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

Embedded SDK (Software Development Kit)

G.168 Line Echo Canceller Library

SDK132/D
Rev. 1, 07/19/2002

MOTOROLA intelligence everywhere™ | digital dna™

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

Contents

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

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 Examples

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

About This Document

This manual describes the G.168 Line Echo Canceller algorithm for use with Motorola's Embedded Software Development Kit, (SDK).

Audience

This document targets software developers implementing echo cancellation functions within software applications.

Organization

This manual is arranged in the following sections:

- **Chapter 1, Introduction**—provides a brief overview of this document
- **Chapter 2, Directory Structure**—provides a description of the required core directories
- **Chapter 3, G.168 Library Interfaces**—describes all of the G.168 Library functions
- **Chapter 4, Building the G.168 Library**—tells how to execute the system library project build
- **Chapter 5, Linking Applications with the G.168 Library**—describes organization of the G.168 Library
- **Chapter 6, G.168 Applications**—describes the use of G.168 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:

- *DSP56800 Family Manual*, DSP56800FM/AD
- *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`	Commands, command parameters, code examples, expressions, datatypes, and directives	...`*Foundational include files`... ...a data structure of type `vad_tConfigure`...
Italic	Calls, functions, statements, procedures, routines, arguments, file names and applications	...the *pConfig* argument... ...defined in the C header file, *aec.h*... ...makes a call to the *Callback* procedure...
Bold	Reference sources, paths, emphasis	...refer to the **Targeting DSP56824 Platform** manual.... ... see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
Bold/Italic	Directory name, project name	...and contains these core directories: ***applications*** contains applications software.... ...CodeWarrior project, ***3des.mcp***, is.....
Blue Text	Linkable on-line	...refer to **Chapter 7**, License
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}
ALL CAPITAL LETTERS	Variables, directives, defined constants, files libraries	INCLUDE_DSPFUNC #define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]...
Quotation marks "... "	Returned messages	...the message, "Test Passed" is displayed.... ...if unsuccessful for any reason, it will return "NULL"....

Definitions, Acronyms, and Abbreviations

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

DSP Digital Signal Processor or Digital Signal Processing

FFT Fast Fourier Transforms

FIR Finite Impulse Response

HRL Hold Release logic

I/O	Input/Output
IDE	Integrated Development Environment
IIR	Infinite Impulse Response
LMS	Least Mean Square
LSB	Least Significant Bit
MAC	Multiply/Accumulate
MIPS	Million Instructions Per Second
MSB	Most Significant Bit
NLMS	Normalized Least Mean Square
OnCE™	On-Chip Emulation
OMR	Operating Mode Register
PC	Personal Computer
RLS	Recursive Least Squares
SDK	Software Development Kit
SP	Stack Pointer
SPI	Serial Peripheral Interface
SR	Status Register
SRC	Source
TD	Tone Disabler

References

The following sources were used to produce this book:

1. *DSP56800 Family Manual*, DSP56800FM/AD
2. *DSP568xx User's Manual*, for the DSP device you're implementing
3. *Embedded SDK Programmer's Guide*
4. *ITU-T Recommendation G.168*, pre-published version, 04/2001

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

Chapter 1
Introduction

Welcome to Motorola's Family of Digital Signal Processors (DSPs). This document describes the G.168 Echo Canceller Library, which is a part of Motorola's comprehensive Embedded Software Development Kit (SDK) for its DSPs. In this manual, you will find all the information required to use and maintain the G.168 Echo Canceller Library interface and algorithms.

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

Motorola's G.168 Echo Canceller Library is licensed for your use on Motorola processors. Please refer to the standard Software License Agreement in **Chapter 7** for license terms and conditions; please consult with your Motorola representatie for premium product licensing.

1.1 Quick Start

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

For example, the **Targeting DSP56824 Platform** manual provides more specific information and examples about this hardware architecture. If you are developing an application for an DSP56824EVM board or any other DSP56824 development system, refer to the **Targeting DSP56824 Platform** manual for **Quick Start** or other DSP56824-specific information.

1.2 Overview of G.168

G.168-based echo cancellers conform to the ITU-T Recommendation G.168, previously known as the CCITT Recommendation.

Echo cancellers are voice-operated devices placed in the four-wire portion of the circuit and reduce the echo by subtracting the echo estimate from the circuit echo. In this usage, the echo cancellers are assumed to be "half" echo cancellers; i.e., those in which cancellation takes place only in the circuit's send path, due to signals present in the receive path.

1.2.1 Background

Electrical echo cancellers are widely used in full duplex modems, in which the transmit signal and the receive signal are sent over the same pair of telephone wires. A hybrid unit is required to convert four-wire communication into two-wire communications and vice-versa. These hybrid units are located in the telephone sets, as well as at various telephone exchanges. Because of the non-stationary nature of various communication links, the hybrids cannot be perfectly tuned using hardware adjustments. Thus, a part of the received signal is reflected as echo, known as the electrical echo. Electrical echo cancellers are designed to remove this echo from the transmitted signal and are categorized into near-end echo cancellers and far-end echo cancellers.

Echo canceller is an adaptive FIR filter, which synthesizes the replica of the echo path impulse response. The performance of the echo canceller depends on the linearity of the echo path between the receive path and the send path. Non-linearity will cause degradation in the echo cancellation. As the echo canceller is a linear filter, it will cancel only the linear component of the echo.

There are various mean-squared, error-based methods for echo cancellation, including Least Mean Square, (LMS); Normalized LMS, (NLMS); and Recursive Least Squares, (RLS). Considering complexity, an LMS-based methods is more efficient than an RLS-based method. In the G.168 library, the NLMS algorithm is used.

Basic requirements for echo cancellers are:

- Rapid convergence
- Low echo return level during single talk
- Low divergence during double talk

1.2.2 Features and Performance

The G.168 library is multichannel and re-entrant.

For details on Memory and MIPS for a particular DSP, refer to the **Libraries** Chapter of the appropriate Targeting manual.

Chapter 2
Directory Structure

2.1 Required Core Directories

Figure 2-1 details required platform directories:

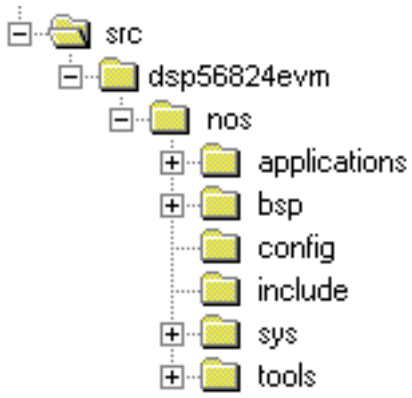

Figure 2-1. Core Directories

In this example, the DSP56824EVM has no operating system (nos) support. This platform contains 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

Also, there are some optional directories that include domain-specific libraries.

2.2 Optional (Domain-Specific) Directories

Figure 2-2 demonstrates how the G.168 algorithm is encapsulated in the domain-specific directories under the directory ***telephony***.

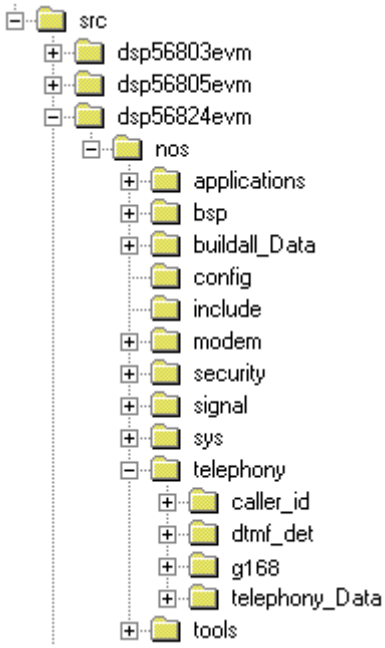

Figure 2-2. DSP56824 Directories

The ***telephony*** directory includes telephony-specific algorithms. **Figure 2-3** shows the ***g168*** directory structure under ***telephony*** directory.

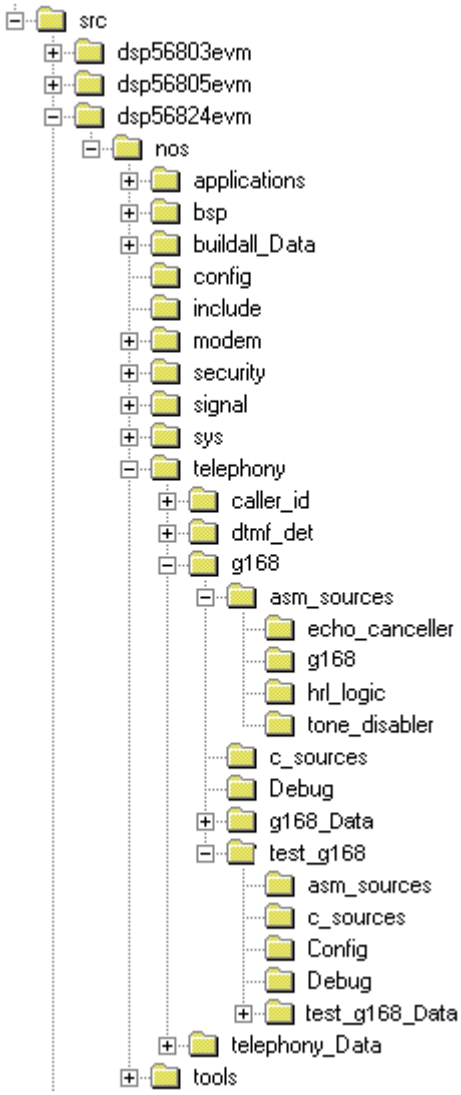

Figure 2-3. *telephony* Directory Structure

The ***g168*** directory includes the following sub-directories:

- ***asm_sources*** includes all asm sources required for echo canceller, hold release logic, tone disabler logic and complete integration module for G.168
- ***c_sources*** includes APIs for G.168
- ***test_g168_Data*** includes C sources and configuration necessary for testing G.168 library modules
  - — ***c_sources*** contains an example test code for the requirements mentioned in design document
  - — ***Config*** contains configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to G.168

The ***applications*** directory includes high-level software that exercises the G.168 library. As shown in **Figure 2-4**, the ***applications*** directory contains the ***g168*** application under ***telephony***.

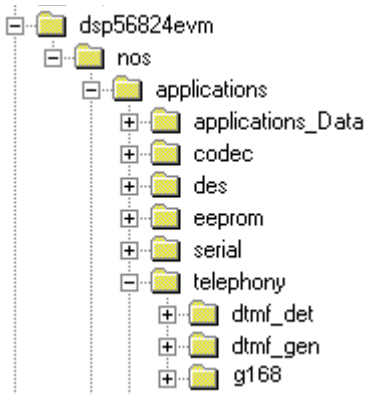

Figure 2-4. G.168 Application

Chapter 3
G.168 Library Interfaces

3.1 G.168 Services

The G.168 library cancels the echo from the near-end speech signal with the use of reference (far-end speech) signal. The data to be supplied must be in 16-bit word fixed-point (1.15) format, which is shown below.

i = information bit
s = sign

3.2 Interface

The C interface for the G.168 library services is defined in the C header file *g168.h,* shown in **Code Example 3-1**.

Code Example 3-1. C Header File *g168.h*

```
/* File: g168.h */

#ifndef __G168_H
#define __G168_H
/*
   Electronic Echo Cancellation (G.168) interface
*/
/***************************
 Foundational Include Files
***************************/
```

```
#include "port.h"


/*****************************************
 #define for G.168 configuration flags.
 Please refer to the user document for more
 details
*****************************************/
#define G168_CONFIG_NL_OPTION              1
#define G168_CONFIG_DISABLE_TONE_DETECTION 2
#define G168_CONFIG_INHIBIT_CONVERGENCE    4
#define G168_CONFIG_RESET_COEFFICIENTS     8
*****************************************
     Structure for G.168 Configuration. This
     structure is used by G168 library and user
     should not alter neither any element nor
     disturb the order of the elements
*****************************************/
typedef struct
{
// Echo Canceller Variables
    Int16  *FilterStates;
    Int16  *HFilt;
    Int16  *HFiltBak1;
    Frac16 LengthFactor;
    Int16  TrainLevel;
    Int16  NLSupress;
    UInt16 FrameCounter;
    Int16  SinEnergyHigh;
    Int16  SinEnergyLow;
    Int16  RinEnergyHigh;
    Int16  RinEnergyLow;
    Int16  SoutEnergyHigh;
    Int16  SoutEnergyLow;
    Int16  RinSample;
    Int16  SinSample;
    Int16  SoutSample;
    Int16  Mu;
    Int16  MuBase;
    Int16  EcFrameFull;
    UInt16 DisableTD;
    UInt16 InhibitConvergence;
    UInt16 DoubleTalk;
    UInt16 DontAdapt;
```

```
    UInt16 ResetCoeff;
    UInt16 NLOption;
    UInt16 G168ECEnable;
    UInt16 ChangeFlag;
    Int16  FilterStatePtr;
    Int16  HFiltPtr;
    Int16  HFiltBak1Ptr;
    UInt16 EchoSpan;
    UInt16 FiltLen;


// Tone Disabler Variables


   // Receive Channel specific variables

    Int16  *ZcAmps1;
    Int16  *AlpStates1;
    Int16  *BlpStates1;
    Int16  DontAdapt1;
    Int16  StateTD1;
    UInt16 ResetTD1;
    Int16  ToneCount1;
    UInt16 G168ToneDisable1;
    Int16  Goertzel1[2];
    Int16  TDFrmEnergy1High;
    Int16  TDFrmEnergy1Low;
    Int16  GoertzelCount1;
    Int16  AvgTDTone1High;
    Int16  AvgTDTone1Low;
    Int16  AvgTDNoise1High;
    Int16  AvgTDNoise1Low;
    Int16  TonePassCount1;
    Int16  PhRevAmp1;
    Int16  PhRevInst1;
    Int16  PhRevAmp11;
    Int16  PhRevAmp21;
    Int16  PhRevFlag1;
    Int16  PhRevInst11;
    Int16  PhRevInst21;
    Int16  FirstZcFlag1;
    Int16  Count1;
    Int16  ZeroCross1;
    Int16  HfPeriod1;
    Int16  NumHfPeriod1;
    Int16  SumHfPeriod1;
    Int16  HfPeriodLim1;
    Int16  FCount1;
    Int16  ZcCount1;
    Int16  SumZc1;
    Int16  ZcAmps1Ptr;
    Int16  AlpStates1Ptr;
    Int16  BlpStates1Ptr;
    Int16  SineP1;
```

```
   // Send Channel specific variables

    Int16  *ZcAmps2;
    Int16  *AlpStates2;
    Int16  *BlpStates2;
    Int16  DontAdapt2;
    Int16  StateTD2;
    UInt16 ResetTD2;
    Int16  ToneCount2;
    UInt16 G168ToneDisable2;
    Int16  Goertzel2[2];
    Int16  TDFrmEnergy2High;
    Int16  TDFrmEnergy2Low;
    Int16  GoertzelCount2;
    Int16  AvgTDTone2High;
    Int16  AvgTDTone2Low;
    Int16  AvgTDNoise2High;
    Int16  AvgTDNoise2Low;
    Int16  TonePassCount2;
    Int16  PhRevAmp2;
    Int16  PhRevInst2;
    Int16  PhRevAmp12;
    Int16  PhRevAmp22;
    Int16  PhRevFlag2;
    Int16  PhRevInst12;
    Int16  PhRevInst22;
    Int16  FirstZcFlag2;
    Int16  Count2;
    Int16  ZeroCross2;
    Int16  HfPeriod2;
    Int16  NumHfPeriod2;
    Int16  SumHfPeriod2;
    Int16  HfPeriodLim2;
    Int16  FCount2;
    Int16  ZcCount2;
    Int16  SumZc2;
    Int16  ZcAmps2Ptr;
    Int16  AlpStates2Ptr;
    Int16  BlpStates2Ptr;
    Int16  SineP2;

    Int16  disableTD;


// Hold Release Logic Variables

    Int16  AvgHRLtoneHigh;
    Int16  AvgHRLtoneLow;
    Int16  AvgHRLnoiseHigh;
    Int16  AvgHRLnoiseLow;
    Int16  HRLfrmFull;
    Int16  ReleaseFlag;
    Int16  ReleaseCount;
```

```
    Int16  FrameCount;
    Int16  PrevKmax;
    Int16  PrevToneChange;
    Int16  *BufA;
    Int16  BufPtr;
    Int16  FrameBufPtr;
    Int16  Temp[4];

    UInt16 SkipCount;

} ECDataStruct;


/* This structure has to be initialized by the user
   before calling the g168Create function */
typedef struct
{
      UInt16            Flags;    /* takes the values from
                                     #defines described above */
      UInt16            EchoSpan; /* Refer to the user document
                                     for more details */
} g168_sConfigure;


/* This is a handle for G168 instance. User should
   not alter any member of the structure as it is maintained
   by G168 library */
typedef struct
{
      UInt16          Flags;
      ECDataStruct   * EchoCancellerData;
}g168_sHandle;
/*****************************************
     Commands for G.168 Control
*****************************************/
#define G168_INHIBIT_CONVERGENCE 2
#define G168_RESET_COEFFICIENTS  4
#define G168_REENABLE_CONVERGENCE 8
/*************************
 Function Prototypes
*************************/
EXPORT g168_sHandle * g168Create (g168_sConfigure * pConfig);
EXPORT Result g168Init (g168_sHandle * pG168,
```

```
                          g168_sConfigure * pConfig);
EXPORT Result g168Process (g168_sHandle * pG168,
                           Int16 *pSamples_Rin,
                           Int16 *pSamples_Sin,
                           Int16 *pSamples_Sout,
                           UInt16 NumSamples);
EXPORT UWord16 g168Control (g168_sHandle * pG168, UInt16 Command);

EXPORT void g168Destroy (g168_sHandle * pG168);

#endif
```

3.3 Specifications

The following pages describe the G.168 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 function. 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 *g168Create*

Call(s):

```
g168_sHandle *g168Create (g168_sConfigure *pConfig);
```

Required Header: *g168.h*

Arguments:

Table 3-1. *g168Create* Arguments

pConfig	in	Points to the configuration data for G.168

Description: The *g168Create* function creates an instance of the G.168. The *pConfig* argument points to the *g168_sConfigure* structure which configures the G.168 operation; for additional information, see *g168Init*, **Section 3.3.2**. Multiple instances are possible. During the *g168Create* call, any dynamic resources required by the G.168 algorithm are allocated; each call of *g168Create* allocates 282 + 3*EchoSpan (EchoSpan = number of filter taps at 8KHz sampling) words of data memory. The library allocates dynamic memory using the *mem* library routines shown in **Code Example 3-2**.

Code Example 3-2. *mem* Library

```
#include "mem.h"
#include "g168.h"
#define HRL_FRM_LEN 128

g168_sHandle * g168Create (g168_sConfigure * pConfig)
{
    g168_sHandle *pG168;
    bool IsInternalMem = true;
    bool IsMemAligned = true;



    /*Allocate the memory for the handle */
    pG168 = (g168_sHandle *) memMallocEM (sizeof (g168_sHandle));
    if (pG168 == NULL) return (NULL);

     /* Allocate the memory for the EchoCancellerData Struct */

    pG168->EchoCancellerData = (ECDataStruct *) memMallocEM (sizeof (ECDataStruct));

    /* Allocate Circular Buffer for EC Filter States */
    pG168->EchoCancellerData->FilterStates = (Int16 *) memMallocAlignedIM
                                              ((pConfig->EchoSpan)*sizeof (Int16));
    /* Check whether the memory is internal */
    IsInternalMem &= memIsIM (pG168->EchoCancellerData->FilterStates);

    /* Check the alignment for the allocated buffer */
    IsMemAligned &= memIsAligned (pG168->EchoCancellerData->FilterStates,
                                  (pConfig->EchoSpan)*sizeof (Int16));
```

```
/* Allocate Buffer for EC Filter Coeffs */
pG168->EchoCancellerData->HFilt = (Int16 *) memMallocIM
                                        ((pConfig->EchoSpan)*sizeof (Int16));

/* Check whether the memory is internal */
IsInternalMem &= memIsIM (pG168->EchoCancellerData->HFilt);

/* Allocate the memory for H Filter Backup 1*/
pG168->EchoCancellerData->HFiltBak1 = (Int16 *) memMallocEM
                                        ((pConfig->EchoSpan)*sizeof (Int16));

/* Allocation of memory for HRL Data Structure */
pG168->EchoCancellerData->BufA = (Int16 *) memMallocEM ((HRL_FRM_LEN + 2) *
                                                    sizeof(Int16));

/* Allocation of memory for Tone Disabler Data Structure */

pG168->EchoCancellerData->ZcAmps1 = (Int16 *) memMallocAlignedEM
                                            (6*sizeof(Int16));
IsMemAligned &= memIsAligned (pG168->EchoCancellerData->ZcAmps1, 6*sizeof
                                                                (Int16));

pG168->EchoCancellerData->ZcAmps2 = (Int16 *) memMallocAlignedEM
                                            (6*sizeof(Int16));
IsMemAligned &= memIsAligned (pG168->EchoCancellerData->ZcAmps2, 6*sizeof
                                                                (Int16));

pG168->EchoCancellerData->AlpStates1 = (Int16 *) memMallocAlignedEM
                                            (3*sizeof(Int16));
IsMemAligned &= memIsAligned (pG168->EchoCancellerData->AlpStates1, 3*sizeof
                                                                (Int16));

pG168->EchoCancellerData->BlpStates1 = (Int16 *) memMallocAlignedEM
                                            (3*sizeof(Int16));
IsMemAligned &= memIsAligned (pG168->EchoCancellerData->BlpStates1, 3*sizeof
                                                                (Int16));

pG168->EchoCancellerData->AlpStates2 = (Int16 *) memMallocAlignedEM
                                            (3*sizeof(Int16));
IsMemAligned &= memIsAligned (pG168->EchoCancellerData->AlpStates2, 3*sizeof
                                                                (Int16));

pG168->EchoCancellerData->BlpStates2 = (Int16 *) memMallocAlignedEM
                                            (3*sizeof(Int16));
IsMemAligned &= memIsAligned (pG168->EchoCancellerData->BlpStates2, 3*sizeof
                                                                (Int16));

if (!(pG168->EchoCancellerData && pG168->EchoCancellerData->FilterStates &&
      pG168->EchoCancellerData->HFilt && pG168->EchoCancellerData->HFiltBak1 &&
      pG168->EchoCancellerData->BufA && pG168->EchoCancellerData->ZcAmps1 &&
    pG168->EchoCancellerData->ZcAmps2 && pG168->EchoCancellerData->AlpStates1 &&
    pG168->EchoCancellerData->BlpStates1 && pG168->EchoCancellerData->AlpStates2
    && pG168->EchoCancellerData->BlpStates2))
{
```

```
        g168Destroy (pG168);
        return (NULL);
    }

    if ((IsMemAligned & IsInternalMem) == false)
    {
        g168Destroy (pG168);
        return (NULL);
    }

    g168Init (pG168, pConfig);

    return(pG168);
}
```


For details on the *g168_sHandle* structure, please refer to **Code Example 3-1**. The preceding *pConfig* argument points to the *g168_sConfigure* structure, which configures G.168 operation as shown in **Code Example 3-3**.

If a *g168Create* function is called to create an instance, then *g168Destroy* (see **Section 3.3.5**) should be used to destroy the instance.

Alternatively, the user can allocate memory statically, which requires duplicating all statements in the *g168Create* function. In this case, the user can call the *g168Init* function directly, bypassing the *g168Create* function. If the user dynamically allocates memory without calling *g168Create*, then the user himself must destroy the memory allocated.

Returns: Upon successful completion, the *g168Create* function will return a pointer to the specific instance of G.168 created. If *g168Create* is unsuccessful for any reason, it will return “NULL”.

Special Considerations:

- The G.168 application is multichannel and re-entrant
- If *g168Create* is called, then the user need not call the *g168Init* function, which is called internally in the *g168Create* function

Code Example: In **Code Example 3-3**, the application creates an instance of G.168.

Code Example 3-3. Use of the *g168Create* Interface

```
#include “g168.h”
#include "mem.h"

void test_g168 (void)
{
g168_sHandle *pG168;
g168_sConfigure *pConfig;

pConfig = (g168_sConfigure *) memMallocEM(sizeof (g168_sConfigure));
```

```
    /* User configuration of G.168 */
    pConfig->Flags= 0;
    pConfig->EchoSpan = 320;
/* Create and init the instance of G.168 */
    pG168 = g168Create(pConfig);
}
```

3.3.2 *g168Init*

Call(s):

```
Result g168Init (g168_sHandle *pG168, g168_sConfigure *pConfig);
```

Required Header: *g168.h*

Arguments:

Table 3-2. *g168Init* Arguments

pG168	in	Handle to an instance of G.168.
pConfig	in	A pointer to a data structure containing data for initializing the G.168 algorithm

Description: The *g168Init* function will initialize the G.168 algorithm. During initialization, each resource will be set to its initial values in preparation for G.168 operation. Before calling the *g168Init* function, a G.168 instance must be created either by calling the *g168Create* function (see **Section 3.3.1**), or by statically allocating memory, which does not require calling the *g168Create* function.

The parameter *pConfig* points to a data structure of type *g168_sConfigure*; its fields initialize G.168 operation in the following manner:

Flags A set of configuration options for G.168. Flags options include:
G168_CONFIG_NL_OPTION, a flag for enabling non-linear processing in the *g168Process*
G168_CONFIG_DISABLE_TONE_DETECTION, which disables the 2100Hz tone detection in the *g168Process*
G168_CONFIG_INHIBIT_CONVERGENCE, which stops adapting the equalizer in the *g168Process*
G168_CONFIG_RESET_COEFFICIENTS, which resets all coefficients of the equalizer to zero in the *g168Process*

EchoSpan Specifies the number of taps of the equalizer. The valid range is upto 40 - 512 taps, which corresponds to echo tail of 5 - 64 ms.

Returns: Upon successful completion, a value of “TRUE” will be returned; otherwise, a value of “FALSE” will be returned.

Special Considerations:

- If *g168Create* is called, then the user need not call the *g168Init* function, which is called internally in the *g168Create* function

Code Example: In **Code Example 3-4**, the application creates an instance of G.168. The instance is passed to *g168Init* along with the G.168 configuration structure *pConfig*.

Code Example 3-4. Use of *g168Init* Interface

```
#include “g168.h”
#include "mem.h"

void test_g168 (void)
{
g168_sHandle *pG168;
g168_sConfigure *pConfig;
Result res;

pConfig = (g168_sConfigure *) memMallocEM(sizeof (g168_sConfigure));

    /* User configuration of G.168 */
    pConfig->Flags= 0;
    pConfig->EchoSpan = 320;

/* Statically Create the instance of G.168 */
    ...
res = g168Init (pG168, pConfig);
}
```

3.3.3 *g168Process*

Call(s):

```
Result g168Process ( g168_sHandle * pG168,
                     Int16 *pSamples_Rin,
                     Int16 *pSamples_Sin,
                     Int16 *pSamples_Sout,
                     UInt16 NumSamples);
```

Required Header: *g168.h*

Arguments:

Table 3-3. *g168Process* Arguments

pG168	in	Handle to an instance of G.168
pSamples_Rin	in	Pointer to the user data reference signal to be used by the G.168 algorithm
pSamples_Sin	in	Pointer to the user data for the send-path signal (echo of reference signal + far-end speech) for echo cancellation by the G.168 algorithm
pSamples_Sout	out	Pointer to the array of the echo-canceller output data
NumSamples	in	The number of data words to be processed

Description: The *g168Process* function will process the samples and cancel the echo from the data supplied by *pSamples_Sin*. The processed output is stored in the array pointed by *pSamples_Sout*. The user can call the *g168Process* function any number of times, as long as there is data.

Returns: Upon successful completion, *g168Process* will return “PASS”; if any error occurred, *g168Process* will return “FAIL”.

Special Considerations:

- In-place computation is allowed, *i.e.*, input and output buffers could be identical

Suppose 89 words of data are to be processed. The length to be passed for processing is shown below:

```
g168Process (pG168, pSamples_Rin, pSamples_Sin, pSamples_Sout, 89); /* for
process*/
```

Code Example: **Code Example 3-5** shows the use of the *g168Process* Interface.

Code Example 3-5. Use of *g168Process* Interface

```
#include “g168.h”
#include "mem.h"

void test_g168 (void)
{
g168_sHandle *pG168;
g168_sConfigure *pConfig;
Int16 RinBuffer[350], SinBuffer[350], SoutBuffer[350];
Result res;

pConfig = (g168_sConfigure *) memMallocEM(sizeof (g168_sConfigure));

    /* User configuration of G.168 */
    pConfig->Flags= 0;
    pConfig->EchoSpan = 320;

    /* Create and init the instance of G.168 */
    pG168 = g168Create(pConfig);

res = g168Process (pG168, RinBuffer, SinBuffer, SoutBuffer, 350);
}
```

3.3.4 *g168Control*

Call(s):

```
UWord16 g168Control(g168_sHandle *pG168, UWord16 Command);
```

Required Header: *g168.h*

Arguments:

Table 3-4. *g168Control* Arguments

pG168	in	Handle to an instance of G.168
Command	in	The command to be executed by the *g168Control* procedure

Description: The *g168Control* function provides control functions to the G.168 algorithm.

The parameter *pG168* must have been generated from a call to *g168Create*. The parameter *Command* determines which action the *g168Control* algorithm will perform, including:

G168_INHIBIT_CONVERGENCE freezes to the current coefficients of the echo canceller

G168_RESET_COEFFICIENTS resets the echo canceller filter coefficients to zero

G168_REENABLE_CONVERGENCE enables the echo cancellation operation.

Note: The *g168Control* function can be called with only one command in the call; i.e., no two commands can be combined to form a command for *g168Control*.

Returns: Upon successful completion, *g168Control* will return “PASS”; if any error occurred, “FAIL” will be returned.

Special Considerations: Calling the *g168Control* function does not free the memory allocated during the *g168Create* function; if buffers are to be deallocated, the *g168Destroy* function must be called.

Code Example 3-6. Use of the *g168Control* Interface

```
#include “g168.h”
#include "mem.h"

void test_g168 (void)
{
g168_sHandle *pG168;
g168_sConfigure *pConfig;
Int16 RinBuffer[350], SinBuffer[350], SoutBuffer[350];
Result res;
UInt16 Command = G168_INHIBIT_CONVERGENCE;

pConfig = (g168_sConfigure *) memMallocEM(sizeof (g168_sConfigure));

    /* User configuration of G.168 */
    pConfig->Flags= 0;
    pConfig->EchoSpan = 320;
```

```
    /* Create and init the instance of G.168 */
    pG168 = g168Create(pConfig);

res = g168Process (pG168, RinBuffer, SinBuffer, SoutBuffer,13);

res = g168Control (pG168, Command);

res = g168Process (pG168, RinBuffer, SinBuffer, 350);


}
```

3.3.5 *g168Destroy*

Call(s):

```
void g168Destroy (g168_sHandle *pG168);
```

Required Header: *g168.h*

Arguments:

Table 3-5. *g168Destroy* Arguments

pG168	in	Handle to an instance of G.168 generated by a call to *g168Create*

Description: The *g168Destroy* function destroys the instance of the G.168 originally created by a call to *g168Create*.

Returns: None

Special Considerations: Calling the *g168Control* function frees the memory allocated during the *g168Create* function. The *g168Destroy* function deactivates G.168 and frees the memory allocated during the *g168Create* function. The *g168Destroy* function is called only if the *g168Create* function was used to create the instance. If user created the instance himself, bypassing the *g168Create* function, then the user must free the memory.

Code Example 3-7. Use of *g168Destroy* Interface

```
#include “g168.h”
#include "mem.h"

void test_g168 (void)
{
g168_sHandle *pG168;
g168_sConfigure *pConfig;
Int16 RinBuffer[350], SinBuffer[350], SoutBuffer[350];
Result res;

pConfig = (g168_sConfigure *) memMallocEM(sizeof (g168_sConfigure));

    /* User configuration of G.168 */
    pConfig->Flags= 0;
    pConfig->EchoSpan = 320;

    /* Create and init the instance of G.168 */
    pG168 = g168Create(pConfig);

res = g168Process (pG168, RinBuffer, SinBuffer, SoutBuffer, 13);

res = g168Process (pG168, RinBuffer, SinBuffer, SoutBuffer, 350);

g168Destroy (pG168);
}
```

Chapter 4
Building the G.168 Library

4.1 Building the G.168 Library

The G.168 library combines all of the components described in previous sections into one library: *g168.lib*. To build this library, a Metrowerks' CodeWarrior project, ***g168.mcp,*** is provided. This project and all the necessary components to build the G.168 library are located in the ***...\nos\telephony\g168*** directory of the SDK directory structure.

There are two methods to execute a system library project build: dependency build and direct build.

4.1.1 Dependency Build

Dependency build is the easiest approach and requires no additional work on the user's part. If you add the G.168 library project, ***g168.mcp***, to your application project, as shown in **Figure 4-1**, the G.168 library will automatically build when the application is built.

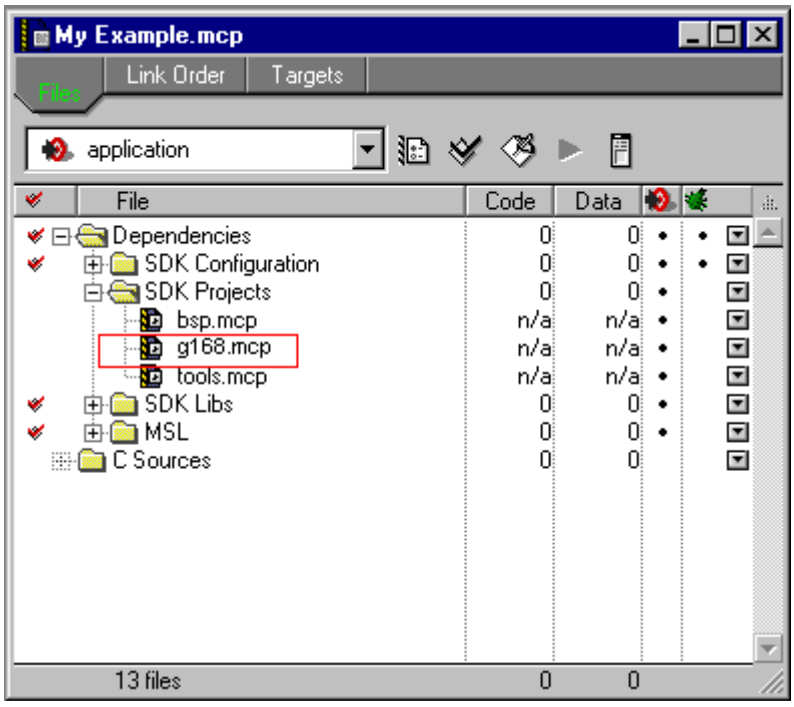

Figure 4-1. Dependency Build for the G.168 Library

4.1.2 Direct Build

Direct build allows you to build a G.168 Line Echo Canceller library independently of any other build. Follow these steps:

Step 1. Open ***g168.mcp*** project, as shown in **Figure 4-2**.

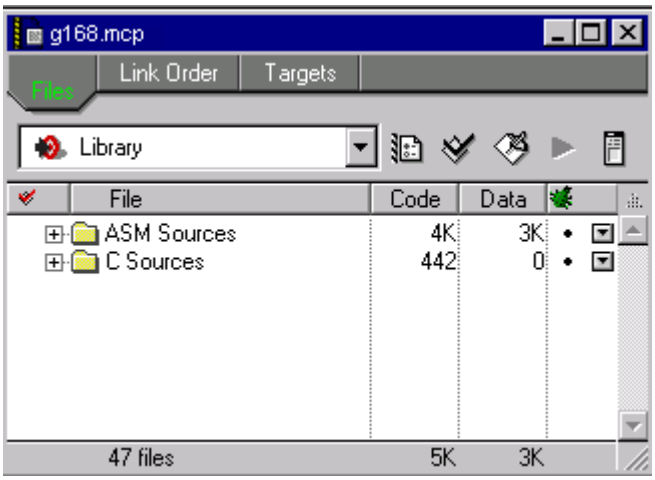

Figure 4-2. *g168.mcp* Project

Step 2. Execute the build by pressing function key [F7] or by choosing the *Make* command from the Project menu; see **Figure 4-3**.

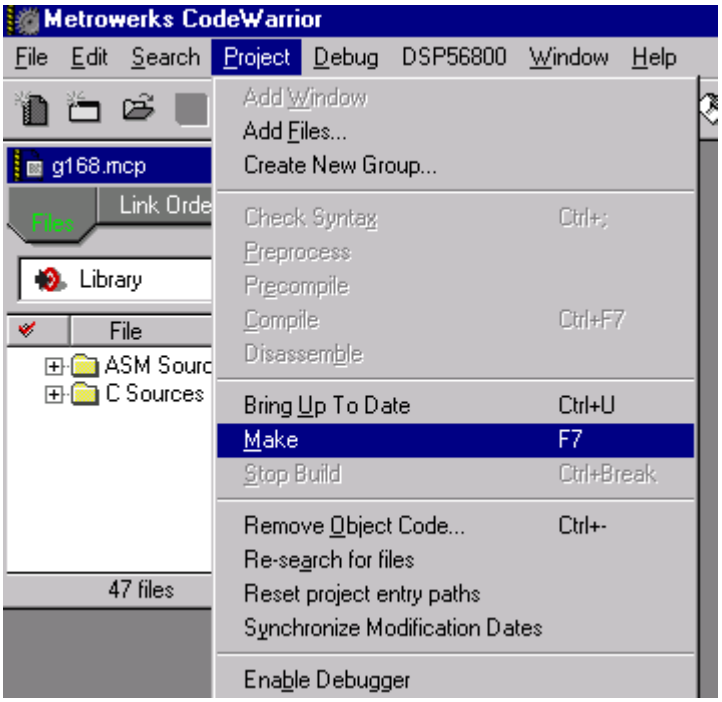

Figure 4-3. Execute Make

At this point, if the build is successful, a ***g168.lib*** library file is created in the **...\nos\telephony\g168\Debug** directory.

Chapter 5
Linking Applications with the G.168 Library

5.1 G.168 Library

The G.168 library should be called after setting the required EchoSpan and Flags and may be configured before calling the *g168Init* function; for details on the G.168 interface, see **Chapter 3**. The library also consists of APIs, which provide an interface between the user application and the G.168 modules. To invoke G.168 (echo cancellation), APIs must be called in this order:

— g168Create (.....);
— g168Init (.....);
— g168Process (.....);
— g168Control (.....); or g168Destroy (.....);

5.1.1 Library Sections

The G.168 Library contains the following data ROM section that must be placed in memory through the linker command file:

- HRL_CONST contains defined constants used in Hold Release Logic
- TD_CONST contains defined constants used in Tone Disabler Logic
- EC_CONST contains defined constants used in Echo Canceller

Code Example 5-1 shows a sample *linker.cmd* file which may be used in testing the G.168 library.

Code Example 5-1. *linker.cmd* File

```
# Linker.cmd file for DSP56824EVM External RAM
# using both internal and external data memory (EX = 0)
# and using external program memory (Mode = 3)

MEMORY {

       .pram   (RWX) : ORIGIN = 0x0000, LENGTH = 0xFF80  # ? external program memory
       .avail  (RW)  : ORIGIN = 0x0000, LENGTH = 0x0030  # available
```

```
      .cwregs (RW)  : ORIGIN = 0x0030, LENGTH = 0x0010  # C temp registrs in
                                CodeWarrior
      .im1    (RW)  : ORIGIN = 0x0040, LENGTH = 0x07C0  # data 1
      .rom    (R)   : ORIGIN = 0x0800, LENGTH = 0x0800  # internal data ROM
      .im2    (RW)  : ORIGIN = 0x1000, LENGTH = 0x0600  # data 2
      .hole   (R)   : ORIGIN = 0x1600, LENGTH = 0x0A00  # hole
      .data   (RW)  : ORIGIN = 0x2000, LENGTH = 0xC000  # data segment
      .em     (RW)  : ORIGIN = 0xE000, LENGTH = 0x1000  # data 3
      .stack  (RW)  : ORIGIN = 0xF000, LENGTH = 0x0F80  # stack
      .onchip1(RW)  : ORIGIN = 0xFF80, LENGTH = 0x0040  # on-chip peripheral
                                registers
      .onchip2(RW)  : ORIGIN = 0xFFC0, LENGTH = 0x0040  # on-chip peripheral
                                registers

}
FORCE_ACTIVE {FconfigInterruptVector}

SECTIONS {

                  #
                  # Data (X) Memory Layout
                  #
                  _EX_BIT     = 0;

                  # Internal Memory Partitions (for mem.h partitions)
                  _NUM_IM_PARTITIONS = 2;  # .im1 and .im2

                  # External Memory Partition (for mem.h partitions)
                  _NUM_EM_PARTITIONS = 1;   # .em

            .main_application_code :
            {
                  # .text sections

                  #  config.c MUST be placed first, otherwise the Interrupt Vector
                  #  configInterruptVector will not be located at the correct
                  address, P:0x0000

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

            } > .pram

            .main_application_data :
            {
                  #
                  # Define variables for C initialization code
                  #
                  F_Xdata_start_addr_in_ROM = ADDR(.rom) + SIZEOF(.rom) / 2;
                  F_StackAddr               = ADDR(.stack);
                  F_StackEndAddr           = ADDR(.stack) + SIZEOF(.stack) / 2  - 1;
```

```
F_Xdata_start_addr_in_RAM = .;

#
# Memory layout data for SDK INCLUDE_MEMORY (mem.h) support
#

FmemEXbit = .;
WRITEH(_EX_BIT);
FmemNumIMpartitions = .;
WRITEH(_NUM_IM_PARTITIONS);
FmemNumEMpartitions = .;
WRITEH(_NUM_EM_PARTITIONS);
FmemIMpartitionList = .;
#WRITEH(ADDR(.im1));
#WRITEH(SIZEOF(.im1) / 2);
WRITEH(ADDR(.im2));
WRITEH(SIZEOF(.im2) / 2);
FmemEMpartitionList = .;
WRITEH(ADDR(.em));
WRITEH(SIZEOF(.em) /2);

# .data sections
* (.data)
* (fp_state.data)
* (rtlib.data)

# G.168 external data starts here
#-------------------------------

* (EC_CONST.data)
* (TD_CONST.data)
* (HRL_CONST.data)

# G.168 external data ends here
#-----------------------------

F_Xdata_ROMtoRAM_length = 0;

F_bss_start_addr = .;
_BSS_ADDR = .;

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

# G.168 external data starts here
#-------------------------------

* (EC_CONST.bss)
* (TD_CONST.bss)
* (HRL_CONST.bss)

# G.168 external data ends here
```

```
        #-----------------------------
        F_bss_length = . - _BSS_ADDR;  # Copy DATA

    } > .data

    FArchIO   = ADDR(.onchip2);
}
```

Chapter 6
G.168 Applications

6.1 G.168 Test and Demo Applications

To verify the G.168 algorithm, test and demo applications have been developed. Refer to the **Targeting Motorola DSP568xx Platform** Manual for the DSP you are using to see if the test and demo applications are available for your target.

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

Chapter 7
License

7.1 Limited Use License Agreement

LIMITED USE LICENSE AGREEMENT

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

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

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

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

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

Licensee agrees to: (a) not use, modify, or copy the Software except as expressly provided herein, (b) not distribute, disclose, transfer, sell, assign, rent, lease, or otherwise make available the Software, any derivatives thereof, or this license to a third party except as expressly provided herein, (c) not remove obliterate, or otherwise defeat any copyright, trademark, patent or proprietary notices, related to the

Software (d) not in any form export, re-export, resell, ship or divert or cause to be exported, re-exported, resold, shipped, or diverted, directly or indirectly, the Software or a direct product thereof to any country which the United States government or any agency thereof at the time of export or re-export requires an export license or other government approval without first obtaining such license or approval.

THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS AND WITHOUT WARRANTY OF ANY KIND INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MOTOROLA BE LIABLE FOR ANY LIABILITY OR DAMAGES OF ANY KIND INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT OR INCIDENTAL OR CONSEQUENTIAL OR PUNITIVE DAMAGES OR LOST PROFITS OR LOSS OF USE ARISING FROM USE OF THE SOFTWARE OR THE PRODUCT REGARDLESS OF THE FORM OF ACTION OR THEORY OF LIABILITY (INCLUDING WITHOUT LIMITATION, ACTION IN CONTRACT, NEGLIGENCE, OR PRODUCT LIABILITY) EVEN IF MOTOROLA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THIS DISCLAIMER OF WARRANTY EXTENDS TO LICENSEE OR USERS OF PRODUCTS AND IS IN LIEU OF ALL WARRANTIES WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE.

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

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

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

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

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

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

Index

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/

MOTOROLA

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

SDK132/D