MOTOROLA

Embedded SDK (Software Development Kit)

Automatic Gain Control Library

SDK150/D
Rev. 3, 07/16/2002

MOTOROLA intelligence everywhere™ digital dna™

Contents

Chapter 5
Linking Applications with the Automatic Gain Control Library

Chapter 6
Automatic Gain Control Applications

Chapter 7
License

List of Tables

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 Automatic Gain Control (AGC) algorithm for use with Motorola's Embedded Software Development Kit, (SDK).

Audience

This document targets software developers implementing the AGC 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, Automatic Gain Control Library Interfaces**—describes all of the Automatic Gain Control Library functions
- **Chapter 4, Building the Automatic Gain Control Library**—tells how to execute the system library project build
- **Chapter 5, Linking Applications with the Automatic Gain Control Library**—describes organization of the Automatic Gain Control Library
- **Chapter 6, Automatic Gain Control Applications**—describes the use of Automatic Gain Control 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:

- *DSP56852 Family Manual*, DSP56852FM/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 DSP56852 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, ***ns.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.

AGC Automatic Gain Control

API Application Programming Interface

COM1 Communication Port 1

DSP Digital Signal Processor or Digital Signal Processing

EVM	Evaluation Module
I/O	Input/Output
IDE	Integrated Development Environment
IRQA	Interrupt Request A
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

References

The following sources were used to produce this book:

1. *DSP56852 Family Manual*, DSP56852FM/AD
2. *DSP568xx User's Manual*
3. *Embedded SDK Programmer's Guide*

Chapter 1 Introduction

Welcome to Motorola's Family of Digital Signal Processors (DSPs). This document describes the Automatic Gain Control (AGC) 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 Automatic Gain Control 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 Automatic Gain Control 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'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 Motorola DSP568xx Platform** documentation.

For example, the **Targeting Motorola DSP56852 Platform** manual provides more specific information and examples about this hardware architecture. If you are developing an application for an DSP56852EVM board or any other DSP56852 development system, refer to the **Targeting Motorola DSP56852 Platform** manual for **Quick Start** or other DSP56852-specific information.

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

1.2 Overview of Automatic Gain Control

1.2.1 Background

The AGC algorithm maintains the input signal level within a desired range. The mean square level of the input signal is tracked and, based on this, a decision is made to scale up/down the signal or to switch off the AGC. The scaling factor, or gain, is applied to the input signal if the AGC is within the desired range; otherwise, the gain is not applied to the signal. The gain value is updated based on the difference in the input signal level and in the reference level. If the input level is lower/higher than the reference, the gain value is incremented/decremented accordingly, as depicted in **Figure 1-1**.

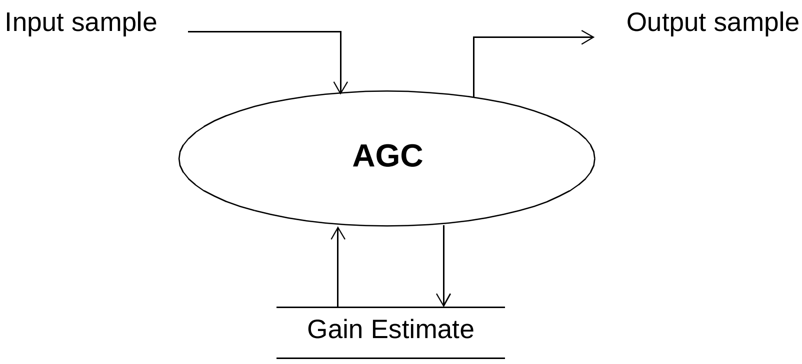

Figure 1-1. Functional Blocks of Automatic Gain Control Library

1.2.2 Features and Performance

The Automatic Gain Control 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

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:

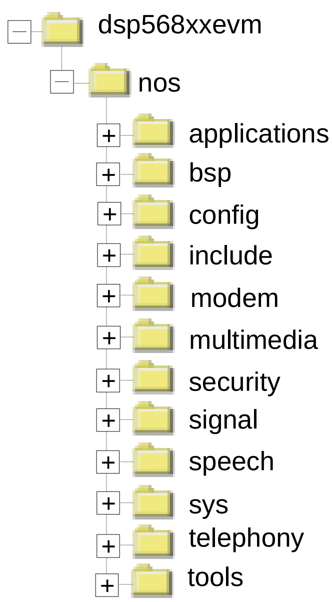

Figure 2-1. Core Directories

In this example, the DSP568xxEVM has a no operating system (nos) support directory. 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

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

2.2 Optional (Domain-Specific) Directories

Figure 2-2 demonstrates how the AGC algorithm is encapsulated in the domain-specific directories under the directory *telephony*. which includes telephony-specific algorithms.

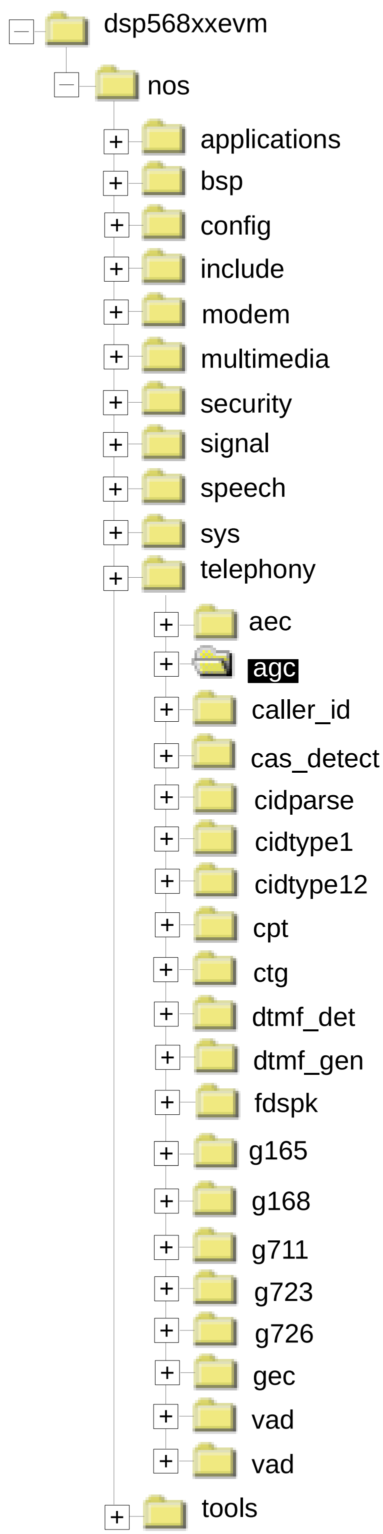

Figure 2-2. Telephony Directory Structure

Figure 2-3 shows the *agc* directory structure under the *telephony* directory.

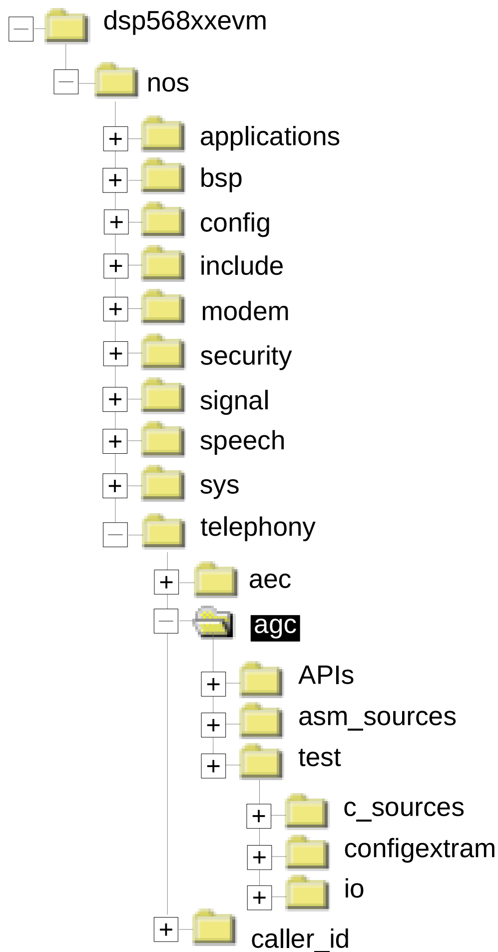

Figure 2-3. *agc* Directory Structure

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

- ***APIs*** includes C-APIs for Automatic Gain Control
- ***asm_sources*** includes all asm sources required for Automatic Gain Control
- ***test*** includes C sources and configuration necessary for testing Automatic Gain Control Library modules
  - ***c_sources*** contains an example test code
  - ***configextram*** contains configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to Automatic Gain Control
  - ***io*** contains input and reference test files

The *applications* directory includes high-level software that exercises the Automatic Gain Control Library. As shown in Figure 2-4, the *applications* directory contains the *agc* application under *telephony*.

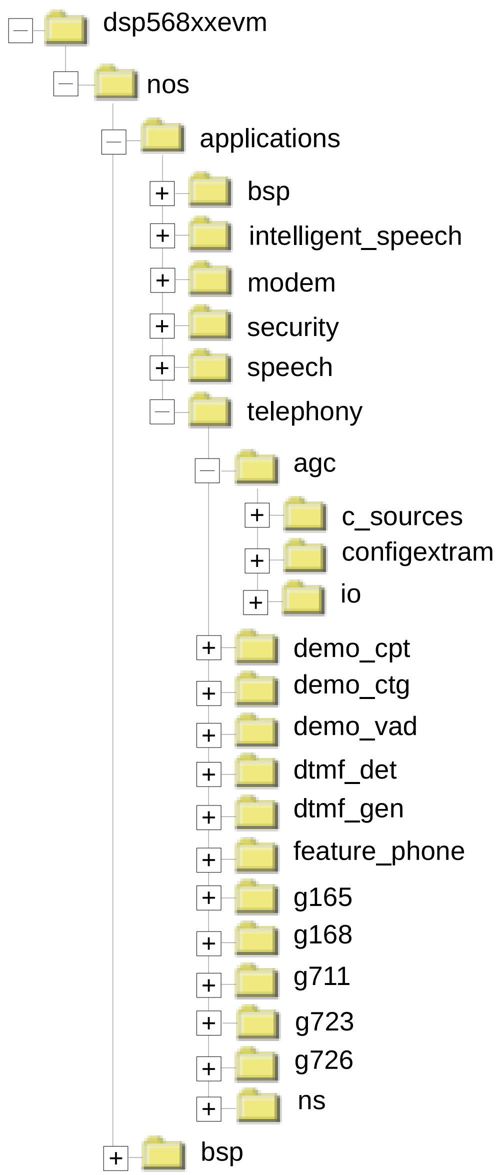

Figure 2-4. AGC Application

Chapter 3
Automatic Gain Control Library Interfaces

3.1 Automatic Gain Control (AGC) Services

The AGC library controls the gain of the input 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 AGC services is defined in the C header file *agc.h*, shown in **Code Example 3-1**.

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

```
#ifndef __AGC_H
#define __AGC_H

/* This include file is the master include file for the
 * Automatic Gain Control. The applications using AGC
 * should include this file */

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

#include "port.h"
```

```
/*******************************************
     Structure for AGC
*********************************************/

/* AGC handle structure */

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

typedef struct
{
  long SignalEnergy;  /* Input Siganl Energy */
  long GainFine;      /* AGC previous Fine Gain */
} agc_sHandle;


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

EXPORT agc_sHandle *agcCreate(void);

EXPORT Result agcInit(agc_sHandle *pAgc);

EXPORT Result agcProcess (agc_sHandle *pAgc,
                          Word16 *pInputSample,
                          Word16 *pOutputSample,
                          UWord16 NumSamples);

EXPORT void agcDestroy (agc_sHandle *pAgc);

#endif
```

3.3 Specifications

The following pages describe the Automatic Gain Control (AGC) 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 *agcCreate*

Call(s):

```
agc_sHandle *agcCreate (void)
```

Required Header: *agc.h*

Arguments: None.

Description: The *agcCreate* function creates an instance of AGC. Multiple instances are possible. During the *agcCreate* call, any dynamic resources required by the AGC algorithm are allocated; each call of *agcCreate* allocates four words of internal 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 "agc.h"
#include "mem.h"

agc_sHandle *agcCreate (void)
{
   agc_sHandle *pAgc;
   Result res;

   /* Memory allocation for Handle */
   pAgc = (agc_sHandle *) memMallocIM (sizeof (agc_sHandle));
   if (pAgc == NULL) return (NULL);

   res = agcInit (pAgc); /* Initialize the AGC variables */

   return (pAgc);
}
```

For details on the *agc_sHandle* structure, please refer to **Code Example 3-1**.

If the *agcCreate* function is called to create an instance, then *agcDestroy* (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 *agcCreate* function. In this case, the user can call the *agcInit* function directly, bypassing the *agcCreate* function. If the user dynamically allocates memory without calling *agcCreate,* then the user himself must destroy the memory allocated.

Returns: Upon successful completion, the *agcCreate* function will return a pointer to the specific instance of AGC created. If *agcCreate* is unsuccessful for any reason, it will return "NULL".

Special Considerations:

- The AGC application is multichannel and re-entrant
- If *agcCreate* is called, then the user need not call the *agcInit* function, which is called internally in the *agcCreate* function

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

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

```
#include "assert.h"
#include "mem.h"
#include "agc.h"
#define FRAME_LEN     80

Word16 input_sample[FRAME_LEN];
Word16 output_sample[FRAME_LEN];

void main (void)
{
    agc_sHandle *pAGC;
    Result res;

    /* AGC handle instance */
    pAGC = agcCreate ();
    if (pAGC == NULL) assert (!"Out of Memory");

    /* Get FRAME_LEN number of samples into input_sample buffer */
    res = agcProcess (pAGC, input_sample, output_sample, FRAME_LEN);

    agcDestroy (pAGC);
    if (res == FAIL) assert (!"Buffers could not be flushed");

    return;
}
```

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

3.3.2 *agcInit*

Call(s):

```
Result agcInit (agc_sHandle *pAGC);
```

Required Header: *agc.h*

Arguments:

Table 3-1. *agcInit* Arguments

pAGC	*in*	Handle to an instance of AGC

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

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

Special Considerations:

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

Code Example: In **Code Example 3-4**, the application creates an instance of AGC. The instance is passed to *agcInit*.

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

```
#include "agc.h"
...

Note: This example shows the use of agcInit function when the user does not use
agcCreate function.

#define FRAME_LEN 80

...

agc_sHandle  pAGC; /* Allocate AGC handle */

...

void main()
{
    Result  res;
    ...
```

```
    /* Call to AGC Init */
    res = agcInit(&pAGC);
    ...
    ...
}
```

3.3.3 *agcProcess*

Call(s):

```
Result agcProcess (agc_sHandle *pAGC, Word16 *pInputSample,
                   Word16 *pOutputSample, UWord16 NumSamples);
```

Required Header: *agc.h*

Arguments:

Table 3-2. *agcProcess* Arguments

pAGC	*in*	Handle to an instance of AGC
pInputSample	*in*	Pointer to the input sample buffer
pOutputSample	*out*	Pointer to the output sample buffer
NumSamples	*in*	Number of samples in the input buffer

Description: The *agcProcess* function monitors input signal power to decide whether or not to switch AGC on. Output signal power is tracked with respect to the reference level (-18 dBm) and the fine gain is adjusted adaptively on a sample-by-sample basis. Input signal power should be in the range of -35 dBm to 0 dBm. The output signal is maintained in the range -26 dBm to -9 dBm. The AGC switches off if the input signal power is less than -35 dBm.The user can call the *agcProcess* function any number of times, as long as there is data.

Returns: Upon successful completion, *agcProcess* returns “PASS”; if there is any error, *agcProcess* returns “FAIL”.

Special Considerations: None

Code Example: See **Code Example 3-3** to learn how to use the *agcProcess* function.

3.3.4 *agcDestroy*

Call(s):

```
void agcDestroy (agc_sHandle *pAGC);
```

Required Header: *agc.h*

Arguments:

Table 3-3. *agcDestroy* Arguments

pAGC	*in*	Handle to an instance of AGC generated by a call to *agcCreate*

Description: The *agcDestroy* function destroys the instance of the AGC originally created by a call to the *agcCreate* function.

Returns: None

Special Considerations: Calling the *agcDestroy* function frees the memory allocated during the *agcCreate* function. The *agcDestroy* function should only be called if the *agcCreate* function was used to create the instance. If the user created the instance himself, bypassing the *agcCreate* function, then the user must free the memory.

Code Example: See **Code Example 3-3** to learn how to use the *agcDestroy* function.

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

Chapter 4
Building the Automatic Gain Control Library

4.1 Building the Automatic Gain Control (AGC) Library

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

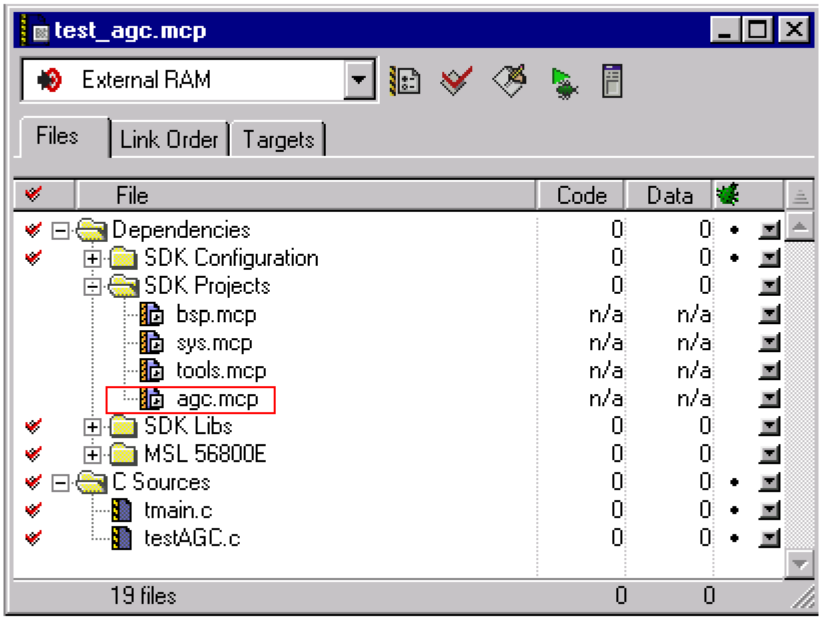

Figure 4-1. Dependency Build for the Automatic Gain Control Library

4.1.2 Direct Build

Direct build allows you to build an AGC Library independently of any other build. Follow these steps:

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

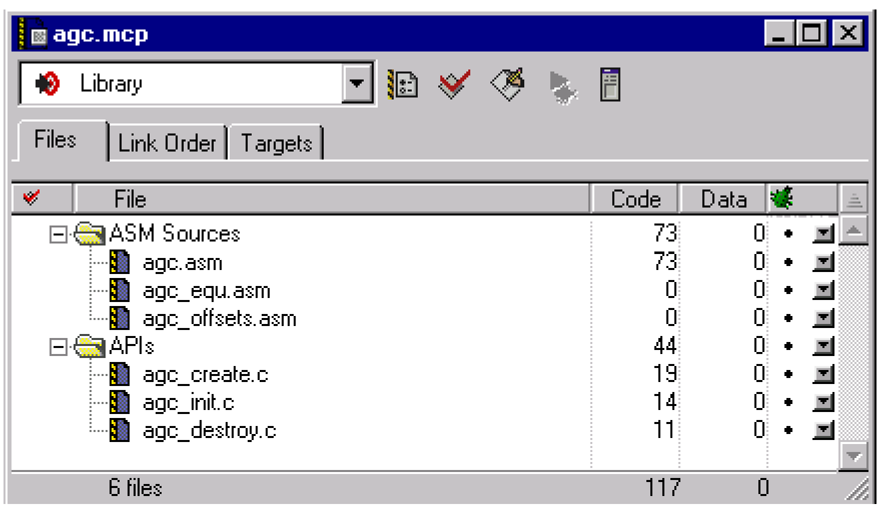

Figure 4-2. *agc.mcp* Project

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

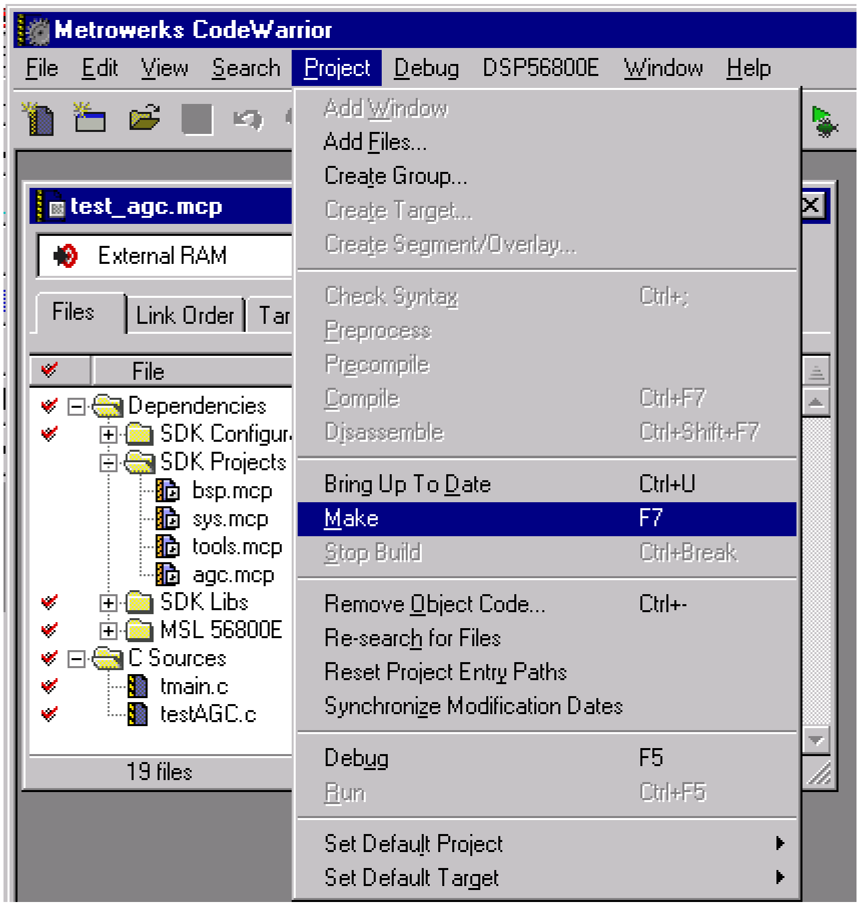

Figure 4-3. Execute *Make*

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

Chapter 5
Linking Applications with the Automatic Gain Control Library

5.1 Automatic Gain Control (AGC) Library

The AGC Library includes APIs, which provide an interface between the user application and the Automatic Gain Control modules. To invoke Automatic Gain Control, APIs must be called in this order:

agcCreate (.......);

agcInit (.......);

agcProcess (.......);

agcDestroy (.......);

For details on the AGC interface, see **Chapter 3**.

5.1.1 Library Sections

Code Example 5-1 shows a sample *linker.cmd* file which may be used in testing the Automatic Gain Control Library.

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

```
#********************************************************************************
#
# Linker.cmd file for DSP56852 External RAM
#    using only external program and data memory.
#
#********************************************************************************

MEMORY {

             .pInterruptVector   (RWX) : ORIGIN = 0x000000, LENGTH = 0x000082
             .pIntRAM            (RWX) : ORIGIN = 0x000082, LENGTH = 0x00177e
             .pExtRAM            (RWX) : ORIGIN = 0x001800, LENGTH = 0x1EE800
```

```
             .pIntROM            (RX)  : ORIGIN = 0x1F0000, LENGTH = 0x000400
             .xIntRAM            (RW)  : ORIGIN = 0x000000, LENGTH = 0x000800
             .xIntRAM_DynamicMem (RW)  : ORIGIN = 0x000800, LENGTH = 0x000800
             .xStack             (RW)  : ORIGIN = 0x001000, LENGTH = 0x000800
             .xExtRAM_DynamicMem (RW)  : ORIGIN = 0x001800, LENGTH = 0x001000
             .xExtRAM            (RW)  : ORIGIN = 0x002800, LENGTH = 0x1FD400
             .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

   #****************************************************************************
```

```
.ApplicationData :
{
      # Define variables for C initialization code

      F_Xdata_start_addr_in_ROM = .;
      F_StackAddr               = ADDR(.xStack);
      F_StackEndAddr            = ADDR(.xStack) +  SIZEOFW(.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 = 1;  # 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)*2);
              WRITEH(SIZEOF(.xIntRAM_DynamicMem)*1);
         FmemEMpartitionList = .;
              WRITEH(ADDR(.xExtRAM_DynamicMem)*2);
              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);

}
```

Chapter 6
Automatic Gain Control Applications

6.1 Test and Demo Applications

To verify the Automatic Gain Control (AGC) 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

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