MOTOROLA

Embedded SDK (Software Development Kit)

Acoustic Echo Canceller Library

SDK125/D
Rev. 2, 07/16/2002

MOTOROLA
intelligence everywhere™

digital dna™

Contents

About This Document

Chapter 1
Introduction

Chapter 2
Directory Structure

Chapter 3
AEC Library Interfaces

Chapter 4
Building the AEC Library

Chapter 5
Linking Applications with the AEC Library

Chapter 6
AEC Applications

Chapter 7
License

List of Tables

List of Figures

List of Examples

About This Document

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

Audience

This document targets software developers implementing the acoustic echo cancellation function 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, AEC Library Interfaces**—describes all of the AEC Library functions
- **Chapter 4, Building the AEC Library**—tells how to execute the system library project build
- **Chapter 5, Linking Applications with the AEC Library**—describes organization of the AEC Library
- **Chapter 6, AEC Applications**—describes the use of AEC 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
- *DSP56824 User's Manual*, DSP56824UM/AD
- *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 DSP56F80x Platform** manual.... ...see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
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	# 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”...

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.

AEC	Acoustic Echo Canceller/Cancellation
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
LSB	Least Significant Bit
MAC	Multiply/Accumulate
MIPS	Million Instructions Per Second
MSB	Most Significant Bit
OnCE™	On-Chip Emulation
OMR	Operating Mode Register
PC	Program Counter
SDK	Software Development Kit
SP	Stack Pointer
SPI	Serial Peripheral Interface
SR	Status Register
SRC	Source
TD	Tone Disabler

References

The following sources were referenced to produce this book:

1. *DSP56800 Family Manual*, DSP56800FM/AD
2. *DSP56824 User’s Manual*, DSP56824UM/AD
3. *Embedded SDK Programmer’s Guide*, SDK101/D

Chapter 1 Introduction

Welcome to Motorola's Family of Digital Signal Processors (DSPs). This document describes the Acoustic 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 Acoustic Echo Canceller Library interface and algorithms.

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

Motorola's Acoustic 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 representative for premium product licensing.

1.1 Quick Start

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

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

1.2 Overview of AEC

Acoustic echo cancellers (AECs) are voice-operated devices which eliminate acoustic echoes and protect the communication from howling due to acoustic feedback from loudspeaker to microphone. AECs are placed in audio terminals on the customer premises.

1.2.1 Background

Many applications, such as full duplex speaker phones and mobile telephones, required AECs with high performance. In a speaker-mic telephone system, a part of the speaker output gets picked up by the microphone, either directly or indirectly, causing annoying echoes heard by the far-end telephone user. Acoustic echo cancellers circumvent these echoes.

The AECs perform echo cancellation by estimating the signal fed back to the microphone from the speaker, then subtracting it from the microphone input as shown in **Figure 1-1**. Room impulse response must also be estimated and is approximated by a linear transversal filter. The coefficients of this filter constitute the room impulse response. The existing echo cancellers use Normalized Least Mean Squares, (NLMS), algorithms for estimating the transversal filter coefficients. The identification, or adaptation, of the transversal filter coefficients is possible only in the absence of near-end speech. Also, when there is a near-end signal present, the estimation of the filter must be frozen; a double talk detection algorithm provides this functionality.

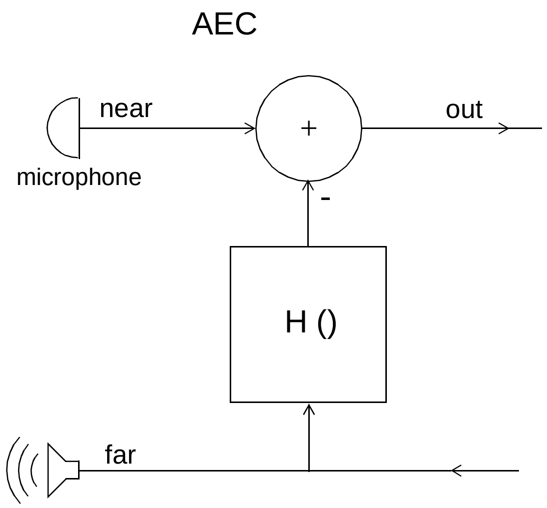


Figure 1-1. Acoustic Echo Cancellation Operation

Some of the parameters for evaluating the performance of the AEC are:

- **ERLE** - Echo Return Loss Enhancement is the ratio of the reference input to the residual echo expressed in dBm. The larger the ERLE, the better the performance of the AEC. Convergence of the adaptive algorithm determines the ERLE.
- **Rate of Convergence** - Acoustic echo is characterized by changing echo paths. Rate of convergence of the adaptive algorithm determines how fast these changes can be tracked. The faster the tracking, the better the performance of the AEC.

The basic requirements for the echo cancellers are:

1. Rapid convergence
2. Low echo return level during single talk
3. Low divergence during double talk

1.2.2 Features and Performance

The AEC 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.

Table 1-1 gives typical values of ERLE and Convergence time.

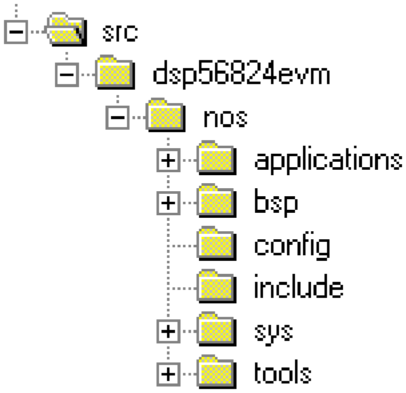
Table 1-1. Typical Values of ERLE and Convergence Time

ERLE	Time to 10dB
25 dB	420 ms

Chapter 2
Directory Structure

2.1 Required Core Directories

Figure 2-1 details required platform directories:



```
src
  dsp56824evm
    nos
      applications
      bsp
      config
      include
      sys
      tools
```

Figure 2-1. Core Directories

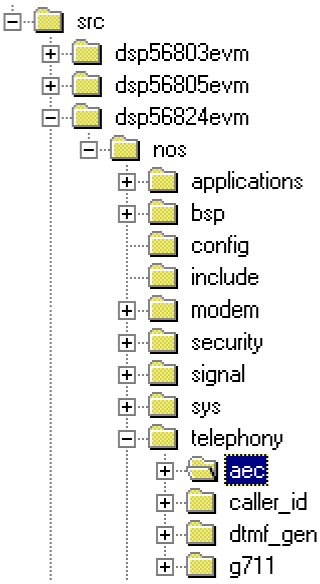
As shown in **Figure 2-1**, DSP56824EVM has no operating system (nos) support and contains these core directories:

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

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

2.2 Optional (Domain-Specific) Directories

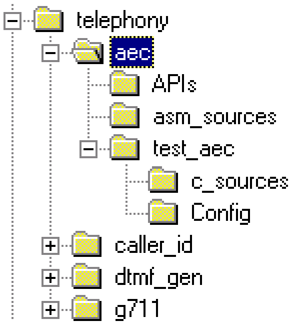
Figure 2-2 demonstrates how the AEC algorithm is encapsulated in the domain specific directories under the directory, *telephony*.



```
src
  dsp56803evm
  dsp56805evm
  dsp56824evm
    nos
      applications
      bsp
      config
      include
      modem
      security
      signal
      sys
      telephony
        aec
        caller_id
        dtmf_gen
        g711
```

Figure 2-2. DSP56824 Directories

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



```
telephony
  aec
    APIs
    asm_sources
    test_aec
      c_sources
      Config
  caller_id
  dtmf_gen
  g711
```

Figure 2-3. *aec* Directory Structure

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

- ***APIs*** contains APIs for AEC
- ***asm_sources*** includes asm sources required for AEC
- ***test_aec*** includes C source files and configuration necessary for testing AEC library modules
  - — ***c_sources*** contains an example test code
  - — ***Config*** contains the configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to AEC

Chapter 3
AEC Library Interfaces

3.1 AEC Services

The AEC 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, as shown below:

s MSB	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i LSB

i = information bit

s = sign bit

3.2 Interface

The C interface for the AEC library services is defined in the C header file *aec.h*, shown in **Code Example 3-1** as a reference.

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

```
#ifndef __AEC_H
#define __AEC_H

/*
   This include file is the master include file for the
   Acoustic Echo Canceller. The applications using AEC
   should include this file
*/

/***************************
 Foundational Include Files
****************************/
```

```
#include "port.h"


/*************************
 *************************/
 Flags
 *************************/

typedef enum
{
      AEC_ES_OFF,               /* 0 implies Echo suppressor OFF */
      AEC_ES_ON                 /* 1 implies Echo suppressor ON */
}aec_eESSwitch;


/********************************************
     Structure for AEC
*********************************************/

/* AEC Energy structure */

/* This structure should not be filled up by the user.
 * This is only for AEC internal use */

typedef struct
{
      long FgEnrg;              /* Foreground Fixed Filter energy */
      long BgEnrg;              /* Background Adaptive Filter energy */
      long FarEnrg;             /* Far end signal energy (Reference) */
      long NearEnrg;            /* Near end signal energy (Input) */
      long xTz;                 /* Vector X Transpose Vector Z */
      long AcR0;                /* Auto Correlation R0 */
      long AcR1;                /* Auto Correlation R1 */
} aec_sEnergy;


/* User configurable structure */

/* This structure has to be used by the user
 * to configure AEC */

typedef struct
{
      UInt16 TailLen;           /* Tail length of AEC in "taps"(integer) */
      aec_eESSwitch esFlag;     /* Echo Suppressor On/Off flag; */
} aec_sConfigure;


/* AEC handle structure */

/* This structure is used internally by AEC for its
 * operation. The user should not setup this structure */

typedef struct
{
      aec_eESSwitch esFlag;     /* ES On/Off flag */
```

```
        UInt16 AecFilLen;           /* AEC filter length */
        UInt16 AecFrameLen;         /* Frame length in samples */
        Frac16 *AecFilterStates;    /* Filter states' buffer (dsm) */
        Frac16 *AecZStates;         /* Decorrelated filter states (dsm)*/
        Frac16 **AecPtrVars;        /* Temp locations for storing pointers */
        Frac16 *AecVariables;       /* All variables used in AEC */
        Frac16 *AecFgCoeff;         /* Foreground Filter Coeffs */
        Frac16 *AecBgCoeff;         /* Background Filter Coeffs */
        aec_sEnergy *AecEnrg;       /* Energy values used in AEC */
} aec_sHandle;


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

EXPORT aec_sHandle *aecCreate (aec_sConfigure *pConfig);

EXPORT Result aecInit (aec_sHandle *pAec, aec_sConfigure *pConfig);

EXPORT Result aecProcess (aec_sHandle *pAec,
                          Word16 *pFarSamples,
                          Word16 *pNearSamples,
                          Word16 *pOutSamples,
                          UWord16 NumSamples);

EXPORT void aecDestroy (aec_sHandle *pAec);


/*****************************************
    #defines used in AEC
*****************************************/

#define AEC_ALPHA_SHIFT   6
#define AEC_THRESH1       0x7000     /* 0.875 */
#define AEC_THRESH2_SHIFT 3
#define AEC_THRESH3       0x7000     /* 0.875 */
#define AEC_FG_COUNT      1          /* Foreground hang over count
                                        of one instant */
#define AEC_BG_COUNT      3          /* Background hang over count
                                        of three instants */

#define AEC_DELTA_LOW     0xdc5d     /* -0.2784001 = 0xdc5d */
#define AEC_DELTA_HIGH    0x0003     /* 0.00009989738 = 0x0003 */
#define AEC_DELTA_RHO     0x0003     /* 0.0001 */
#define AEC_DELTA_STEP    0x0021     /* 0.001 */

#define FG_2_BG_COPY      0x5555     /* Flag value */
#define BG_2_FG_COPY      0xaaaa     /* Flag value */
#define NO_COPY           0xffff     /* Flag value */

#define VARIABLE_SIZE     26         /* Size of AecVariables buffer */
#define AEC_PTR_VAR_SIZE  4          /* Size of AecPtrVars buffer */
#define PTR_TEMP_STATES   3

#endif
```

3.3 Specifications

The following pages describe the AEC 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 *aecCreate*

Call(s):

```
aec_sHandle *aecCreate (aec_sConfigure *pConfig);
```

Required Header: *aec.h*

Arguments:

Table 3-1. *aecCreate* Arguments

pConfig	*in*	Points to the configuration data for AEC

Description: The *aecCreate* function creates an instance of AEC. During the *aecCreate* call, any dynamic resources required by the AEC algorithm are allocated. The memory allocation is:

External Memory: (55 + 2 * AecFilLen) words

Internal Memory: (2 * AecFilLen) words

AecFilLen = (Sampling freq (Hz) * Tail length (ms))/1000

The *pConfig* argument points to the `aec_sConfigure` structure used to configure AEC operation; for details on this structure, see **Section 3.3.2**.

Code Example: The *aecCreate* function allocates memory dynamically using the *mem* library routines as shown in **Code Example 3-2**.

Code Example 3-2. *mem* Library

```
#include "aec.h"
#include "mem.h"

aec_sHandle *aecCreate (aec_sConfigure *pConfig)
{

    aec_sHandle *pAec;
    UInt16 AecFilLen;
    bool   memflag = true;
    Result res;

    /* Calculate AEC Filter length */
    AecFilLen = pConfig->TailLen;

    /* Memory allocation for Handle */
    pAec = (aec_sHandle *) memMallocEM (sizeof (aec_sHandle));
    if (pAec == NULL) return (NULL);

    /* Filter states buffer */
    pAec->AecFilterStates =
           (Frac16 *) memMallocAlignedEM (AecFilLen * sizeof (Frac16));
     memflag = memflag & memIsAligned (pAec->AecFilterStates, pConfig->TailLen) ;

    /* De-correlated filter states */
    pAec->AecZStates =
           (Frac16 *) memMallocAlignedEM (AecFilLen * sizeof (Frac16));
    memflag = memflag & memIsAligned (pAec->AecZStates, pConfig->TailLen);

    /* Pointer to an array of pointers */
    pAec->AecPtrVars =
         (Frac16 **) memMallocEM (AEC_PTR_VAR_SIZE * sizeof (Frac16));

    /* Variables used in AEC */
    pAec->AecVariables =
           (Frac16 *) memMallocEM (VARIABLE_SIZE * sizeof (Frac16));

    /* Forward filter coefficients */
    pAec->AecFgCoeff =
           (Frac16 *) memMallocIM (AecFilLen * sizeof (Frac16));
    memflag = memflag & memIsIM (pAec->AecFgCoeff);

    /* Backward filter coefficients */
    pAec->AecBgCoeff =
           (Frac16 *) memMallocIM (AecFilLen * sizeof (Frac16));
    memflag = memflag & memIsIM (pAec->AecBgCoeff);

    /* Energy structure */
    pAec->AecEnrg =
           (aec_sEnergy *) memMallocEM (sizeof (aec_sEnergy));

    if ( (memflag == false) || (pAec->AecFilterStates == NULL) ||
         (pAec->AecZStates == NULL) || (pAec->AecPtrVars == NULL) )
    {
        aecDestroy (pAec);
        return (NULL);
    }
    else if ( (pAec->AecVariables == NULL) || (pAec->AecFgCoeff == NULL) ||
              (pAec->AecBgCoeff == NULL) ||  (pAec->AecEnrg == NULL) )
    {
        aecDestroy (pAec);
        return (NULL);
    }

    res = aecInit (pAec, pConfig);

    return (pAec);
}
```

For details on the *aec_sHandle* structure and constants used above, please refer to **Code Example 3-1**.

If the *aecCreate* function is called to create an instance, then *aecDestroy* (see **Section 3.3.4**) should be used to destroy the instance.

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

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

Special Considerations:

- Uses both internal and external memories; internal memory is used for filter coefficients.
- AEC application is multichannel and re-entrant; i.e., more than one instance can exist at a time.

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

Code Example 3-3. Use of *aecCreate* Interface

```
#include “aec.h”
#include “mem.h”

/* Function prototype */
void testAEC ();

void testAEC ()
{
       aec_sHandle *pAec;
       aec_sConfigure pConfig;
       Result res;

       /* Initialize Configuration structure */
       pConfig.TailLen = 512;       /* No. of taps corresponding to 64 ms
                                       tail length */
       pConfig.esFlag = AEC_ES_ON;   /* Echo suppressor is on */

       pAec = aecCreate (&pConfig); /* Create and init instance of AEC */
       ....
}
```

For details on structures used in the above example, see **Code Example 3-1**.

3.3.2 *aecInit*

Call(s):

```
Result aecInit (aec_sHandle *pAec, aec_sConfigure *pConfig);
```

Required Header: *aec.h*

Arguments:

Table 3-2. *aecInit* Arguments

pAec	*in*	Handle to an instance of AEC
pConfig	*in*	A pointer to a data structure containing data for initializing the AEC algorithm

Description: The *aecInit* function will initialize the AEC algorithm. During the initialization, all resources will be set to their initial values in preparation for AEC operation. Before calling the *aecInit* function, an AEC instance must be created. The AEC instance (*pAec*) can be created either by calling the *aecCreate* function (see **Section 3.3.1**), or by statically allocating memory, which does not require a call to the *aecCreate* function.

The parameter *pConfig* points to a data structure of type *aec_sConfigure*; its fields initialize AEC operation in the following manner:

TailLen The tail length of AEC in number of tap lengths (e.g., 512 taps). Note that there is no check on tail length in the library; for the DSP56824 (35 MIPS) processor, the tail length should not exceed 512 taps, (i.e., 64 ms). TailLen (in taps) is calculated as follows:

$$\text{TailLen (in taps)} = (F_s * T)/1000$$

$$F_s = \text{Sampling frequency in Hz}$$

$$T = \text{Tail length in milliseconds}$$

esFlag ES (Echo Suppressor) On/Off flag. Echo suppressor cuts off the near-end speech during double talk.

Returns: Upon successful completion, a value of “PASS” will be returned. Otherwise, a value of “FAIL” will be returned.

Special Considerations: None

Code Example: In **Code Example 3-4**, the application creates an instance of AEC. The instance is passed to the *aecInit* function with the AEC configuration structure *pConfig*.

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

```
#include “aec.h”
#include “mem.h”

/* Function prototype */
void testAEC ();

void testAEC ()
{
       aec_sHandle *pAec;
       aec_sConfigure pConfig;
       Result res;

       /* Initialize Configuration structure */
       pConfig.TailLen = 512;        /* No. of taps corresponding to 64 ms
                                        tail length */
       pConfig.esFlag = AEC_ES_ON;   /* Echo suppressor is on */

       pAec = aecCreate (&pConfig); /* Create and init instance of AEC */
       ....
}
```

For details on structures used in the above example, see **Code Example 3-1**.

3.3.3 *aecProcess*

Call(s):

```
Result aecProcess (aec_sHandle *pAec,
                   Word16 *pFarSamples,
                   Word16 *pNearSamples,
                   Word16 *pOutSamples,
                   UWord16 NumSamples);
```

Required Header: *aec.h*

Arguments:

Table 3-3. *aecProcess* Arguments

pAec	*in*	Handle to an instance of AEC
pFarSamples	*in*	Pointer to the reference signal (far end samples) to be used by the AEC algorithm
pNearSamples	*in*	Pointer to near end samples (from which the echo must be cancelled) used by the AEC algorithm
pOutSamples	*out*	Pointer to buffer where the echo cancelled output must be stored
NumSamples	*in*	The number of samples to be processed

Description: The *aecProcess* function will cancel the echo from the near-end samples with *pFarSamples* as reference. The user can call the *aecProcess* function any number of times, as long as there are samples to be processed.

Returns: This function always returns “PASS”.

Special Considerations:

- If internal memory is not allocated for the filter coefficients during a call to *aecCreate* function, the *aecInit* function will fail.
- In-place computation is allowed. To get the output in the near-end samples' buffer itself, make the pointers to near- and out-buffers the same.
- Callback is not implemented, because the AEC code works on a sample-by-sample basis; there is no blocking of samples in AEC code.
- Length of the *pFarSamples* and *pNearSamples* buffer should be identical. The parameter *NumSamples* indicates the length of both input and output buffers.

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

```
#include “aec.h”
#include “mem.h”

Note: This test file describes the AEC test procedure for 160 samples only.

#define FRAME_LEN 160
```

```
/* Function prototype */
void testAEC ();

/* Input and output buffers */
Frac16 FarSamples[FRAME_LEN] = {0x1234, 0xabcd, ....}; /* Put FRAME_LEN far-end
                                                          samples here */
Frac16 NearSamples[FRAME_LEN] = {0x8978, 0xff11, ....}; /* Put FRAME_LEN near-end
                                                           samples here */
Frac16 OutSamples[FRAME_LEN]; /* Buffer for storing echo cancelled samples */

void testAEC ()
{
     aec_sHandle *pAec;
     aec_sConfigure pConfig;
     Result res;

     /* Initialize Configuration structure */
     pConfig.TailLen = 512;        /* No. of taps corresponding to 64 ms
                                      tail length */
     pConfig.esFlag = AEC_ES_ON;   /* Echo suppressor is on */

     pAec = aecCreate (&pConfig); /* Create and init instance of AEC */
     ....
     res = aecProcess (pAec, FarSamples, NearSamples, OutSamples, FRAME_LEN);
     ....
}
```

For details on structures used in the above example, see **Code Example 3-1**.

3.3.4 *aecDestroy*

Call(s):

```
void aecDestroy (aec_sHandle *pAec);
```

Required Header: *aec.h*

Arguments:

Table 3-4. *aecDestroy* Arguments

pAec	*in*	Handle to an instance of AEC generated by a call to *aecCreate*

Description: The *aecDestroy* function destroys the instance of AEC originally created by a call to *aecCreate*. If the user bypassed the *aecCreate* function to create an instance on his own, the *aecDestroy* function should not be called.

Returns: None

Special Considerations: Calling the *aecDestroy* function deactivates AEC and frees the memory allocated during the *aecCreate* function.

Code Example 3-6. Use of *aecDestroy* Interface

```
#include “aec.h”
#include “mem.h”

Note: This test file describes the AEC test procedure for 160 samples only.

#define FRAME_LEN 160

/* Function prototype */
void testAEC ();

/* Input and output buffers */
Frac16 FarSamples[FRAME_LEN] = {0x1234, 0xabcd, ....}; /* Put FRAME_LEN far-end
                                                          samples here */
Frac16 NearSamples[FRAME_LEN] = {0x8978, 0xff11, ....}; /* Put FRAME_LEN near-end
                                                           samples here */
Frac16 OutSamples[FRAME_LEN]; /* Buffer for storing echo cancelled samples */

void testAEC ()
{
     aec_sHandle *pAec;
     aec_sConfigure pConfig;
     Result res;

     /* Initialize Configuration structure */
     pConfig.TailLen = 512;        /* No. of taps corresponding to 64 ms
                                      tail length */
     pConfig.esFlag = AEC_ES_ON;   /* Echo suppressor is on */

     pAec = aecCreate (&pConfig); /* Create and init instance of AEC */
     ....
     res = aecProcess (pAec, FarSamples, NearSamples, OutSamples, FRAME_LEN);
     ....
     aecDestroy (pAec);
     ....
}
```

For details on structures used in the above example, see **Code Example 3-1**.

Chapter 4
Building the AEC Library

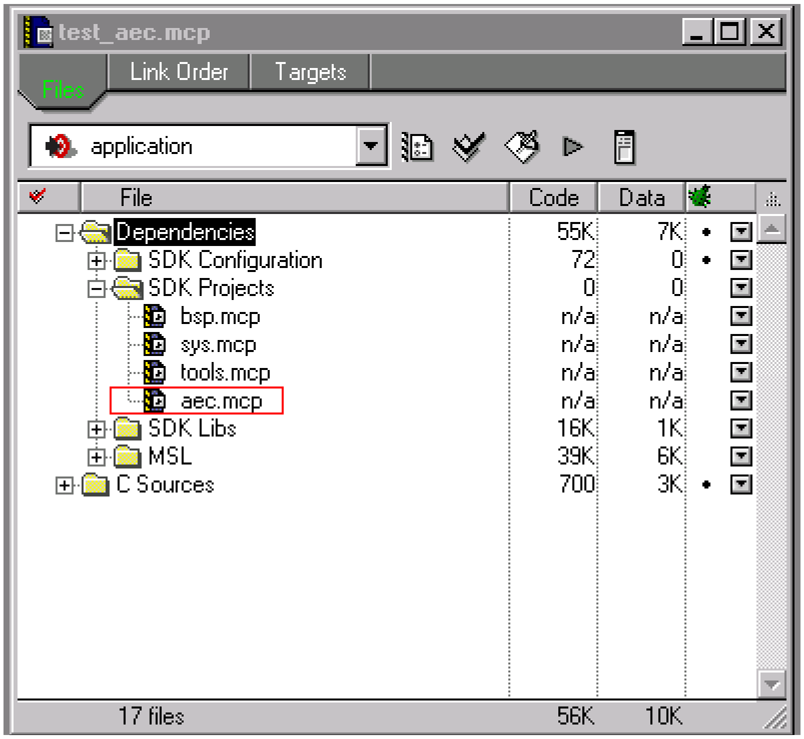
4.1 Building the AEC Library

The AEC library combines all of the components described in previous sections into one library: *aec.lib*. To build this library, a Metrowerks’ CodeWarrior projec, *aec.mcp*, is provided. This project and all the necessary components to build the AEC library are located in the **...\nos\telephony\aec** 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 AEC library project, *aec.mcp*, to your application project, as shown in **Figure 4-1**, the AEC library will automatically build when the application is built.



application

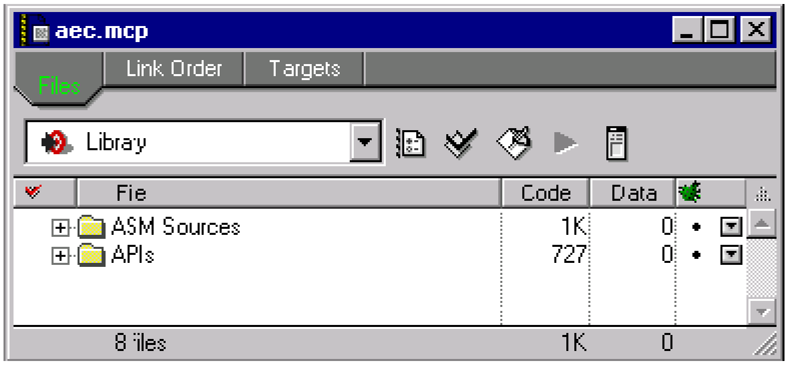
File	Code	Data		
Dependencies	55K	7K	•	
SDK Configuration	72	0	•	
SDK Projects	0	0		
bsp.mcp	n/a	n/a		
sys.mcp	n/a	n/a		
tools.mcp	n/a	n/a		
aec.mcp	n/a	n/a		
SDK Libs	16K	1K		
MSL	39K	6K		
C Sources	700	3K	•	
17 files	56K	10K		

Figure 4-1. Dependency Build for AEC Project

4.1.2 Direct Build

Direct build allows you to build an AEC library independently of any other build. Follow these steps:

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



Libray

Fie	Code	Data		
ASM Sources	1K	0	•	
APIs	727	0	•	
8 iles	1K	0		

Figure 4-2. *aec.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**.

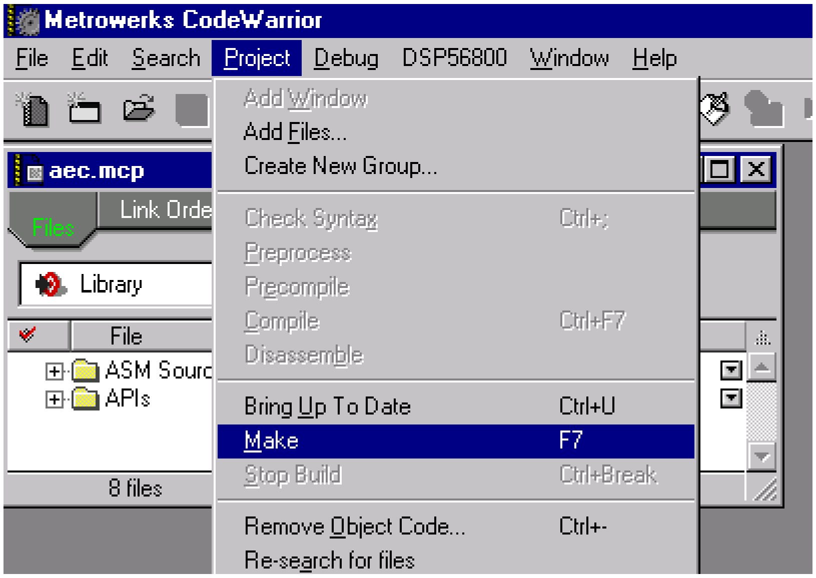


Figure 4-3. Execute Make

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

Chapter 5
Linking Applications with the AEC Library

5.1 AEC Library

The library includes APIs, which define the interface between the user application and the AEC modules. To invoke AEC (Acoustic Echo Canceller), APIs must be called in this order:

- aecCreate (.......);
- aecInit (.......);
- aecProcess (.......);
- aecDestroy (.......);

5.1.1 Library Sections

An example *linker.cmd* file used in the test application follows in **Code Example 5-1**.

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 = 1;  # .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)

                F_Xdata_ROMtoRAM_length = 0;

                F_bss_start_addr = .;
                _BSS_ADDR = .;

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

                F_bss_length = . - _BSS_ADDR;  # Copy DATA
        } > .data

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

Chapter 6
AEC Applications

6.1 Test and Demo Applications

To verify the AEC 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.

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

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

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