecLengthIndex = 0;       // choosing filter length
      pgec1Data = gecEchoCancellerCreate(&line1Control);

      line1Control.hookSwitch = 1;
      line1Control.handsFreeLayer1 = 0;

      // All for test mode. Do not use in regular mode.
      line1Control.disableAnalysis = 1;
      line1Control.pre2lecPowMsw = 1000;
      line1Control.dtHang = -1;
      line1Control.nesPresent = 1;
      line1Control.trainLec = 1;

      While(!done) {
            for(i=0; i<5; i++){
                  line1Samples.line[i] = lineSamples[index];
                  line1Samples.audio[i] = audioSamples[index];
                  index++;
            }

            for(i=0; i<5; i++) {
                  line1Control.lecDelaySample = line1Samples.line[i];
                  if (line1Control.hookSwitch == 1)      {
                        gecEchoCanceller(pgec1Data,&line1Control,&line1Samples);
                        if(line1Control.trainLec == 0)
                              asm(debughlt);
                  }
            }

            cntr++;

            if(cntr >= 160){
                  compare = 1;
                  cntr = 0;
            }
```

```
                if(compare == 1){
                        for(i=0, i<64, i++){
                                if(correctCoeffs[i] != pgec1Data -> lecSpkcoefs[i]){
                                        fail = 1;
                                }
                        }
                        done = 1;
                }
        }
        gecEchoCancellerDestroy(pgec1Data, &line1Control);
        if (fail == 0) printf("PASS \n");
        else printf("FAIL \n");

        return 0;
}
```

Should the application print “FAIL”, this would indicate that the Generic Echo Canceller Library is not operating correctly. This is not expected to happen for the test application; however, if it does, please report the failure to Motorola for resolution.

Chapter 7
License

7.1 Limited Use License Agreement

This software is available under a separate license agreement from Motorola Incorporated. Licensing information can be obtained from your Motorola sales representative or authorized distributor.

For additional product information, see **http://www.motorola.com/semiconductors/**.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Index

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

L

M

N

O

P

S

T

V

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors/