

Order this document by  
DSP56321EVMUM/D  
Rev. 1.0, 5/2001


# **DSP56321EVM User's Manual**

Motorola, Incorporated  
Semiconductor Products Sector  
6501 William Cannon Drive West  
Austin TX 78735-8598



© Copyright MOTOROLA INC., 2001. All rights reserved.

MOTOROLA reserves the right to make changes without further notice to any products included and covered hereby. 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 incidental, consequential, reliance, exemplary, or any other similar such damages, by way of illustration but not limitation, such as, loss of profits and loss of business opportunity. "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 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  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer. All other tradenames, trademarks, and registered trademarks are the property of their respective owners.

**For More Information On This Product,  
Go to: [www.freescale.com](http://www.freescale.com)**

Quick Start Guide	<b>1</b>
Example Program	<b>2</b>
DSP56321EVM Technical Summary	<b>3</b>
DSP56321EVM Schematics	<b>A</b>
DSP56321EVM Parts List	<b>B</b>
Index	<b>I</b>

<b>1</b>	Quick Start Guide
<b>2</b>	Example Program
<b>3</b>	DSP56321EVM Technical Summary
<b>A</b>	DSP56321EVM Schematics
<b>B</b>	DSP56321EVM Parts List
<b>I</b>	Index

# List of Figures

1-1	DSP56321EVM Connectors and Default Jumper and Switches Settings	1-4
1-2	Connecting the DSP56321EVM Cables . . . . .	1-5
2-1	Development Process Flow. . . . .	2-2
2-2	Example—Initial Maximized Debugger Window Display. . . . .	2-19
2-3	Example Debugger Window Display . . . . .	2-21
3-1	DSP56321EVM Component Layout . . . . .	3-2
3-2	DSP56321EVM Functional Block Diagram. . . . .	3-3
3-3	FSRAM Connections to the DSP56321 . . . . .	3-4
3-4	Example Memory Map with the Unified External Memory. . . . .	3-6
3-5	Address Attribute Register AAR0 . . . . .	3-6
3-6	Flash Connections. . . . .	3-7
3-7	Codec Analog Input/Output Diagram. . . . .	3-10
3-8	Codec Digital Interface Connections . . . . .	3-11
3-9	Parallel Port Interface . . . . .	3-14



# List of Examples

---

2 -1	Simple DSP56321EVM Code Example . . . . .	2-4
------	---	-----





# List of Tables

1-1	DSP56321EVM Default Jumper & Switches Options . . . . .	1-5
2-1	Assembler Options . . . . .	2-6
2-2	Assembler Significant Characters . . . . .	2-8
2-3	Assembly Control . . . . .	2-9
2-4	Symbol Definition . . . . .	2-10
2-5	Data Definition/Storage Allocation . . . . .	2-10
2-6	Listing Control and Options . . . . .	2-11
2-7	Object File Control . . . . .	2-12
2-8	Macros and Conditional Assembly . . . . .	2-12
2-9	Structured Programming . . . . .	2-13
2-10	Linker Options . . . . .	2-14
2-11	Linker Directives . . . . .	2-17
2-12	Steps to Setup Debugger Windows . . . . .	2-20
2-13	Debugger Window Functions . . . . .	2-22
2-14	Steps to Execute and Verify Example Program Results . . . . .	2-23
2-15	Steps to Set and Execute Breakpoints . . . . .	2-24
3-1	J12 FSRAM Memory Configuration Options . . . . .	3-4
3-2	Boot Mode Selection Options . . . . .	3-9
3-3	CS4218 Sampling Frequency Selection . . . . .	3-10
3-4	J22 Jumper Block Options . . . . .	3-13
3-5	J24 Jumper Block Options . . . . .	3-13
3-6	On-Board JTAG Enable/Disable Option . . . . .	3-14
3-7	JTAG/OnCE (J23) Connector Pinout . . . . .	3-16
3-8	Command Converter Connector (P3) Pinout . . . . .	3-16
3-9	SCI Header (J20) Pinout . . . . .	3-17
3-10	J12 Jumper Options . . . . .	3-17
3-11	DSP Serial Port (P1) Connector Pinout . . . . .	3-18

**Freescale Semiconductor, Inc.**

3-12	ESSI0 Header (J22) Pinout. . . . .	3-19
3-13	ESSI1 Header (J24) Pinout. . . . .	3-19
3-14	HI08 Header (J14) Pinout. . . . .	3-20
3-15	Control Bus Signal Header (J10) Pinout . . . . .	3-20
3-16	Address Bus Signal Header (J13) Pinout . . . . .	3-21
3-17	Data Bus Signal Header (J21) Pinout. . . . .	3-22
3-18	Timer Output Test Points(TIO1-TIO3) . . . . .	3-24
3-19	Interrupt Request Test Points(J25). . . . .	3-24
3-20	LED Indicators (LD9, LD10 & LD11). . . . .	3-25
B-1	DSP56321EVM Parts List . . . . .	B-1

# Table of Contents

---



---

## Chapter 1 Quick Start Guide

1.1	Equipment .....	1-1
1.1.1	What You Get with the DSP56321EVM .....	1-1
1.1.2	What You Need to Supply .....	1-2
1.2	Installation Procedure .....	1-2
1.2.1	Preparing the DSP56321EVM .....	1-3
1.2.2	Connecting the DSP56321EVM to the PC and Power. ....	1-5

## Chapter 2 Example Program

2.1	Writing the Program. ....	2-2
2.1.1	Source Statement Format .....	2-2
2.1.1.1	Label Field .....	2-3
2.1.1.2	Operation Field .....	2-3
2.1.1.3	Operand Field .....	2-3
2.1.1.4	Data Transfer Fields .....	2-3
2.1.1.5	Comment Field .....	2-4
2.1.2	Example Program .....	2-4
2.2	Assembling the Program .....	2-5
2.2.1	Assembler Command Format. ....	2-5
2.2.2	Assembler Options .....	2-6
2.2.3	Assembler Directives .....	2-7
2.2.3.1	Assembler Significant Characters. ....	2-8
2.2.3.2	Assembly Control .....	2-9
2.2.3.3	Symbol Definition. ....	2-10
2.2.3.4	Data Definition/Storage Allocation .....	2-10
2.2.3.5	Listing Control and Options .....	2-11
2.2.3.6	Object File Control .....	2-12
2.2.3.7	Macros and Conditional Assembly .....	2-12
2.2.3.8	Structured Programming. ....	2-12
2.2.4	Assembling the Example Program .....	2-13
2.3	Motorola DSP Linker. ....	2-13
2.4	Linker Options .....	2-14
2.4.1	Linker Directives .....	2-17

2.5	Introduction to the Debugger Software . . . . .	2-18
2.6	Running the Program . . . . .	2-20

## Chapter 3 DSP56321EVM Technical Summary

3.1	DSP56321EVM Description and Features . . . . .	3-1
3.2	DSP56321 Description . . . . .	3-1
3.3	Memory . . . . .	3-2
3.3.1	FSRAM . . . . .	3-3
3.3.1.1	FSRAM Connections . . . . .	3-4
3.3.1.2	Example: Programming AAR0 . . . . .	3-5
3.3.2	Flash . . . . .	3-7
3.3.2.1	Flash Connections . . . . .	3-7
3.3.2.2	Programming for Stand-Alone Operation . . . . .	3-7
3.4	Mode Selector . . . . .	3-8
3.5	Audio Codec . . . . .	3-8
3.5.1	Codec Analog Input/Output . . . . .	3-10
3.5.2	Codec Digital Interface . . . . .	3-11
3.6	Command Converter . . . . .	3-13
3.7	Off-Board Interfaces . . . . .	3-15
3.7.1	Serial Communication Interface Port (SCI) . . . . .	3-15
3.7.2	Enhanced Synchronous Serial Port 0 (ESSIO) . . . . .	3-17
3.7.3	Enhanced Synchronous Serial Port 1 (ESSI1) . . . . .	3-17
3.7.4	Host Port (HI08) . . . . .	3-18
3.7.5	Control Bus . . . . .	3-18
3.7.6	Address Bus . . . . .	3-19
3.7.7	Data Bus . . . . .	3-20
3.8	Power Supplies . . . . .	3-20
3.9	Test Points . . . . .	3-21
3.9.1	Ground Test Points . . . . .	3-21
3.9.2	Auxiliary Test Points . . . . .	3-21
3.9.3	Timer Output Test Points . . . . .	3-21
3.9.4	External Interrupts Test Points . . . . .	3-21
3.10	Debug LED's . . . . .	3-22
3.11	Reset . . . . .	3-22
3.12	Clock Source . . . . .	3-22

## Appendix A DSP56321 EVM Schematics

Appendix B  
DSP56321EVM Parts List

B.1	Parts Listing . . . . .	B-1
-----	-------------------------	-----



# Chapter 1

## Quick Start Guide

This section summarizes the evaluation module contents and additional requirements and provides quick installation and test information. The remaining sections of this manual give details on the DSP56321EVM design and operation.

### 1.1 Equipment

The following subsections list the equipment required to use the DSP56321 Evaluation Module (**DSP56321EVM**), some of which is supplied with the module, and some of which must be supplied by the user.

#### 1.1.1 What You Get with the DSP56321EVM

The following material comes with the **DSP56321EVM**:

- DSP56321 Evaluation Module board
- Assorted Hardware (four nylon standoffs and four nylon screws)
- Parallel interface cable (DB25 male to DB25 female)
- The following printed documentation:
  - DSP56321EVM Getting Started Guide
- DSP56321 Evaluation Module CD-ROM that contains the following:
  - Suite56 DSP56300 Software Development Tools Set
  - DSP56321EVM Users's Manual (this document)
  - DSP56300 Family User's Manual
  - DSP56321 Chip User's Manual
  - DSP56321 Technical Data Sheet
  - DSP56321EVM Product Brief
  - DSP56321 Product Brief
  - DSP56321 Chip Errata
  - Sample DSP Software for the DSP56321EVM Kit

— A Documentation Viewer

### 1.1.2 What You Need to Supply

The user must provide the following:

- Windows PC (Pentium class processor or higher) with:
  - Windows 95/98/Me or Windows NT 4.0/2000 operating system
  - Minimum of 16 Mbytes of memory for Windows 95 and 32 Mbytes of memory for Windows 98/Me/NT 4.0/2000
  - CD-ROM drive
  - Hard drive with 20 Mbytes of free disk space
  - Mouse
  - Parallel port
  - RS-232 serial port supporting 9,600–115,200 bit-per-second transfer rates (Optional)
- RS-232 interface cable (DB9 male to DB9 female) (Optional)
- Power supply, (10–12V/1A) AC/DC output into a 2.1mm power connector
- Audio source (tape player, radio, CD player, etc.) (Optional)
- Audio interface cable with 1/8-inch stereo plugs (Optional)
- Headphones (Optional)

## 1.2 Installation Procedure

Installation requires the following four basic steps:

1. Preparing the DSP56321EVM board
2. Connecting the board to the PC and power



### 1.2.1 Preparing the DSP56321EVM

#### **Warning!**

Because all electronic components are sensitive to the effects of electrostatic discharge (ESD) damage, correct procedures should be used when handling all components in this kit and inside the supporting personal computer. Use the following procedures to minimize the likelihood of damage due to ESD:

Always handle all static-sensitive components only in a protected area, preferably a lab with conductive (antistatic) flooring and bench surfaces.

Always use grounded wrist straps when handling sensitive components.

Do not remove components from antistatic packaging until required for installation.

Always transport sensitive components in antistatic packaging.

Locate Figure 1-1 and refer to Table 1-1 for the default jumper and switches settings illustrated in the figure. Table 1-1 contains a brief description of the default jumper setting. More detailed information on the jumper configurations can be found in the Document Section listed in the last column of Table 1-1.

### LEGEND:

■ - DEFAULT INSTALLED JUMPER

▬ - PERMANENTLY INSTALLED JUMPER (PRODUCTION TEST ONLY)

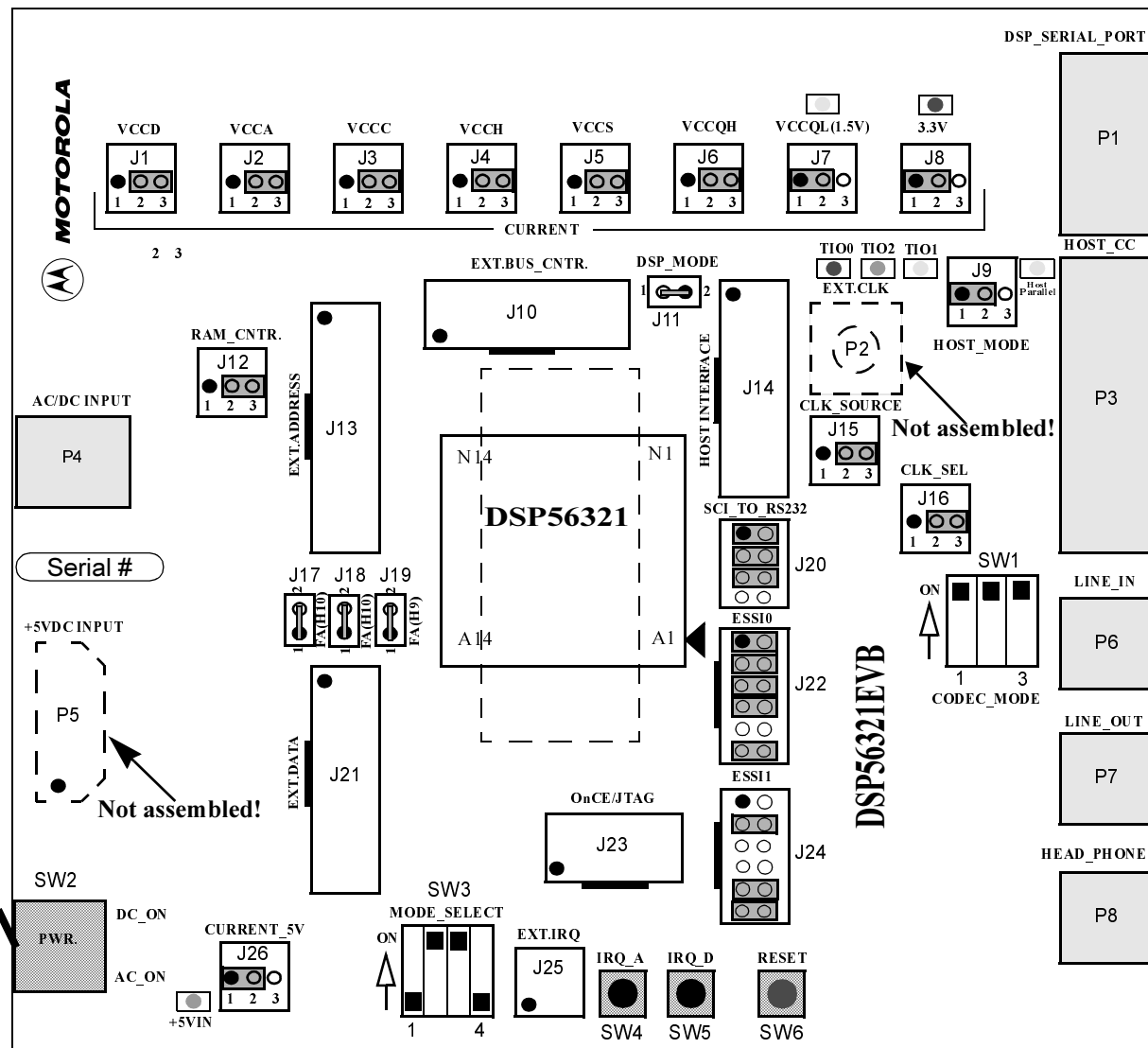


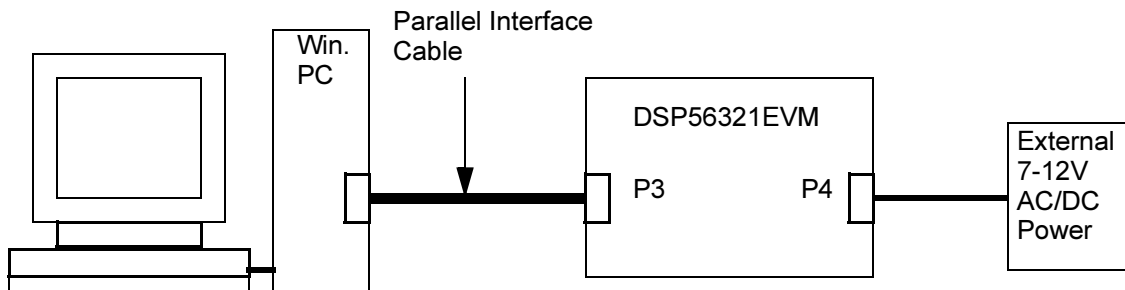
Figure 1-1. DSP56321EVB Connectors and Default Jumper and Switches Settings

**Table 1-1. DSP56321EVM Default Jumper & Switches Options**

<b>Jumper /Switch</b>	<b>Comment</b>	<b>Jumper Connections/ Switch position</b>	<b>Document Section</b>
J1...J6	DSP peripheral voltage supply	2-3	3.8
J7	DSP Core current measuring	1-2	3.8
J8	3.3V current measuring	1-2	3.8
J9	Enables on-board Parallel Command Converter Interface	1-2	3.6
J11,J17, J18,J19	Auxiliary jumpers	Close	-
J12	Selects unified memory map configuration for FSRAM	2-3	3.3.1.1
J15	Select DSP Clock source	2-3	3.12
J16	Select DSP Clock value	2-3	3.12
J20	Connects serial port connector signals RxD and TxD to DSP's SCI port	1-2, 3-4	3.7.1
J20	Connects On-board 153.6kHz oscillator to SCI port's SCLK input (used for baud rate generation}	5-6	3.7.1
J22	Selects DSP's ESSIO port interface for use with on-board Codec	1-2, 3-4, 5-6, 7-8, 11-12	3.7.2
J24	Selects DSP's ESSII port interface for use with on-board Codec	3-4, 9-10, 11-12	3.7.3
J25	External Interrupts	1-2,3-4,5-6 -Open	3.9.4
J26	5V digital/analog supply	1-2	3.8
SW1	Selects 48KHz sample rate for Codec	1,2,3-On	3.5
SW3	Selects DSP Mode 1 operation upon exit from reset	2,3-On. 1,4-Off	3.4, 3.3.2.2

## 1.2.2 Connecting the DSP56321EVM to the PC and Power

Figure 1-2 shows the interconnection diagram for connecting the PC and the external power supply to the DSP56321EVM board.



**Figure 1-2. Connecting the DSP56321EVM Cables**

Use the following steps to complete the cable connections:

1. Connect the male end of the DB25 parallel port interface cable to the parallel port connection on the PC.
2. Connect the female end of the DB25 cable to P3, shown in Figure 1-1, on the DSP56321EVM board. This provides the connection to allow the PC to control the boards functions.
3. Make sure that Power switch SW2 on the DSP56321EVM board in “AC\_On” position
4. Connect the 2.1 mm output power plug into P4, shown in Figure 1-1, on the DSP56321EVM board.
5. Apply power to the power supply.
6. Toggle Power switch SW2 in “DC\_On” position, the green LED (+5Vin.), red LED (3.3V) and yellow LED (VCCQL 1.5V) are lights up when power is correctly applied.

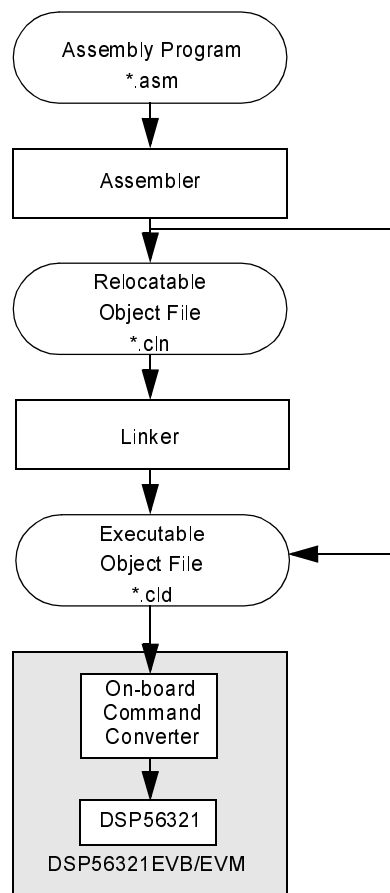
## **Chapter 2**

# **Example Program**

This section contains an example that illustrates how to develop a very simple program for the DSP56321. This example is for users with little or no experience with the DSP development tools. The example demonstrates the form of assembly programs, gives instructions on how to assemble programs, and shows how the Debugger can verify the operation of programs.

Figure 2-1 shows the development process flow for assembly programs. The rounded blocks represent the assembly and object files. The white blocks represent software programs to assemble and link the assemble programs. The gray blocks represent hardware products. The following sections give basic information on the assembly program, the assembler, the linker and the object files. For detailed information on these subjects, consult the assembler and linker manuals provided with the Motorola DSP software package, available through your Motorola sales office or distributor. The documentation is also available through the Motorola Wireless internet URL:

<http://www.mot.com/SPS/DSP/documentation>



**Figure 2-1. Development Process Flow**

## 2.1 Writing the Program

The following sections describe the format of assembly language source statements and give an example assembly program.

### 2.1.1 Source Statement Format

Programs written in assembly language consist of a sequence of source statements. Each source statement may include up to six fields separated by one or more spaces or tabs: a label field, an operation field, an operand field, up to two data transfer fields, and a comment field. For example, the following source statement shows all six possible fields:

trm	mac	x0, y0, a	x: (r0)+, x0	y: (r4)+, y0	;Text
↑	↑	↑	↑	↑	↑
Label	Operation	Operand	X Data Transfer	Y Data Transfer	Comment

### 2.1.1.1 Label Field

The label field is the first field of a source statement and can take one of the following forms:

- A space or tab as the first character on a line ordinarily indicates that the label field is empty and that the line has no label.
- An alphabetic character as the first character indicates that the line contains a symbol called a label.
- An underscore as the first character indicates that the label is local.

With the exception of some directives, a label is assigned the value of the location counter of the first word of the instruction or data being assembled. A line consisting of only a label is a valid line and assigns the value of the location counter to the label.

### 2.1.1.2 Operation Field

The operation field appears after the label field and must be preceded by at least one space or tab. Entries in the operation field may be one of three types:

- Opcode—mnemonics that correspond directly to DSP machine instructions
- Directive—special operation codes known to the assembler that control the assembly process
- Macro call—invocation of a previously-defined macro that is to be inserted in place of the macro call.

### 2.1.1.3 Operand Field

The interpretation of the operand field depends on the contents of the operation field. The operand field, if present, must follow the operation field and must be preceded by at least one space or tab.

### 2.1.1.4 Data Transfer Fields

Most opcodes specify one or more data transfers to occur during the execution of the instruction. These data transfers are indicated by two addressing mode operands separated by a comma, with no embedded blanks. If two data transfers are specified, they must be separated by one or more blanks or tabs. Refer to the *DSP56300 Family Manual* for a complete discussion of addressing modes that are applicable to data transfer specifications.

## 2.1.1.5 Comment Field

Comments are not considered significant to the assembler but can be included in the source file for documentation purposes. A comment field is composed of any characters that are preceded by a semicolon.

## 2.1.2 Example Program

The example program discussed in this section takes two lists of data, one in X memory and one in Y memory, and calculates the sum of the products of the two lists. Calculating the sum of products is the basis for many DSP functions. Therefore, the DSP56321 has a special instruction, Multiply–Accumulate (MAC), that multiplies two values and adds the result to the contents of an accumulator.

### Example 2 -1. Simple DSP56321EVM Code Example

```

;*****
;A SIMPLE PROGRAM: CALCULATING THE SUM OF PRODUCTS
;*****
PBASE    EQU        $100        ;instruct the assembler to replace
                                   ;every occurrence of PBASE with $100
XBASE    EQU        $0          ;used to define the position of the
                                   ;data in X memory
YBASE    EQU        $0          ;used to define the position of the
                                   ;data in Y memory
;*****
;X MEMORY
;*****
        org         x:XBASE      ;instructs the assembler that we
                                   ;are referring to X memory starting
                                   ;at location XBASE
list1    dc         $475638,$738301,$92673a,$898978,$091271,$f25067
        dc         $987153,$3A8761,$987237,$34b852,$734623,$233763
        dc         $f76756,$423423,$324732,$f40029
;*****
;Y MEMORY
;*****
        org         y:YBASE      ;instructs the assembler that we
                                   ;are referring to Y memory starting
                                   ;at location YBASE
list2    dc         $f98734,$800000,$fedcba,$487327,$957572,$369856
        dc         $247978,$8a3407,$734546,$344787,$938482,$304f82
        dc         $123456,$657784,$567123,$675634
;*****
;PROGRAM
;*****
        org         p:0          ;put following program in program
                                   ;memory starting at location 0

```



### Example 2-1. Simple DSP56321EVM Code Example (Continued)

```

begin
    jmp      begin      ;p:0 is the reset vector i.e. where
                        ;the DSP looks for instructions
                        ;after a reset
    org      p:PBASE    ;start the main program at p:PBASE

    move     #list1,r0   ;set up pointer to start of list1
    move     #list2,r4   ;set up pointer to start of list2
    clr      a           ;clear accumulator a
    move     x:(r0)+,x0   y:(r4)+,y0
                        ;load the value of X memory pointed
                        ;to by the contents of r0 into x0 and
                        ;post-increment r0
                        ;load the value of Y memory pointed
                        ;to by the contents of r4 into y0 and
                        ;post-increment r4
    do       #15,endloop;do 15 times
    mac      x0,y0,a      x:(r0)+,x0   y:(r4)+,y0
                        ;multiply and accumulate, and load
                        ;next values
endloop jmp  *           ;this is equivalent to
                        ;label jmp label
                        ;and is therefore a never-ending,
                        ;empty loop
;*****
;END OF THE SIMPLE PROGRAM
;*****

```

## 2.2 Assembling the Program

The following sections describe the format of the assembler command, list the assembler special characters and directives, and give instructions to assemble the example program.

### 2.2.1 Assembler Command Format

The Motorola DSP Assembler is included with the DSP56321EVM on the Motorola Tools CD and can be installed by following the instructions in the *readme.txt* file on the CD. The Motorola DSP assembler is a program that translates assembly language source statements into object programs compatible with the DSP56321. The general format of the command line to invoke the assembler is

***asm56300 [options] <filenames>***

where ***asm56300*** is the name of the Motorola DSP Assembler program, and ***<filenames>*** is a list of the assembly language programs to be assembled.

## 2.2.2 Assembler Options

Table 2-1 describes the assembler options. To avoid ambiguity, the option arguments should immediately follow the option letter with no blanks between them.

**Table 2-1. Assembler Options**

Option	Description
<b>-A</b>	Puts the assembler into absolute mode and generates an absolute object file when the <b>-B</b> command line option is given. By default, the assembler produces a relocatable object file that is subsequently processed by the Motorola DSP linker.
<b>-B&lt;objfil&gt;</b>	<p>Specifies that an object file is to be created for assembler output. &lt;objfil&gt; can be any legal operating system filename, including an optional pathname. The type of object file depends on the assembler operation mode. If the <b>-A</b> option is supplied on the command line, the assembler operates in absolute mode and generates an absolute object (.cld) file. If there is no <b>-A</b> option, the assembler operates in relative mode and creates a relocatable object (.cln) file. If the <b>-B</b> option is not specified, the assembler does not generate an object file. If no &lt;objfil&gt; is specified, the assembler uses the basename (filename without extension) of the first filename encountered in the source input file list and appends the appropriate file type (.cln or .cld) to the basename. The <b>-B</b> option should be specified only once.</p> <p>Example: <i>asm56300 -Bfilter main.asm fft.asm fio.asm</i></p> <p>This example assembles the files main.asm, fft.asm, and fio.asm together to produce the relocatable object file filter.cln.</p>
<b>-D &lt;symbol&gt; &lt;string&gt;</b>	<p>Replaces all occurrences of &lt;symbol&gt; with &lt;string&gt; in the source files to be assembled.</p> <p>Example: <i>asm56300 -DPOINTS 16 prog.asm</i></p> <p>Replaces all occurrences of the symbol POINTS in the program prog.asm by the string '16'.</p>
<b>-EA&lt;errfil&gt; or -EW&lt;errfil&gt;</b>	<p>Allows the standard error output file to be reassigned on hosts that do not support error output redirection from the command line. &lt;errfil&gt; must be present as an argument but can be any legal operating system filename, including an optional pathname. The <b>-EA</b> option causes the standard error stream to be written to &lt;errfil&gt;; if &lt;errfil&gt; exists, the output stream is appended to the end of the file. The <b>-EW</b> option also writes the standard error stream to &lt;errfil&gt;; if &lt;errfil&gt; exists, it is overwritten.</p> <p>Example: <i>asm56300 -EWerrors prog.asm</i></p> <p>Redirects the standard output to the file errors. If the file already exists, it is overwritten.</p>
<b>-F&lt;argfil&gt;</b>	<p>Indicates that the assembler should read command line input from &lt;argfil&gt;, which can be any legal operation system filename, including an optional pathname. &lt;argfil&gt; is a text file containing further options, arguments, and filenames to be passed to the assembler. The arguments in the file need to be separated only by white space. A semicolon on a line following white space makes the rest of the line a comment.</p> <p>Example: <i>asm56300 -Fopts.cmd</i></p> <p>Invokes the assembler and takes the command line options and source filenames from the command file opts.cmd.</p>

**Table 2-1. Assembler Options**

Option	Description
<b>-G</b>	<p>Sends the source file line number information to the object file. This option is valid only in conjunction with the <b>-B</b> command line option. Debuggers can use the generated line number information to provide source-level debugging.</p> <p>Example: <i>asm56300 -B -Gmyprog.asm</i></p> <p>Assembles the file myprog.asm and sends the source file line number information to the resulting object file myprog.cln.</p>
<b>-I&lt;pathname&gt;</b>	<p>Causes the assembler to look in the directory defined by &lt;pathname&gt; for any include file not found in the current directory. &lt;pathname&gt; can be any legal operating system pathname.</p> <p>Example: <i>asm56300 -I\project\ testprog</i></p> <p>Uses IBM PC pathname conventions and causes the assembler to prefix any include files not found in the current directory with the \project\ pathname.</p>
<b>-L&lt;lstfil&gt;</b>	<p>Specifies that a listing file is to be created for assembler output. &lt;lstfil&gt; can be any legal operating system filename, including an optional pathname. If no &lt;lstfil&gt; is specified, the assembler uses the basename (filename without extension) of the first filename encountered in the source input file list and appends .lst to the basename. The <b>-L</b> option is specified only once.</p> <p>Example: <i>asm56300 -L filter.asm gauss.asm</i></p> <p>Assembles the files filter.asm and gauss.asm together to produce a listing file. Because no filename is given, the output file is named using the basename of the first source file, in this case filter, and the listing file is called filter.lst.</p>
<b>-M&lt;pathname&gt;</b>	<p>Causes the assembler to look in the directory defined by &lt;pathname&gt; for any macro file not found in the current directory. &lt;pathname&gt; can be any legal operating system pathname.</p> <p>Example: <i>asm56300 -Mfftlib\ trans.asm</i></p> <p>Uses IBM PC pathname conventions and causes the assembler to look in the fftlib subdirectory of the current directory for a file with the name of the currently invoked macro found in the source file, trans.asm.</p>
<b>-V</b>	<p>Causes the assembler to report assembly progress to the standard error output stream.</p>
<b>-Z</b>	<p>Causes the assembler to strip symbol information from the absolute load file. Normally symbol information is retained in the object file for symbolic references purposes. This option is valid only with the <b>-A</b> and <b>-B</b> options.</p> <p>Note: Multiple options can be used. A typical string might be as follows:</p> <p>Example: <i>asm56300 -A -B -L -G filename.asm</i></p>

### 2.2.3 Assembler Directives

In addition to the DSP56321 instruction set, the assembly programs can contain mnemonic directives that specify auxiliary actions to be performed by the assembler. These are the assembler directives. These directives are not always translated into

machine language. The following sections briefly describe the various types of assembler directives.

### 2.2.3.1 Assembler Significant Characters

Table 2-2 lists one-and two-character sequences are significant to the assembler:

**Table 2-2. Assembler Significant Characters**

Character	Definition
;	Comment delimiter
::	Unreported comment delimiter
\	Line continuation character or macro dummy argument concatenation operator
?	Return value of symbol character
%	Return Hex value of symbol character
^	Macro local label override operator
“	Macro string delimiter or quoted string DEFINE expansion character
@	Function delimiter
*	Location counter substitution
++	String concatenation operator
[]	Substring delimiter
<<	I/O short addressing mode force operator
<	Short addressing mode force operator
>	Long addressing mode force operator
#	Immediate addressing mode operator
#<	Immediate short addressing mode force operator
#>	Immediate long addressing mode force operator

### 2.2.3.2 Assembly Control

The directives used for assembly control are listed in Table 2-3:

**Table 2-3. Assembly Control**

Character	Definition
<b>COMMENT</b>	Start comment lines
<b>DEFINE</b>	Define substitution string
<b>END</b>	End of source program
<b>FAIL</b>	Programmer-generated error message
<b>FORCE</b>	Set operand forcing mode
<b>HIMEM</b>	Set high memory bounds
<b>INCLUDE</b>	Include secondary file
<b>LOMEM</b>	Set low memory bounds
<b>MODE</b>	Change relocation mode
<b>MSG</b>	Programmer-generated message
<b>ORG</b>	Initialize memory space and location counters
<b>RADIX</b>	Change input radix for constants
<b>RDIRECT</b>	Remove directive or mnemonic from table
<b>SCSJMP</b>	Set structured control branching mode
<b>SCSREG</b>	Reassign structured control statement registers
<b>UNDEF</b>	Undefine DEFINE symbol
<b>WARN</b>	Programmer-generated warning

### 2.2.3.3 Symbol Definition

The directives used to control symbol definition are as shown in Table 2-4:

**Table 2-4. Symbol Definition**

Directive	Definition
ENDSEC	End section
EQU	Equate symbol to a value
GLOBAL	Global section symbol declaration
GSET	Set global symbol to a value
LOCAL	Local section symbol declaration
SECTION	Start section
SET	Set symbol to a value
XDEF	External section symbol definition
XREF	External section symbol reference

### 2.2.3.4 Data Definition/Storage Allocation

The directives to control constant data definition and storage allocation are listed in Table 2-5:

**Table 2-5. Data Definition/Storage Allocation**

Directive	Definition
BADDR	Set buffer address
BSB	Block storage bit-reverse
BSC	Block storage of constant
BSM	Block storage modulo
BUFFER	Start buffer
DC	Define constant
DCB	Define constant byte
DS	Define storage
DSM	Define modulo storage

Directive	Definition
DSR	Define reverse carry storage
ENDBUF	End buffer

## 2.2.3.5 Listing Control and Options

The directives to control the output listing are listed in Table 2-6:

**Table 2-6. Listing Control and Options**

Directive	Definition
LIST	List the assembly
LSTCOL	Set listing field widths
NOLIST	Stop assembly listing
OPT	Assembler options
PAGE	Top of page/size page
PRCTL	Send control string to printer
STITLE	Initialize program subtitle
TABS	Set listing tab stops
TITLE	Initialize program title

## 2.2.3.6 Object File Control

The directives for control of the object file are listed in Table 2-7:

**Table 2-7. Object File Control**

Directive	Definition
<b>COBJ</b>	Comment object code
<b>IDENT</b>	Object code identification record
<b>SYMOBJ</b>	Write symbol information to object file

## 2.2.3.7 Macros and Conditional Assembly

The directives for macros and conditional assembly are listed in Table 2-8:

**Table 2-8. Macros and Conditional Assembly**

Directive	Definition
<b>DUP</b>	Duplicate sequence of source lines
<b>DUPA</b>	Duplicate sequence with arguments
<b>DUPC</b>	Duplicate sequence with characters
<b>DUPF</b>	Duplicate sequence in loop
<b>ENDIF</b>	End of conditional assembly
<b>ENDM</b>	End of macro definition
<b>EXITM</b>	Exit macro
<b>IF</b>	Conditional assembly directive
<b>MACLIB</b>	Macro library
<b>MACRO</b>	Macro definition
<b>PMACRO</b>	Purge macro definition

## 2.2.3.8 Structured Programming

The directives for structured programming are listed in Table 2-9:



Table 2-9. Structured Programming

Directive	Definition
.BREAK	Exit from structured loop construct
.CONTINUE	Continue next iteration of structured loop
.ELSE	Perform following statements when .IF false
.ENDF	End of .FOR loop
.ENDI	End of .IF condition
.ENDL	End of hardware loop
.ENDW	End of .WHILE loop
.FOR	Begin .FOR loop
.IF	Begin .IF condition
.LOOP	Begin hardware loop
.REPEAT	Begin .REPEAT loop
.UNTIL	End of .REPEAT loop
.WHILE	Begin .WHILE loop

## 2.2.4 Assembling the Example Program

The assembler is an MS-DOS-based program; thus, to use the assembler you must exit Windows or open an MS-DOS Prompt Window. To assemble the example program, type *asm56300 -a -b -l -g example.asm* into the evm311 directory created by the installation process from Section 2.2.1, "Assembler Command Format," on page 2-5. This creates two additional files: example.cld and example.lst. The example.cld file is the absolute object file of the program; it can be downloaded into the DSP56321. The example.lst file is the listing file; it gives full details of where the program and data are placed in the DSP56321 memory.

## 2.3 Motorola DSP Linker

Though not needed for our simple example, the Motorola DSP linker is also included with the DSP56321EVM. The Motorola DSP linker is a program that processes relocatable object files produced by the Motorola DSP assembler, generating an absolute executable file which can be downloaded to the DSP56321. The Motorola DSP linker is included on the Motorola Tools CD and can be installed by following the instructions in the *readme.txt*

file referenced in Section 2.2.1, "Assembler Command Format," on page 2-5. The general format of the command line to invoke the linker is

***dsplnk [options] <filenames>***

where ***dsplnk*** is the name of the Motorola DSP linker program, and ***<filenames>*** is a list of the relocatable object files to be linked.

## 2.4 Linker Options

Table 2-10 describes the linker options. To avoid ambiguity, the option arguments should immediately follow the option letter with no blanks between them.

**Table 2-10. Linker Options**

Option	Description
<b>-A</b>	Auto-aligns circular buffers. Any modulo or reverse-carry buffers defined in the object file input sections are relocated independently in order to optimize placement in memory. Code and data surrounding the buffer are packed to fill the space formerly occupied by the buffer and any corresponding alignment gaps.  Example: <b><i>dsplnk -A myprog.cln</i></b>  Links the file myprog.cln and optimally aligns any buffers encountered in the input.
<b>-B&lt;objfil&gt;</b>	Specifies that an object file is to be created for linker output. <objfil> can be any legal operating system filename, including an optional pathname. If no filename is specified, or if the <b>-B</b> option is not present, the linker uses the basename (filename without extension) of the first filename encountered in the input file list and appends .cld to the basename. If the <b>-I</b> option is present (see below), an explicit filename must be given because if the linker follows the default action, it can overwrite one of the input files. The <b>-B</b> option is specified only once. If the file named in the <b>-B</b> option already exists, it is overwritten.  Example: <b><i>dsplnk -Bfilter.cld main.cln fft.cln fio.cln</i></b>  Links the files main.cln, fft.cln, and fio.cln together to produce the absolute executable file filter.cld.
<b>-EA&lt;errfil&gt;</b> or <b>-EW&lt;errfil&gt;</b>	Allows the standard error output file to be reassigned on hosts that do not support error output redirection from the command line. <errfil> must be present as an argument, but it can be any legal operating system filename, including an optional pathname. The <b>-EA</b> option causes the standard error stream to be written to <errfil>; if <errfil> exists, the output stream is appended to the end of the file. The <b>-EW</b> option also writes the standard error stream to <errfil>; if <errfil> exists it is overwritten.  Example: <b><i>dsplnk -EWerrors myprog.cln</i></b>  Redirects the standard error output to the file errors. If the file already exists, it is overwritten.

Table 2-10. Linker Options

Option	Description
<b>-F&lt;argfil&gt;</b>	<p>Indicates that the linker should read command line input from &lt;argfil&gt;, which can be any legal operating system filename, including an optional pathname. &lt;argfil&gt; is a text file containing further options, arguments, and filenames to be passed to the linker. The arguments in the file need be separated only by white space. A semicolon on a line following white space makes the rest of the line a comment.</p> <p>Example: <b><i>dsplnk -Fopts.cmd</i></b></p> <p>This example invokes the linker and takes command line options and input filenames from the command file opts.cmd.</p>
<b>-G</b>	<p>Sends source file line number information to the object file. The generated line number information can be used by debuggers to provide source-level debugging.</p> <p>Example: <b><i>dsplnk -B -Gmyprog.cln</i></b></p> <p>Links the file myprog.cln and sends source file line number information to the resulting object file myprog.cld.</p>
<b>-I</b>	<p>The linker ordinarily produces an absolute executable file as output. When the -I option is given, the linker combines the input files into a single relocatable object file suitable for reprocessing by the linker. No absolute addresses are assigned and no errors are issued for unresolved external references. Note that the -B option must be used when performing incremental linking in order to give an explicit name to the output file. If the filename is allowed to default, it can overwrite an input file.</p> <p>Example: <b><i>dsplnk -I -Bfilter.cln main.cln fft.cln fio.cln</i></b></p> <p>Combines the files main.cln, fft.cln, and fio.cln to produce the relocatable object file filter.cln.</p>
<b>-L&lt;library&gt;</b>	<p>The linker ordinarily processes a list of input files that each contain a single relocatable code module. Upon encountering the -L option, the linker treats the following argument as a library file and searches the file for any outstanding unresolved references. If it finds a module in the library that resolves an outstanding external reference, it reads the module from the library and includes it in the object file output. The linker continues to search a library until all external references are resolved or no more references can be satisfied within the current library. The linker searches a library only once, so the position of the -L option on the command line is significant.</p> <p>Example: <b><i>dsplnk -B filter main fir -Lio</i></b></p> <p>Illustrates linking with a library. The files main.cln and fir.cln are combined with any needed modules in the library io.lib to create the file filter.cld.</p>
<b>-M&lt;mapfil&gt;</b>	<p>Indicates that a map file is to be created. &lt;mapfil&gt; can be any legal operating system filename, including an optional pathname. If no filename is specified, the linker uses the basename (filename without extension) of the first filename encountered in the input file list and append .map to the basename. If the -M option is not specified, then the linker does not generate a map file. The -M option is specified only once. If the file named in the -M option already exists, it is overwritten.</p> <p>Example: <b><i>dsplnk -M filter.cln gauss.cln</i></b></p> <p>Links the files filter.cln and gauss.cln to produce a map file. Because no filename is given with the -M option, the output file is named using the basename of the first input file, in this case filter. The map file is called filter.map.</p>

**Table 2-10. Linker Options**

Option	Description
<b>-N</b>	<p>For the linker the case of symbol names is significant. When the -N option is given the linker ignores case in symbol names; all symbols are mapped to lower case.</p> <p>Example: <i>dsplnk -N filter.cln fft.cln fio.cln</i></p> <p>Links the files filter.cln, fft.cln, and fio.cln to produce the absolute executable file filetr.cld; Maps all symbol references to lower case.</p>
<b>-O&lt;mem&gt;[&lt;ctr&gt;] [&lt;map&gt;]:&lt;origin&gt;</b>	<p>By default, the linker generates instructions and data for the output file beginning at absolute location zero for all DSP memory spaces. This option allows the programmer to redefine the start address for any memory space and associated location counter. &lt;mem&gt; is one of the single-character memory space identifiers (X, Y, L, P). The letter can be upper or lower case. The optional &lt;ctr&gt; is a letter indicating the high (H) or low (L) location counters. If no counter is specified the default counter is used. &lt;map&gt; is also optional and signifies the desired physical mapping for all relocatable code in the given memory space. It can be I for internal memory, E for external memory, R for ROM, A for Port A, and B for Port B. If &lt;map&gt; is not supplied, then no explicit mapping is presumed. The &lt;origin&gt; is a hexadecimal number signifying the new relocation address for the given memory space. The -O option can be specified as many times as needed on the command line. This option has no effect if incremental linking is being done. (See the -I option.)</p> <p>Example: <i>dsplnk -Ope:200 myprog -Lmylib</i></p> <p>Initializes the default P memory counter to hex 200 and maps the program space to external memory.</p>
<b>-P&lt;pathname&gt;</b>	<p>When the linker encounters input files, it first searches the current directory (or the directory given in the library specification) for the file. If it is not found and the -P option is specified, the linker prefixes the filename (and optional pathname) of the file specification with &lt;pathname&gt; and searches the newly formed directory pathname for the file. The pathname must be a legal operating system pathname. The -P option can be repeated as many times as desired.</p> <p>Example: <i>dsplnk -P\project\ testprog</i></p> <p>Uses IBM PC pathname conventions and causes the linker to prefix any library files not found in the current directory with the \project\ pathname.</p>
<b>-R&lt;ctlfil&gt;</b>	<p>Indicates that a memory control file is to be read to determine the placement of sections into DSP memory and other linker control functions. &lt;ctlfil&gt; can be any legal operating system filename, including an optional pathname. If a pathname is not specified, an attempt is made to open the file in the current directory. If no filename is specified, the linker uses the basename (filename without extension) of the first filename encountered in the link input file list and append .ctl to the basename. If the -R option is not specified, then the linker does not use a memory control file. The -R option is specified only once.</p> <p>Example: <i>dsplnk -Rproj filter.cln gauss.cln</i></p> <p>Links the files filter.cln and gauss.cln using the memory file proj.ctl.</p>
<b>-U&lt;symbol&gt;</b>	<p>Allows the declaration of an unresolved reference from the command line. &lt;symbol&gt; must be specified. This option is useful for creating an undefined external reference in order to force linking entirely from a library.</p> <p>Example: <i>dsplnk -Ustart -Lproj.lib</i></p> <p>Declares the symbol start undefined so that it is resolved by code within the library proj.lib.</p>

Table 2-10. Linker Options

Option	Description
<b>-V</b>	<p>Causes the linker to report linking progress (beginning of passes, opening and closing of input files) to the standard error output stream. This is useful to insure that link editing is proceeding normally.</p> <p>Example: <b><i>dsplnk -V myprog.cln</i></b></p> <p>Links the file myprog.cln and sends progress lines to the standard error output.</p>
<b>-X&lt;opt&gt;</b> <b>[,&lt;opt&gt;,...,&lt;opt&gt;]</b>	<p>Provides for link time options that alter the standard operation of the linker. The options are described below. All options can be preceded by NO to reverse their meaning. The <b>-X&lt;opt&gt;</b> sequence can be repeated for as many options as desired.</p> <p><b>Option      Meaning</b></p> <p>ABC* Perform address bounds checking  AEC* Check form of address expressions  ASC Enable absolute section bounds checking  CSL Cumulate section length data  ESO Do not allocate memory below ordered sections  OVLP Warn on section overlap  RO Allow region overlap  RSC* Enable relative section bounds checking  SVO Preserve object file on errors  WEX Add warning count to exit status</p> <p>(* means default)</p> <p>Example: <b><i>dsplnk -XWEX filter.cln fft.cln fio.cln</i></b></p> <p>Allows the linker to add the warning count to the exit status so that a project build aborts on warnings as well as errors.</p>
<b>-Z</b>	<p>Allows the linker to strip source file line number and symbol information from the output file. Symbol information normally is retained for debugging purposes. This option has no effect if incremental linking is being done. (See the <b>-I</b> option.)</p> <p>Example: <b><i>dsplnk -Zfilter.cln fft.cln fio.cln</i></b></p> <p>Links the files filter.cln, fft.cln, and fio.cln to produce the absolute object file filter.cln. The output file contains no symbol or line number information.</p>

### 2.4.1 Linker Directives

Similar to the assembler directives, the linker includes mnemonic directives which specify auxiliary actions to be performed by the linker. Table 2-11 is a list of the linker directives.

Table 2-11. Linker Directives

Directive	Definition
<b>BALIGN</b>	Auto-align circular buffers
<b>BASE</b>	Set region base address
<b>IDENT</b>	Object module identification

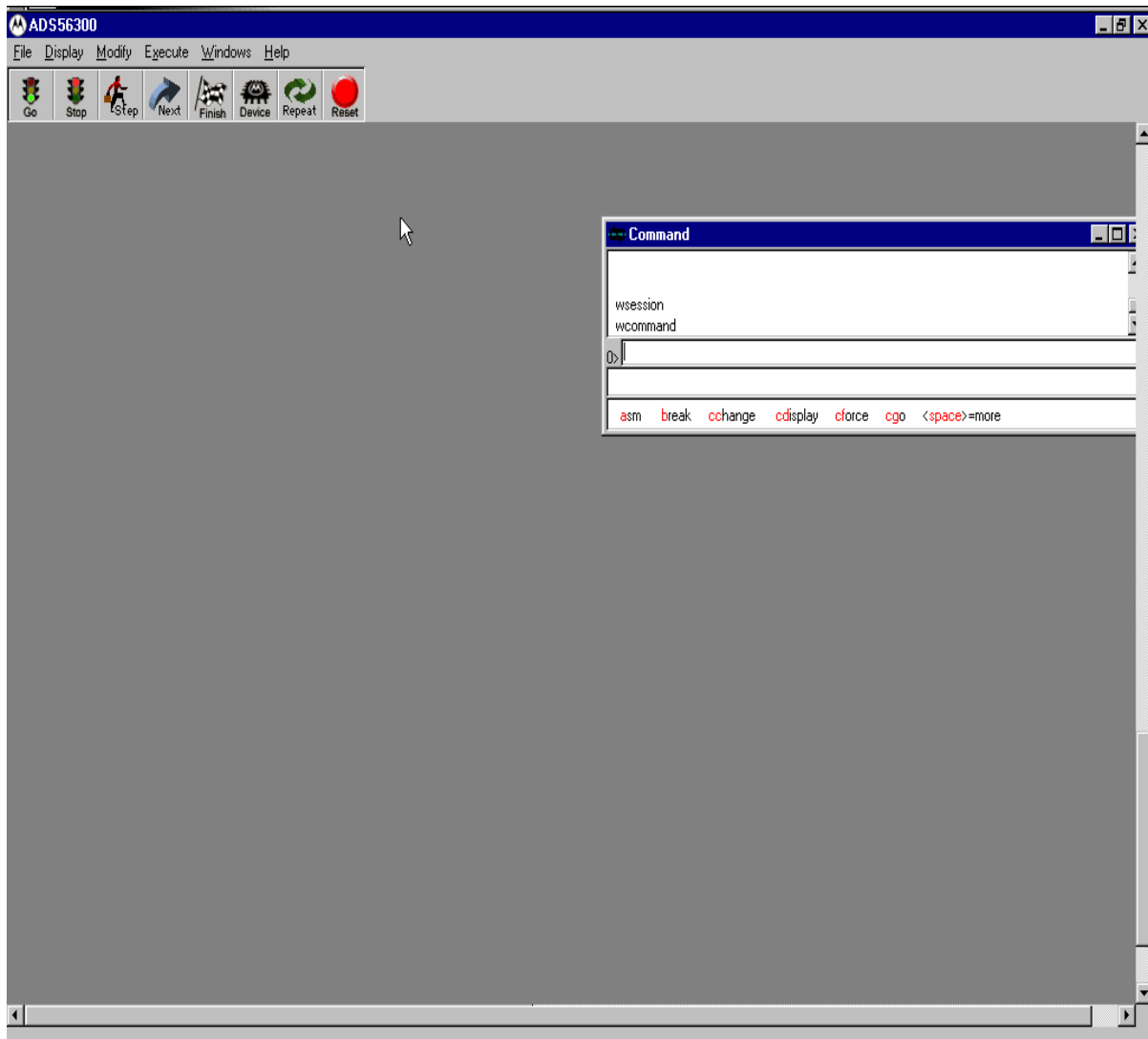
Directive	Definition
INCLUDE	Include directive file
MAP	Map file format control
MEMORY	Set region high memory address
REGION	Establish memory region
RESERVE	Reserve memory block
SBALIGN	Auto-align section buffers
SECSIZE	Pad section length
SECTION	Set section base address
SET	Set symbol value
SIZSYM	Set size symbol
START	Establish start address
SYMBOL	Set symbol value

## 2.5 Introduction to the Debugger Software

This section briefly introduces the Motorola GDS56300 Debugger, giving only the details required to work through this example. To complete this example, you must install the *Motorola GDS56300 Debugger* on your system. The Debugger is located on the *Motorola DSP Suite56 Software Development Tools CD* shipped with your Evaluation Module. Insert the CD, follow the installation instructions and select the ‘DSP 56300 Development Tools’ when prompted to do so. The development tools will be installed on your system and the program icons will be installed in the Windows Start Menu.

For full details on the Debugger, consult the *Suite56 Application Development System User’s Manual*. Section 2 of the manual provides complete installation instructions. Section 4 of the manual discusses the graphical user interface. The *Suite56 Application Development System User’s Manual* can be found on the Tools CD in the following location: <drive>:\Toolsdoc\Ads\Dspadsum.pdf.

To invoke the Debugger, find the ‘Motorola Software Development Tools’ entry in the Windows Start Menu and select the ‘DSP56300 Hardware Debugger’ under this menu. When the Debugger is launched, the program window is not maximized. Once the program window is maximized, the Debugger display should resemble that shown in Figure 2-2.



**Figure 2-2. Example—Initial Maximized Debugger Window Display**

The tool bar is located in the main window just below the menu bar. It comprises a number of buttons providing a convenient way of performing frequently used functions. From left to right the commands are—Go, Stop, Step, Next, Finish, Device, Repeat and Reset.

- The *Go* button starts program execution from the next program counter address.
- The *Stop* button interrupts DSP56321 program execution and returns control to the user.
- The *Step* button executes a single instruction.
- The *Next* button executes one execution step.
- The *Finish* button allows the current function to execute to completion.
- The *Device* button opens the ‘Set Default Device’ dialog box.

- The *Repeat* button repeats the last command in the history buffer, listed in the command window.
- The *Reset* button generates a reset command for the DSP56321

## 2.6 Running the Program

Run the *Motorola Suite56 Application Development System Debugger* by following the steps outlined in section 2.5. Be sure to maximize the program window to full screen immediately after invoking the program.

Table 2-12 is a list of six steps that will open the appropriate windows needed to run the example program. Perform the six steps and then refer to Figure 2-3 for proper sizing and placing of the windows opened.

The last bullet of some steps in Table 2-12 states the command that can be typed in the Command window to perform the same action described using the pull-down menu's.

**Table 2-12. Steps to Setup Debugger Windows**

Step #	To Open:	Action to perform
1	<b>Command window</b>	<ul style="list-style-type: none"> <li>• The Command window is opened by default when the Debugger opens.</li> <li>• Maximize the window and/or click on the scroll bar if it is not visible</li> </ul>
2	<b>Session window</b>	<ul style="list-style-type: none"> <li>• The Session window is opened by default when the Debugger opens.</li> <li>• Maximize the window debugger and/or click on the scroll bars if it is not visible.</li> </ul>
3	<b>Register window</b>	<ul style="list-style-type: none"> <li>• Click on the 'Windows' pull-down menu and select 'Register'. This will open the 'Open Registers Window' option box with 'core' as the selected option. Press OK to confirm this selection.</li> <li>• or type "<b>wregister win1 core</b>" in the Command window</li> </ul>
4	<b>Source window</b>	<ul style="list-style-type: none"> <li>• Click on the 'Windows' pull-down menu and select 'Source'</li> <li>• or type "<b>wsource</b>" in the Command window</li> </ul>
5	<b>X Memory window</b>	<ul style="list-style-type: none"> <li>• Click on the 'Windows' pull-down menu and select 'Memory'. This will open the 'Open Memory Window' option box.</li> <li>• Scroll through the options and select the x memory option that states "x 0\$; 0..fffff,xi, xe or xr depending upon address and omr values".</li> <li>• or type "<b>wmemory win1 x \$0</b>" in the Command window</li> </ul>
6	<b>Y Memory window</b>	<ul style="list-style-type: none"> <li>• Click on the 'Windows' pull-down menu and select 'Memory'. This will open the 'Open Memory Window' option box.</li> <li>• Scroll through the options and select the y memory option that states "y 0\$; 0..fffff,yi, ye or yr depending upon address and omr values".</li> <li>• or type "<b>wmemory win2 y \$0</b>" in the Command window</li> </ul>



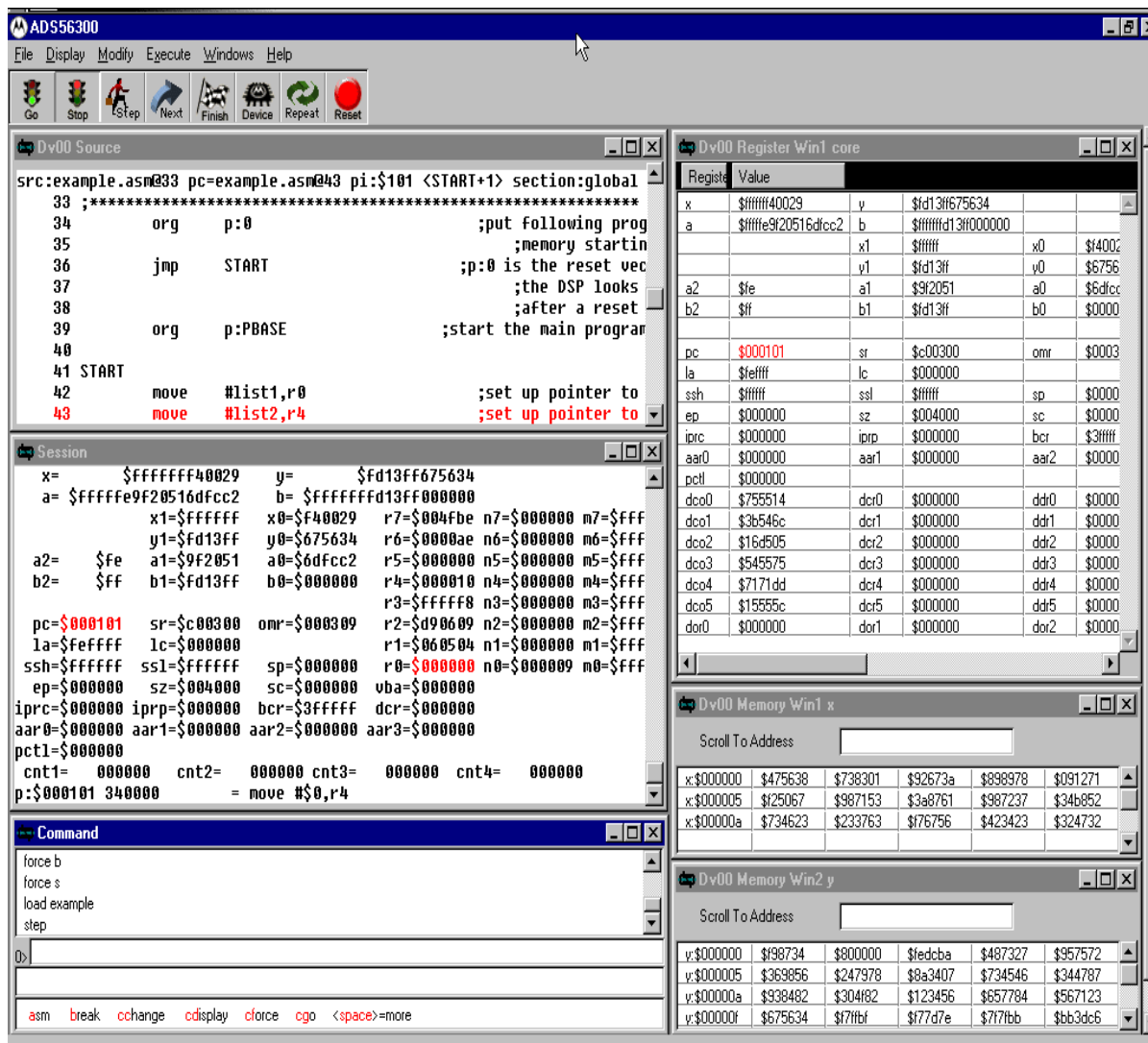


Figure 2-3. Example Debugger Window Display

To save your window setup preferences, the “Save Window Status On Exit” option must be turned on. To do this, go to the “File” pull-down menu and select “Preferences”. Once the dialog box opens, click on the “Save Window Status On Exit” check-box and press OK. The Debugger will save the window arrangement present at the time of exit from the program and reload it when the program is launched at a later time. Only the window preferences will be reloaded when the program is run the next time. Your application program will not be reloaded.

The list below provides a brief description of the windows opened in the steps performed above and shown in Figure 2-3. More details on the functions and syntax of the windows described in Table 2-13 can be found in the *Suite56 Application Development System*

*User's Manual.* The manual is located on the Tools CD in the following location:  
<drive>:\Toolsdoc\Ads\Dspadsum.pdf.

**Table 2-13. Debugger Window Functions**

Window	Functional Description
<b>Command window</b>	<ul style="list-style-type: none"> <li>The Command window provides the main interface between the user interface and the rest of the system.</li> <li>The user may type commands directly into the command window.</li> <li>The last commands executed are displayed and may be retrieved, edited and resubmitted to the Debugger.</li> <li>All commands generated by using the pull-down menu's are displayed in the Command window.</li> </ul>
<b>Source window</b>	<ul style="list-style-type: none"> <li>The Source window displays the source code for the executing program.</li> <li>The window automatically tracks the program counter (PC), displaying the corresponding source line highlighted in red.</li> </ul>
<b>Register window</b>	<ul style="list-style-type: none"> <li>The Register window displays and modifies a group of registers(ie. DSP core, timers, host port...) for the device.</li> <li>Registers may be changed by clicking on the register in question once, typing a new value and storing the value with a carriage return</li> </ul>
<b>Session window</b>	<ul style="list-style-type: none"> <li>The Session window provides the main output from the development system.</li> <li>All commands input via the Command window are echoed in the Session window</li> <li>All output from commands is displayed in the Session window</li> <li>Error messages are sent to the Session window</li> </ul>
<b>X Memory window</b>	<ul style="list-style-type: none"> <li>The X Memory window displays and optionally changes the contents of x memory.</li> <li>Memory values may be changed by clicking on the register in question once, typing a new value and storing the value with a carriage return</li> </ul>
<b>Y Memory window</b>	<ul style="list-style-type: none"> <li>The Y Memory window displays and optionally changes the contents of y memory.</li> <li>Memory values may be changed by clicking on the register in question once, typing a new value and storing the value with a carriage return</li> </ul>

Now that the Debugger has been loaded and configured, we can proceed with executing the program and examining the results. Table 2-14 outlines the steps to complete the next section of the example program tutorial.

**Table 2-14. Steps to Execute and Verify Example Program Results**

Step #	Action	Instructions
1	<b>Load program</b>	<ul style="list-style-type: none"> <li>To load the example program, click on the 'File' pull-down menu and select 'Load'.</li> <li>This will bring up the Load Memory COFF option box. In the option box, type "&lt;drive&gt;:\example\example.cld"</li> <li>Once the program is loaded, the Assembly window displays instructions beginning at address p:\$0, the reset vector.</li> </ul>
2	<b>Display program in source window</b>	<ul style="list-style-type: none"> <li>The example program begins at address p:\$100.</li> <li>To change the Source window to show instructions beginning at p:\$100, select the Source window and type <i>100</i> followed by a carriage return in the box labeled 'Scroll to Address' located at the top of the Assembly window.</li> </ul>
3	<b>Verify x and y memory data</b>	<ul style="list-style-type: none"> <li>The x Memory window should be displaying data beginning at address x:\$000000. Verify that the data being displayed matches the data shown in X Memory section of the listing of Example 2-1 on page 2-4.</li> <li>The y Memory window should be displaying data beginning at address y:\$000000. Verify that the data being displayed matches the data shown in the Y Memory section of the listing of Example 2-1 on page 2-4.</li> </ul>
4	<b>Step through the program</b>	<ul style="list-style-type: none"> <li>To step through the program, click on the Step button located in the toolbar of the Debugger.</li> <li>Typing "step" in the command window will perform the same function</li> <li>The Command window provides a shortcut for typing commands. Type the start of the command and press the space bar, and the debugger completes the remainder of the command. To repeat the last command, press return.</li> </ul>
5	<b>Debugger display</b>	<ul style="list-style-type: none"> <li>As you step through the code, notice that the registers in the Registers window are changed by the instructions.</li> <li>After each cycle, any register that has been changed is highlighted.</li> </ul>
6	<b>Verify results</b>	<ul style="list-style-type: none"> <li>Once you have stepped through the program, ensure that the program has executed correctly by checking that the result in accumulator a is \$FE 9F2051 6DFCC2.</li> </ul>

Stepping through the program like this is good for short programs, but it is impractical for large complex programs. The way to debug large programs is to set breakpoints, which are user-defined points where execution of the code stops, allowing the user to step through the section of interest. Breakpoints can be set using the following methods:

- Double-click on source line in the Source window
- Pull-down menu - Execute pull-down menu, Breakpoint sub-menu, Set or Clear sub-menu
- Double-click on Assembly window address field (An assembly window was not opened for this example)

Follow the steps in Table 2-15 to set a breakpoint to verify that the values in r0 and r4 are correct before the do loop.

**Table 2-15. Steps to Set and Execute Breakpoints**

Step #	Action	Instructions
1	<b>Reset Program Counter (PC)</b>	<ul style="list-style-type: none"> <li>• Press the “Stop” button on the tool bar to stop the program from executing.</li> <li>• In the Command window, type “change pc 0”. This resets the program counter to the reset vector.</li> <li>• To set the breakpoint, click on the “Execute” pull-down menu and select</li> </ul>
2	<b>Set breakpoint</b>	<ul style="list-style-type: none"> <li>• To set the breakpoint, double-click on line #53 in the Source Window. The line will be highlighted in blue to indicate that a breakpoint has been set.</li> <li>• This sets a breakpoint at the beginning of the Do loop.</li> <li>• The breakpoint can also be set by typing “break p:\$106” in the Command window</li> </ul>
3	<b>Execute program</b>	<ul style="list-style-type: none"> <li>• Press the Go button in the tool bar to execute the program or type “go” in the Command window</li> </ul>
4	<b>Debugger display</b>	<ul style="list-style-type: none"> <li>• The program executes up the to the point where the breakpoint was set</li> <li>• You can now step through the remainder of the example program</li> </ul>
	<b>Exit Debugger</b>	<ul style="list-style-type: none"> <li>• To exit the Debugger, select “Exit” from the file menu or type “quit” in the Command window</li> </ul>

# Chapter 3

## DSP56321EVM Technical Summary

### 3.1 DSP56321EVM Description and Features

An overview description of the DSP56321EVM is provided in the DSP56321EVM Product Brief (*DSP56321EVM/P/D*) included with this kit. The main features of the DSP56321EVM include the following:

- DSP56321 24-bit digital signal processor
- FSRAM for expansion memory and Flash memory for stand-alone operation
- 16-bit CD-quality audio codec
- Command converter circuitry

### 3.2 DSP56321 Description

A full description of the DSP56321 part, including functionality and user information, is provided in the following documents included as a part of this kit:

- *DSP56321 Technical Data* (Document order number DSP56321/D): Provides features list and specifications including signal descriptions, DC power requirements, AC timing requirements, and available packaging.
- *DSP56321 User's Manual* (Document order number DSP56321UM/D): Provides an overview description of the DSP and detailed information about the on-chip components including the memory and I/O maps, peripheral functionality, and control and status register descriptions for each subsystem.
- *DSP56300 Family Manual* (Document order number DSP56300FM/AD): Provides a detailed description of the core processor including internal status and control registers and a detailed description of the family instruction set.

Refer to these documents for detailed information about chip functionality and operation.

**Note:** A detailed list of known chip errata is also provided with this kit. Refer to the *DSP56321 Chip Errata* document for information that has changed since the publication of the reference documentation listed above. The latest version can be obtained on the Motorola DSP World Wide Web site at

<http://www.mot.com/SPS/DSP/chiperrata/index.html>

### 3.3 Memory

The DSP56321EVM includes the following external memory:

- 64K  $\times$  24-bit fast static RAM (FSRAM) for expansion memory
- 128K  $\times$  8-bit flash memory for stand-alone operation

Refer to Figure 3-1 for the location of the FSRAM and Flash on the DSP56321EVM. Figure 3-2 shows a functional block diagram of the DSP56321EVM including the memory devices.

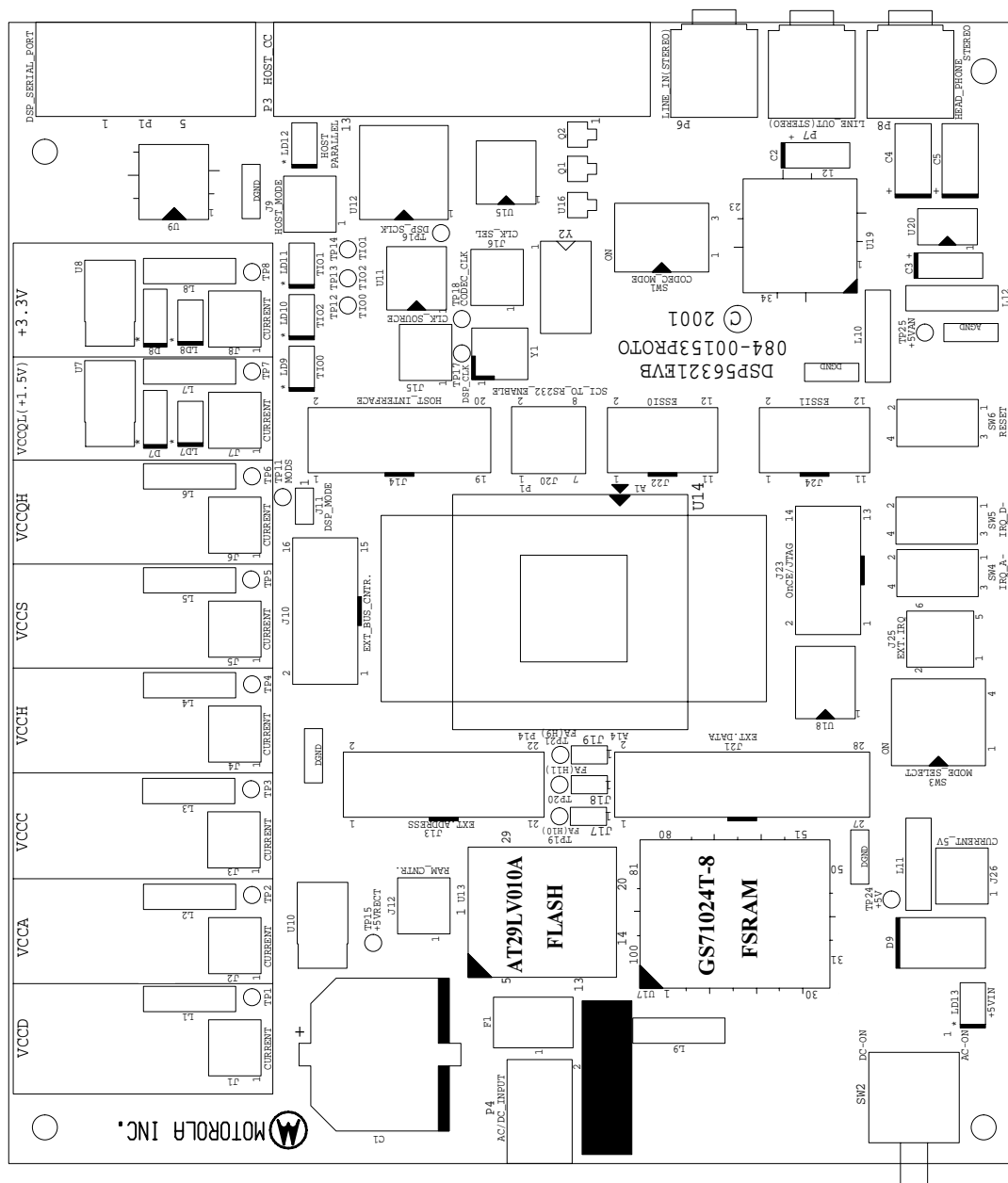


Figure 3-1. DSP56321EVM Component Layout



DSP56321EVM User's Manual

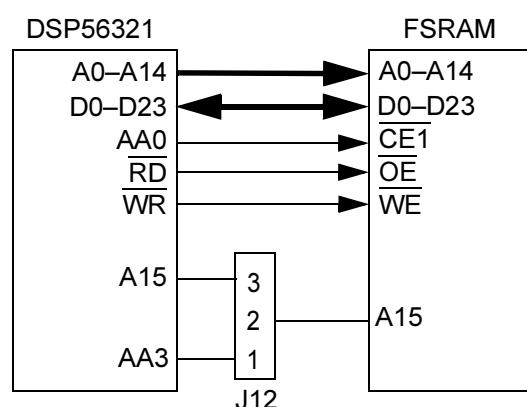
## Motorola

**For More Information On This Product,  
Go to: [www.freescale.com](http://www.freescale.com)**



### 3.3.1.1 FSRAM Connections

The basic connection for the FSRAM is shown in Figure 3-3.



**Figure 3-3. FSRAM Connections to the DSP56321**

The data input/output pins D0—D23 for the FSRAM are connected to the DSP56321 D0—D23 pins. The FSRAM write enable ( $\overline{WE}$ ) and output enable ( $\overline{OE}$ ) lines are connected to the DSP56321 write enable ( $\overline{WR}$ ) and read enable ( $\overline{RD}$ ) lines, respectively. The FSRAM chip enable ( $\overline{CE1}$ ) is generated by the DSP56321 address attribute 0 ( $\overline{AA0}$ ). The FSRAM activity is controlled by AA0 and the corresponding address attribute register 0 ( $\overline{AAR0}$ ). The FSRAM address input pins, A0—A14, are connected to the respective port A address pins of the DSP. Address bit A15 and address attribute 3 ( $\overline{AA3}$ ) are connected to jumper J12 pin 3 and pin 1 respectively. Pin 2 of jumper J12 is then connected to address bit A15 of the FSRAM. This allows for user configuration of the external FSRAM to be either unified or split memory map. Table 3-1 below illustrates the FSRAM configurations selectable via jumper J12. The default board configuration selects a unified memory map of 64K words.

**Table 3-1. J12 FSRAM Memory Configuration Options**

J12	Memory Map Type
2-3	Unified Memory Map (Default)
1-2	Split Memory Map

When a unified memory map is selected, jumper shorting J12 pins 3 and 2, the 64K words of available external FSRAM are not partitioned into X data, Y data, and program memory. Thus, access to P:\$1000, X:\$1000, and Y:\$1000 are treated as accesses to the same memory cell, and 48-bit long memory data moves are not possible to or from the external FSRAM.

By selecting the split memory map, the 64K of available FSRAM memory is partitioned into two contiguous 32K memory blocks. This configuration allows for 48-bit long memory data moves from FSRAM. Thus, access to X:\$1000 and Y:\$1000 could access different memory cells in the partitioned external FSRAM. The split memory map utilizes address attribute pin 3 (AA3). Activity of the DSP56321 pin AA3 is controlled by the corresponding AAR3 register.

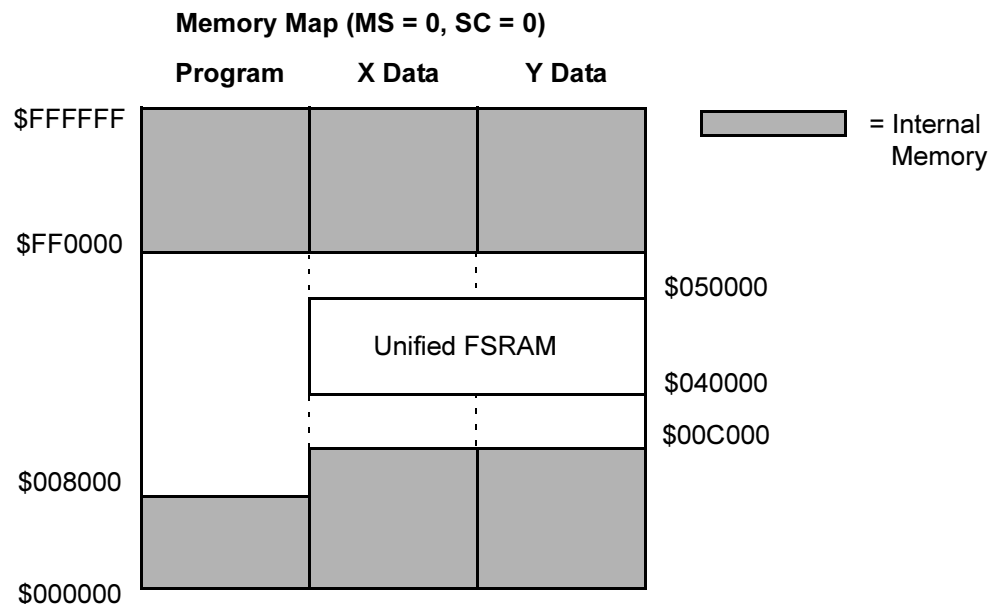
### 3.3.1.2 Example: Programming AAR0

As mentioned above, the FSRAM activity is controlled by the DSP56321 pin AA0 and the corresponding AAR0. AAR0 controls the external access type, the memory type, and which external memory addresses access the FSRAM. Figure 3-4 shows the memory map that is attained with the AAR0 settings described in this example.

**Note:** In this example, the memory switch bit in the operating mode register (OMR) is cleared and the 16-bit compatibility bit in the status register is cleared.

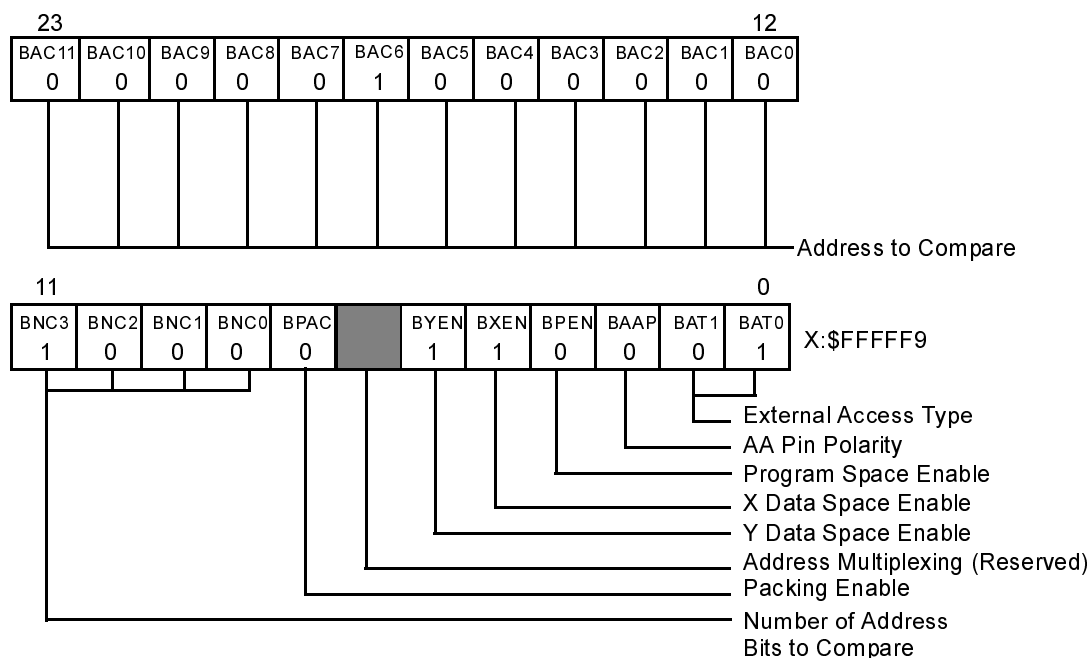
In Figure 3-4, the FSRAM responds to the 64K of X and Y data memory addresses between \$040000 and \$04FFFF. However, with the unified memory map, accesses to the same external memory location are treated as accesses to the same memory cell.

A priority mechanism exists among the four AAR control registers. AAR3 has the highest priority and AAR0 had the lowest. Bit 14 of the OMR, the address priority disable (APD) bit, controls which AA pins are asserted when a selection conflict occurs (i.e., the external address matches the address and the space that is specified in more than one AAR). If the APD bit is cleared when a selection conflict occurs, only the highest priority AA pin is asserted. If the APD bit is set when a selection conflict occurs, the lower priority AA pins are asserted in addition to the higher priority AA pin. For this example, only one AA pin must be asserted, AA0. Thus, the APD bit can be cleared.



**Figure 3-4. Example Memory Map with the Unified External Memory**

Figure 3-5 shows the settings of AAR0 for this example. The external access type bits (BAT1 and BAT0) are set to 0 and 1, respectively, to denote FSRAM access. The address attribute polarity bit (BAAP) is cleared to define AA0 as active low. Bit 6 (BAM) of the AAR is reserved and should be written with zero only. Packing is not needed with the FSRAM; thus, the packing enable bit (BPAC) is cleared to disable this option.



**Figure 3-5. Address Attribute Register AAR0**

The P, X data, and Y data space Enable bits (BPEN, BXEN, and BYEN) define whether the FSRAM is activated during external P, X data, or Y data space accesses, respectively. For this example, the BXEN and BYEN bits are set, and BPEN is cleared to allow the FSRAM to respond to X and Y data memory accesses only.

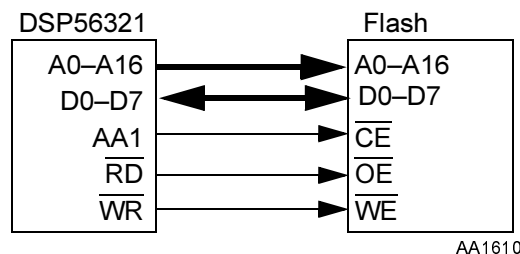
The number of address bits to compare BNC(3:0) and the address to compare bits BAC(11:0) determine which external memory addresses access the FSRAM. The BNC bits define the number of upper address bits that are compared between the BAC bits and the external address to determine if the FSRAM is accessed. For this example, the FSRAM is assigned to respond to addresses between \$040000 and \$04FFFF. Thus, the BNC bits are set to \$8 and the BAC bits are set to \$040. If the eight most significant bits of the external address are 00000100, the FSRAM is accessed.

### 3.3.2 Flash

The DSP56321EVM uses an Atmel AT29LV010A-15JC chip (U13) to provide a 128K× 8-bit CMOS Flash for stand-alone operation (i.e., startup boot operation without accessing the DSP56321 through the JTAG/OnCE port). The AT29LV010 uses a 3.3 V power supply and has a read access time of 150 ns.

#### 3.3.2.1 Flash Connections

The basic connection for the Flash is shown in Figure 3-6.



**Figure 3-6. Flash Connections**

The flash address pins (A0–A16) connect the respective port A address pins on the DSP. The flash data input/output pins D0–D7 are connected to the DSP56321 D0–D7 pins. The flash write enable ( $\overline{WE}$ ) and output enable ( $\overline{OE}$ ) lines connect the DSP56321 write ( $\overline{WR}$ ) and read ( $\overline{RD}$ ) enable lines, respectively. Address attribute 1 (AA1) generates the flash chip enable,  $\overline{CE}$ .

#### 3.3.2.2 Programming for Stand-Alone Operation

The DSP56321 mode pins determine the chip operating mode and start-up procedure when the DSP56321 exits the reset state. The switch SW6 resets the DSP56321 by asserting and then clearing the  $\overline{RESET}$  pin of the DSP56321. The mode pins MODA,

MODB, MODC, and MODD are sampled as the DSP56321 exits the reset state. The mode pins for the DSP56321EVM are controlled by microswitch block SW3 shown in Figure 3-1 on page 3-3 and Table 3-2 on page 3-9. The DSP56321 boots from the Flash after reset if Microswitch block SW3 switches 2,3-”On” while switches 1,4-”Off” (Mode 1: MODA and MODD are set, and MODB and MODC are cleared). The DSP56321 moves the 8-bit data from the flash into internal SRAM and then executes it.

### 3.4 Mode Selector

Boot Up mode selection for the DSP56321 is made by switch selections on Micro-switch. Refer to Table 3-2 for Micro-switch block SW3 switch selections options.

**Table 3-2. Boot Mode Selection Options**

Mode Number	SW3 “Mode Select”				Boot Mode Selected
	D (Sw4)	C (Sw3)	B (Sw2)	A (Sw1)	
8	Off	On	On	On	Jump to program at \$008000
9	Off	On	On	Off	Bootstrap from byte-wide memory
10	Off	On	Off	On	Bootstrap from SCI
12	Off	Off	On	On	HI08 bootstrap in ISA/DSP56300 mode
13	Off	Off	On	Off	HI08 Bootstrap in HC11 non-multiplexed bus mode
14	Off	Off	Off	On	HI08 Bootstrap in 8051 multiplexed bus mode.
15	Off	Off	Off	Off	HI08 Bootstrap in MC68302 bus mode.

### 3.5 Audio Codec

The DSP56321EVM analog section uses Crystal Semiconductor’s CS4218-KQ for two channels of 16-bit A/D conversion and two channels of 16-bit D/A conversion. Refer to Figure 3-1 on page 3-3 for the location of the codec on the DSP56321EVM and to Figure 3-2 on page 3-4 for a functional diagram of the codec within the evaluation module. The CS4218 uses a 3.3 V digital power supply and a 5 V analog power supply.

The CS4218 is driven by a 12.288 MHz signal at the codec Master Clock (CLKIN) input pin. The oscillator Y1 creates a 12.288 MHz signal which is buffered and sent to the codec. Refer to the CS4218 data sheet included with this kit for more information.

The CS4218 is very flexible, offering selectable sampling frequencies between 8 kHz and 48 kHz. The sampling frequency is selected using Micro-switch block SW1. Table 3-3

shows the Micro-switch block SW1 switches positions that select the possible sampling frequencies for the DSP56321EVM.

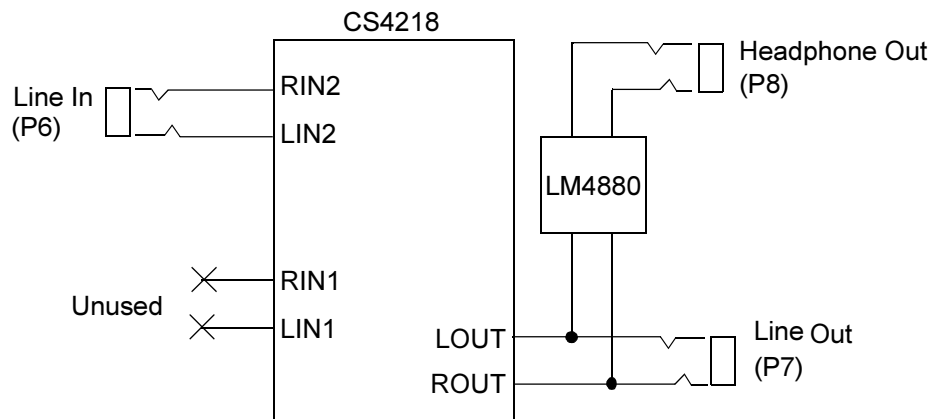
**Table 3-3. CS4218 Sampling Frequency Selection**

SW1 "Codec Mode"			Sampling Frequency (kHz)
Sw1(MF6)	Sw2 (MF7)	Sw3 (MF8)	
On	On	On	48.0
On	On	Off	32.0
On	Off	On	24.0
On	Off	Off	19.2
Off	On	On	16.0
Off	On	Off	12.0
Off	Off	On	9.6
Off	Off	Off	8

The codec is connected to the DSP56321 ESSIO through the shorting jumpers on J22 and to ESSII through the shorting jumpers on J24 shown in Figure 3-1 on page 3-3. Jumper block J24 connects the ESSII pins of the DSP56321 to the control pins of the CS4218. Jumper block J22 connects the ESSIO pins of the DSP56321 to the data pins of the CS4218. By removing these jumpers, the user has full access to the ESSIO and ESSII pins of the DSP56321. The following sections describe the connections for the analog and digital sections of the codec.

### 3.5.1 Codec Analog Input/Output

The DSP56321EVM contains 1/8-inch stereo jacks for stereo input, output, and headphones. Figure 3-7 shows the analog circuitry of the codec.



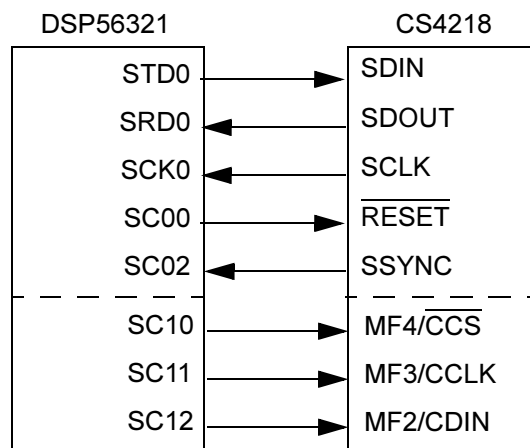
**Figure 3-7. Codec Analog Input/Output Diagram**

The stereo jack labelled LINE IN (P6) on the DSP56321EVM connects to the codec right and left input pins, RIN2 and LIN2. Standard line level inputs are  $2 V_{pp}$  and the codec requires that input levels be limited to  $1 V_{pp}$ . Thus, a voltage divider forms a 6 dB attenuator between P6 and the CS4218.

The codec right and left channel output pins, ROUT and LOUT, provide their output analog signals, through the stereo jack labelled LINE OUT (P7) on the DSP56321EVM. The outputs of the codec are also connected to the stereo jack labelled Headphone Out (P8) on the DSP56321EVM through National Semiconductor's LM4880M dual audio power amplifier at U20. The headphone stereo jack permits direct connection of stereo headphones to the DSP56321EVM.

## 3.5.2 Codec Digital Interface

Figure 3-8 shows the digital interface to the codec. Table 3-4 and Table 3-5 show the jumper selections to Enable/Disable the codec's digital signals.



**Figure 3-8. Codec Digital Interface Connections**



Table 3-4. J22 Jumper Block Options

J22 Pin #	DSP Signal Name	J22 Pin #	Codec Signal Name
1	SCK0	2	SCLK
3	SC00	4	$\overline{\text{RESET}}$
5	STD0	6	SDIN
7	SRD0	8	SDOUT
9	SC01	10	—
11	SC02	12	SSYNC

Table 3-5. J24 Jumper Block Options

J24 Pin #	DSP Signal Name	J24 Pin #	Codec Signal Name
1	SCK1	2	—
3	SC10	4	$\overline{\text{CCS}}$
5	STD1	6	—
7	SRD1	8	—
9	SC12	10	CDIN
11	SC11	12	CCLK

The serial interface of the codec transfers digital audio data and control data into and out of the device. The codec communicates with the DSP56321 through the ESSIO for the data information and through the ESSII for the control information. The codec has three modes of serial operation that are selected by the serial mode select SMODE1, SMODE2, and SMODE3 pins. The SMODE pins on the DSP56321EVM are set to enable serial mode 4, which separates the audio data from the control data. The SMODE pins are also set to enable the master sub-mode with 32-bit frames, the first 16 bits being the left channel, and the second 16 bits being the right channel.

The DSP56321 ESSIO transfers the data information to and from the codec. The DSP56321 serial transmit data (STD0) pin transmits data to the codec. The DSP56321 serial receive data (SRD0) pin receives data from the codec. These two pins are connected to the codec serial port data in (SDIN) and serial port data out (SDOUT) pins, respectively. In master sub-mode, the codec serial port clock (SCLK) pin provides the serial bit rate clock for the ESSIO interface. It is connected to the DSP56321 bidirectional serial clock (SCK0) pin. The DSP56321 serial control 0 (SC00) pin is programmed to control the codec reset signal  $\overline{\text{RESET}}$ . The serial control 2 (SC02) pin is connected to the

codec serial port sync signal (SSYNC) signal. A rising edge on SSYNC indicates that a new frame is about to start.

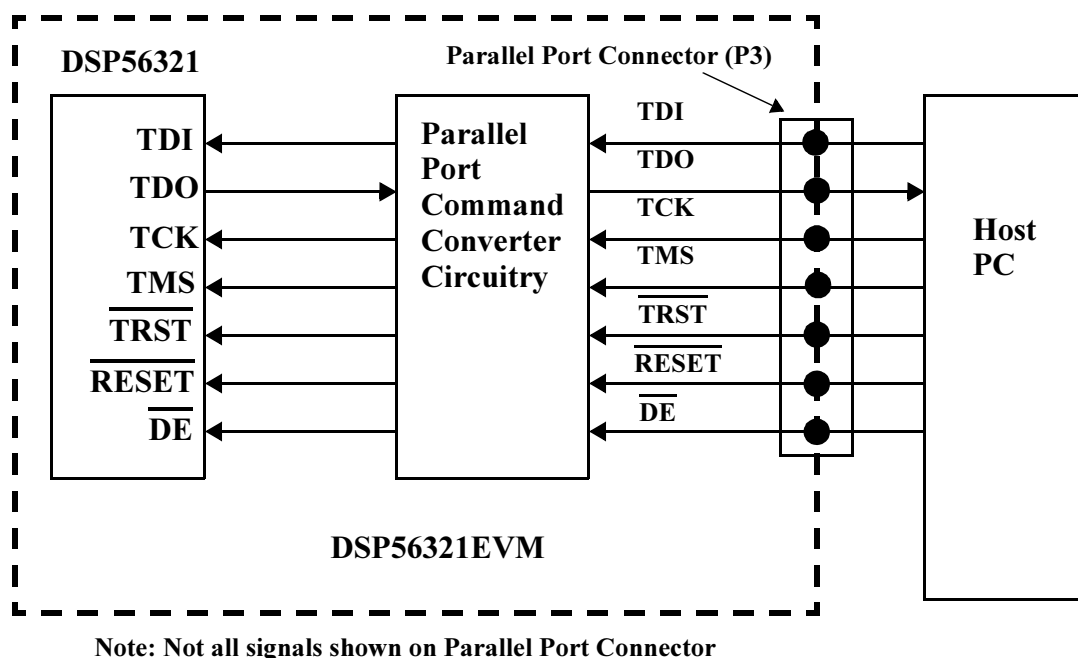
The DSP56321 ESS1 pins are used as general purpose I/O (GPIO) signals to transfer the control data to the codec. The control data needs to be transferred only when it changes. The DSP56321 serial control 0 (SC10) pin is programmed to control the codec multi-function pin 4 or the control data chip select pin, MF4/ $\overline{\text{CCS}}$ . This pin must be low for entering control data. The serial control 1 (SC11) pin connects to the codec multi-function pin 3 or the control data clock pin, MF3/CCLK. The control data is inputted on the rising edge of CCLK. The serial control 2 (SC12) pin is connected to the codec multi-function pin 2 or the control data input pin, MF2/CDIN. This pin contains the control data for the codec.

### 3.6 Command Converter

The DSP56321EVM provides an on-board command converter for use with Motorola's DSP56300 Debugger. The on-board command converter communicates with the host PC through a parallel DB25 connector and the PC's parallel printer port. The command converter receives commands from the host PC. The set of commands may include read data, write data, reset OnCE module, reset DSP56321, request JTAG interface, or release OnCE module. The command conversion is performed in software within the Motorola DSP56300 Debugger on the Host PC. This software interprets the commands and sends a sequence of instructions to the DSP56321's JTAG/OnCE port. The DSP56321 may then continue to receive data or it may transmit data back to the host PC. The on-board command converter circuitry is enabled by shorting pins 1-2 of the jumper J9. Table 3-6 shows the JTAG enable/disable options. Refer to Figure 3-1 on page 3-3 for the location of J9 on the DSP56321EVM and to Figure 3-2 on page 3-4 for a functional diagram. Figure 3-9 is a block diagram of the parallel port command converter interface. Table 3-8 shows the DB25 connector pinout.

**Table 3-6. On-Board JTAG Enable/Disable Option**

J9	Option Selected
1-2	On-Board Command converter enabled (On-Board <b>JTAG</b> interface <b>Disabled</b> )
2-3	On-Board Command converter disabled (On-Board <b>JTAG</b> interface <b>Enabled</b> )



**Figure 3-9. Parallel Port Interface**

As an alternative to the on-board command converter, the DSP56321EVM 14-pin **JTAG/OnCE connector J23** allows the user to connect an external JTAG command converter board directly to the DSP56321EVM. **Jumper J9 (2-3) must be shorted to enable this option!** Pin 8 has been removed from J23 so that the cable cannot be connected to the DSP56321EVM incorrectly. Table 3-7 shows the JTAG/OnCE (J23) connector pinout. The JTAG cable from the external JTAG command converter is similarly keyed so that the cable cannot be connected to the DSP56321EVM incorrectly.

**Table 3-7. JTAG/OnCE (J23) Connector Pinout**

Pin Number	DSP Signal Name	Pin Number	DSP Signal Name
1	TDI	2	GND
3	TDO	4	GND
5	TCK	6	GND
7	—	8	KEY-PIN
9	$\overline{\text{RESET}}$	10	TMS
11	+3.3 V	12	—
13	$\overline{\text{DE}}$	14	$\overline{\text{TRST}}$

**Table 3-8. Command Converter Connector (P3) Pinout**

Pin Number	Signal Name	Pin Number	Signal Name
1	—	14	—
2	$\overline{\text{RESET}}$	15	IDENT
3	TMS	16	—
4	TCK	17	—
5	TDI	18	GND
6	$\overline{\text{TRST}}$	19	GND
7	$\overline{\text{DE}}$	20	GND
8	IDENT	21	GND
9	PORT_VCC	22	GND
10	—	23	GND
11	TDO	24	GND
12	—	25	GND
13	OUT_VCC		

## 3.7 Off-Board Interfaces

The DSP56321EVM User's Manual provides interfaces with off-board devices via its on-chip peripheral ports. Most of the DSP ports are connected to headers on the board to facilitate direct access to these pins by using connectors or jumpers.

### 3.7.1 Serial Communication Interface Port (SCI)

Connection to the DSP's SCI port can be made at J20. Refer to Table 3-9 for pinout. The signals at J20 are +3.3 V signals. Refer to Table 3-10 to route the DSP's SCI signals through an RS-232 level converter to P1. The pinout of P1 is shown in Table 3-11.

By installing a jumper 5-6 at J20, the SCI port will be clocked by the on-board 153.6 kHz oscillator instead of being clocked externally via the SCI Header J20 pin 5.

**Table 3-9. SCI Header (J20) Pinout**

Pin Number	DSP Signal Name	Pin Number	Interface Signal Name
1	RxD	2	RS232(RxD)
3	TxD	4	RS232(TxD)
5	SCLK	6	Osc. out 153.6kHz
7	RESET	8	RS232(DSR)

**Table 3-10. J12 Jumper Options**

J7	DSP Signal Name
1—2	RxD
3—4	TxD
5—6	SCLK
7—8	RESET

**Table 3-11. DSP Serial Port (P1) Connector Pinout**

Pin Number	DSP Signal Name	Pin Number	DSP Signal Name
1	—	6	—
2	TxD	7	—
3	RxD	8	—
4	$\overline{\text{RESET}}$	9	—
5	GND		

### 3.7.2 Enhanced Synchronous Serial Port 0 (ESSI0)

Connection to the DSP's ESSI0 port can be made at J22. Refer to Table 3-12 for the header's pinout.

**Table 3-12. ESSI0 Header (J22) Pinout**

Pin Number	DSP Signal Name	Pin Number	Codec Signal Name
1	SCK0	2	SCLK
3	SC00	4	$\overline{\text{RESET}}$
5	STD0	6	SDIN
7	SRD0	8	SDOUT
9	SC01	10	—
11	SC02	12	SSYNC

### 3.7.3 Enhanced Synchronous Serial Port 1 (ESSI1)

Connection to the DSP's ESSI1 port can be made at J24. Refer to Table 3-13 for the header's pinout.

**Table 3-13. ESSI1 Header (J24) Pinout**

Pin Number	DSP Signal Name	Pin Number	Codec Signal Name
1	SCK1	2	—
3	SC10	4	$\overline{\text{CCS}}$
5	STD1	6	—
7	SRD1	8	—
9	SC12	10	CDIN
11	SC11	12	CCLK

### 3.7.4 Host Port (HI08)

Connection to the DSP's HI08 port can be made at J14. Refer to Table 3-14 for the header's pinout.

**Table 3-14. HI08 Header (J14) Pinout**

Pin Number	DSP Signal Name	Pin Number	DSP Signal Name
1	H0	2	H1
3	H2	4	H3
5	H4	6	H5
7	H6	8	H7
9	HA0	10	HA1
11	HA2	12	$\overline{\text{HCS}}$
13	$\overline{\text{HDS}}$	14	$\overline{\text{HACK}}$
15	$\overline{\text{HREQ}}$	16	HRW
17	$\overline{\text{RESET}}$	18	GND
19	+3.3V	20	GND

### 3.7.5 Control Bus

Connection to the DSP's control bus control signals can be made at J10. Refer to Table 3-15 for header's pinout.

**Table 3-15. Control Bus Signal Header (J10) Pinout**

Pin Number	DSP Signal Name	Pin Number	DSP Signal Name
1	$\overline{\text{WR}}$	2	$\overline{\text{RD}}$
3	$\overline{\text{BB}}$	4	$\overline{\text{BG}}$
5	$\overline{\text{TA}}$	6	$\overline{\text{BR}}$
7	—	8	—
9	—	10	—
11	AA0	12	AA1
13	AA3	14	AA2
15	+3.3V	16	GND



## 3.7.6 Address Bus

Connection to the DSP's address bus signals can be made at J13. Refer to Table 3-16 for header's pinout.

**Table 3-16. Address Bus Signal Header (J13) Pinout**

Pin Number	Signal Name	Pin Number	Signal Name
1	A0	2	A1
3	A2	4	A3
5	A4	6	A5
7	A6	8	A7
9	A8	10	A9
11	A10	12	A11
13	A12	14	A13
15	A14	16	A15
17	A16	18	A17
19	GND	20	GND
21	+3.3V	22	+3.3V

### 3.7.7 Data Bus

Connection to the DSP's data bus signals can be made at J21. Refer to Table 3-17 for header's pinout.

**Table 3-17. Data Bus Signal Header (J21) Pinout**

Pin Number	Signal Name	Pin Number	Signal Name
1	D0	2	D1
3	D2	4	D3
5	D4	6	D5
7	D6	8	D7
9	D8	10	D9
11	D10	12	D11
13	D12	14	D13
15	D14	16	D15
17	D16	18	D17
19	D18	20	D19
21	D20	22	D21
23	D22	24	D23
25	GND	26	GND
27	+3.3V	28	+3.3V

## 3.8 Power Supplies

The main power input, 10-12V AC/DC at 1A, to the DSP56321EVM is through a 2.1mm coax power jack. The DSP56321EVM provides +3.3VDC voltage regulation for the DSP, FSRAM memory, Flash memory, codec, parallel JTAG interface and supporting logic. Power applied to the DSP56321EVM is indicated with a Power-On LED's +5Vin. (Green), 3.3V(Red) and 1.5V(Yellow). The DSP56321EVM also provides +5VDC voltage regulation for use by the codec and oscillators Y1, Y2. Additionally, the DSP56321EVM provides +1.5VDC for use by the DSP core.

Jumper J7(1-2), the Core Measurement Jumper connects the +1.5VDC supply to the DSP core. This jumper provides a convenient point where the average current being consumed by the DSP core can be measured and thus facilitate calculation of the DSP core's average power consumption.

Jumper J8(1-2), the 3.3V Measurement Jumper connects the +3.3VDC supply to the all DSP peripheral circuits. This jumper provides a convenient point where the average current being consumed by the DSP periphery can be measured and thus facilitate calculation of the DSP average power consumption.

Jumper J26(1-2), the 5V Digital/Analog Jumper connects the +5VDC supply to the all DSP56321EVM circuits. This jumper provides a convenient point where the average current being consumed by the board can be measured and thus facilitate calculation of the DSP56321EVM average power consumption.

Jumpers J1...J6(2-3), the DSP Peripheral Parts Measurement Jumpers are connects the +3.3VDC supply to the different peripheral parts of the DSP (like Address interface, Data interface etc.). These jumpers provides a convenient points where the average current being consumed by the different peripheral parts of the DSP can be measured and thus facilitate calculation of the DSP average power consumption.

## 3.9 Test Points

### 3.9.1 Ground Test Points

Four on-board digital ground (DGND) and one analog ground (AGND) are provided to facilitate reference measurements. These are located near the corners of the PCB to allow easy access for oscilloscope probe ground straps.

### 3.9.2 Auxiliary Test Points

Some auxiliary test points like (+5Vrect., DSP\_SCKL, DSP\_CLK, Codec\_CLK, +5V, +5Van.) are provided on the board to facilitate convenient measuring points of internal DC voltages and clock sources.

### 3.9.3 Timer Output Test Points

Additionally, test points are provided for the three timer output pins of the DSP56321. Table 3-18 provides a list of the connections. These test points are located near the P3 connector.

**Table 3-18. Timer Output Test Points(TIO1-TIO3)**

Test Point	DSP Pin Function
TIO0	Timer 0 output pin
TIO1	Timer 1 output pin
TIO2	Timer 2 output pin

## 3.9.4 External Interrupts Test Points

Test points are provided for the DSP56321's external interrupt input pins at jumper J25. Table 3-19 provides the pinout of J25. Additionally, switch SW4 is connected to IRQA while SW5 is connected to IRQD.

**Table 3-19. Interrupt Request Test Points(J25)**

Pin	Test Point	DSP Pin Function
1	NMI	DSP Non Maskable Interrupt
2	IRQA	DSP Interrupt Request A
3	IRQD	DSP Interrupt Request D
4	IRQB	DSP Interrupt Request B
5	GND	-
6	IRQC	DSP Interrupt Request C

### 3.10 Debug LED's

Three on-board Light-Emitting Diodes (LED's) are provided to allow real-time debugging for user programs. These LED's will allow the programmer to monitor program execution without having to stop the program during debugging. LD9 is controlled by the timer output pin TIO0. LD11 is controlled by the timer output pin TIO1 and LD10 is controlled by TIO2. The TIO pins are data outputs when GPIO mode is enabled for the timers and the DIR bit is set in the Timer Control/Status Register (TCSR). Setting the DO bit to a logic 1 will turn on the associated LED. Table 3-20 outlines which LED is associated with which timer output pin.

**Table 3-20. LED Indicators (LD9, LD10 & LD11)**

LED	Associated Timer Output Pin
LD9 (Red)	Timer Output 0 (TIO0)
LD11 (Yellow)	Timer Output 1 (TIO1)
LD10 (Green)	Timer Output 2 (TIO2)

### 3.11 Reset

The DSP56321EVM provides a power-on reset function (Dallas Semiconductor DS1818R, designated U16) for a clean and dependable RESET signal. A reset of the DSP can be generated from the JTAG connector, the Parallel JTAG Interface and the user RESET push-button (SW6).

### 3.12 Clock Source

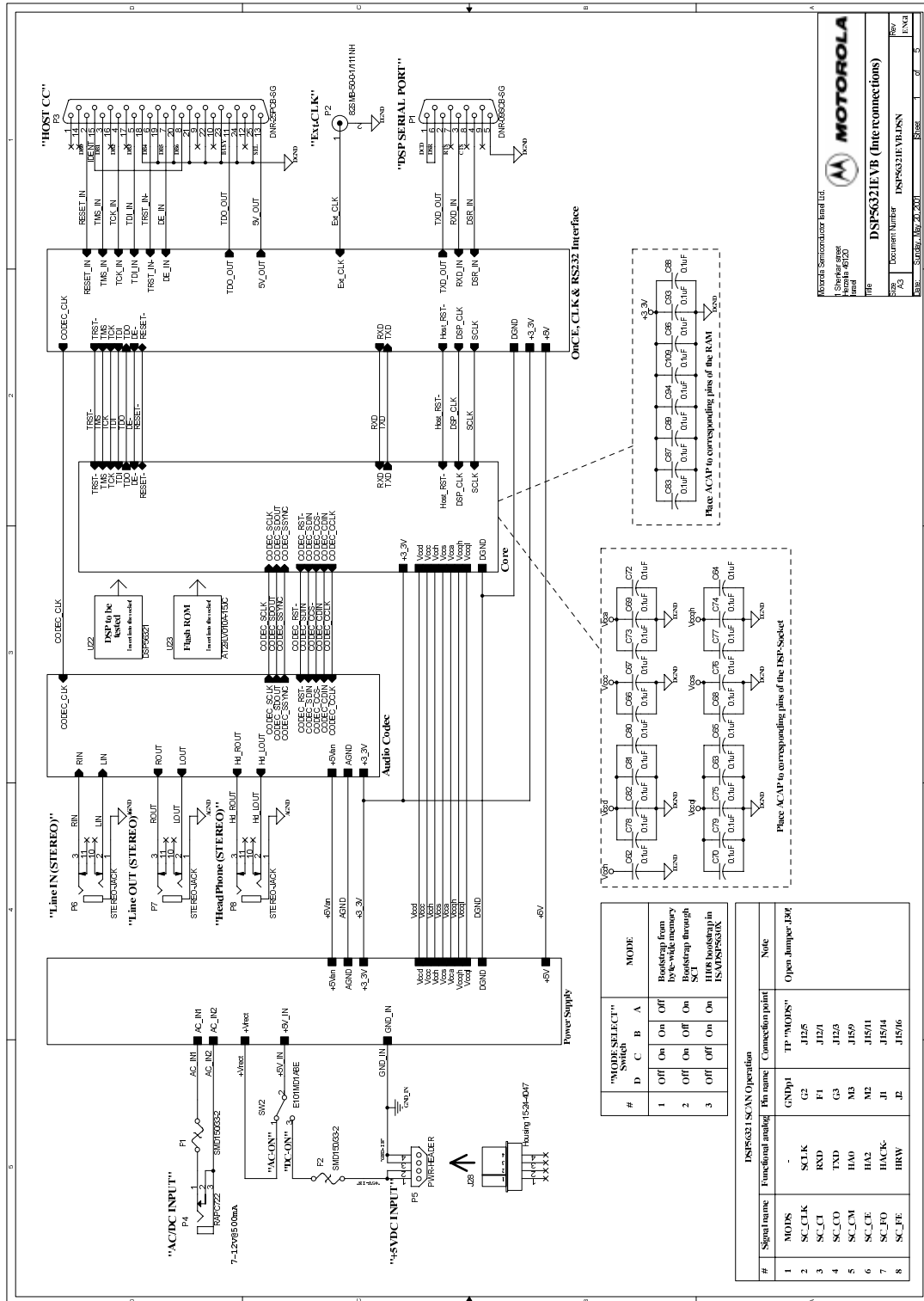
The DSP56321EVM uses a 12.288MHz buffered oscillator Y1 for Codec clock source and 19.6606MHz oscillator Y2 connected to DSP chip via jumpers J16 (2-3)- "CLK\_Select" and J15 (2-3)- "CLK\_Source". The DSP56321 PLL can be programmed to multiply the input frequency to achieve the desired operating speed.



# **Appendix A**

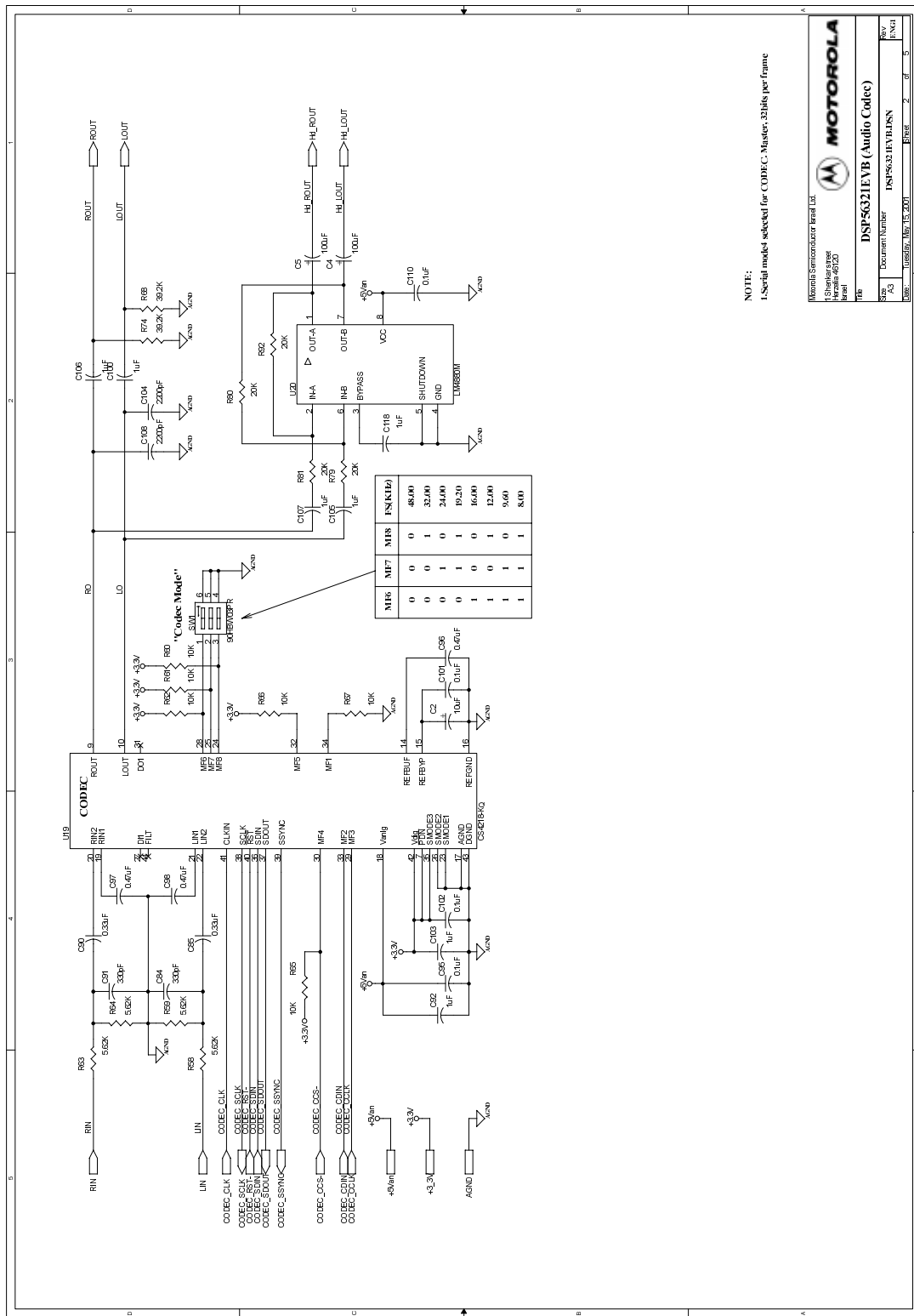
## **DSP56321 EVM Schematics**

Figure A-1. Interconnections





### Figure A-2. AudioCodec



**NOTE:**  
1. Serial mode<sup>4</sup> selected for CODEC Master, 32bits per frame


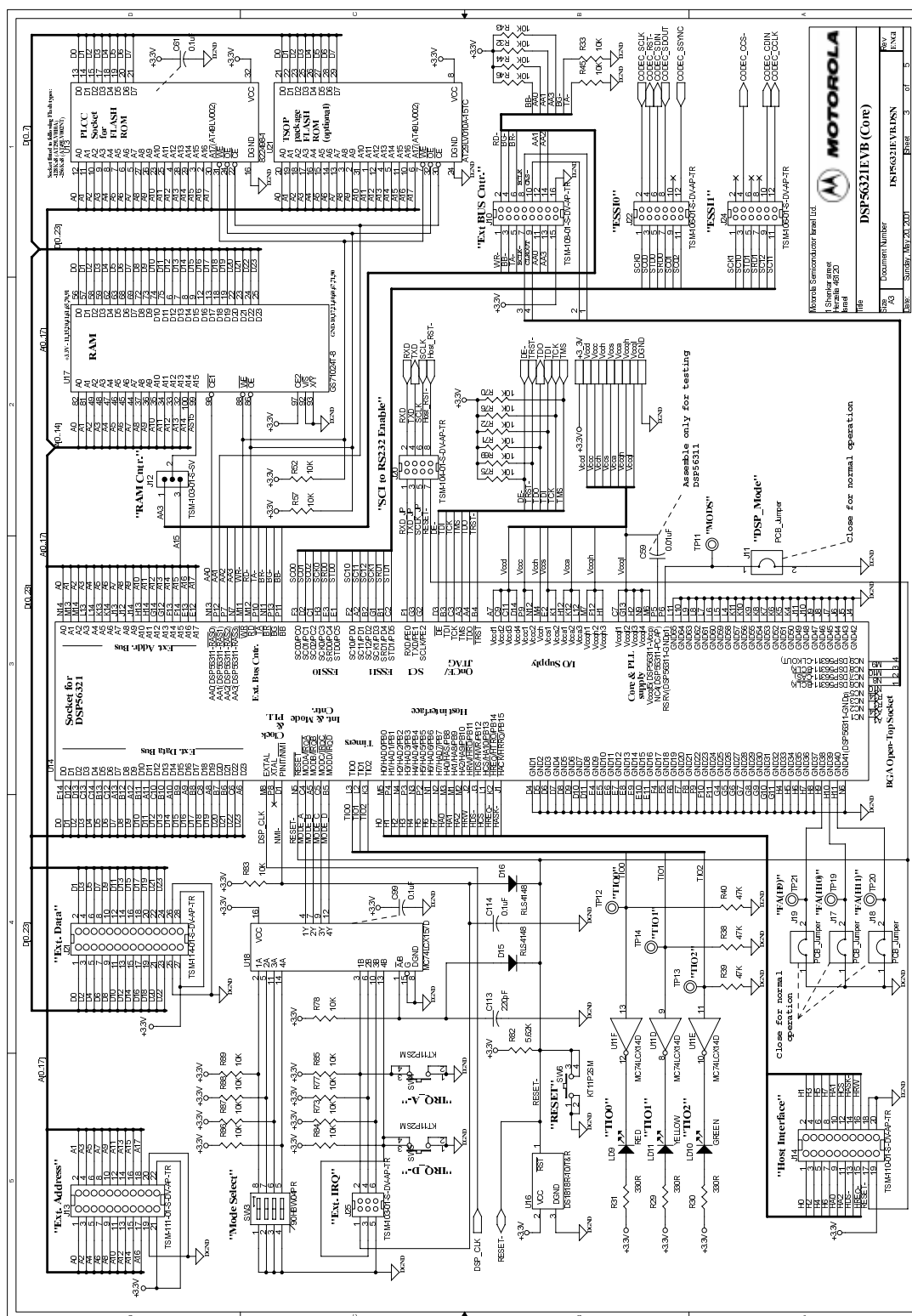
Motorola Semiconductor Israel Ltd.				<b>MOTOROLA</b>	
1 Shenkar Street Herzlia 46120 Israel					
<b>DSP56321EVB (Audio Codec)</b>					
Title					
Size	Document Number	DSP56321EVB.DSN		Rev	ENGL
A3					
Qty	Unit(s)	Rev	2	of	5
Issued: May 15, 2001					

Figure A-3. Core



### Figure A-4. OnCE, CLK & RS232 Interface

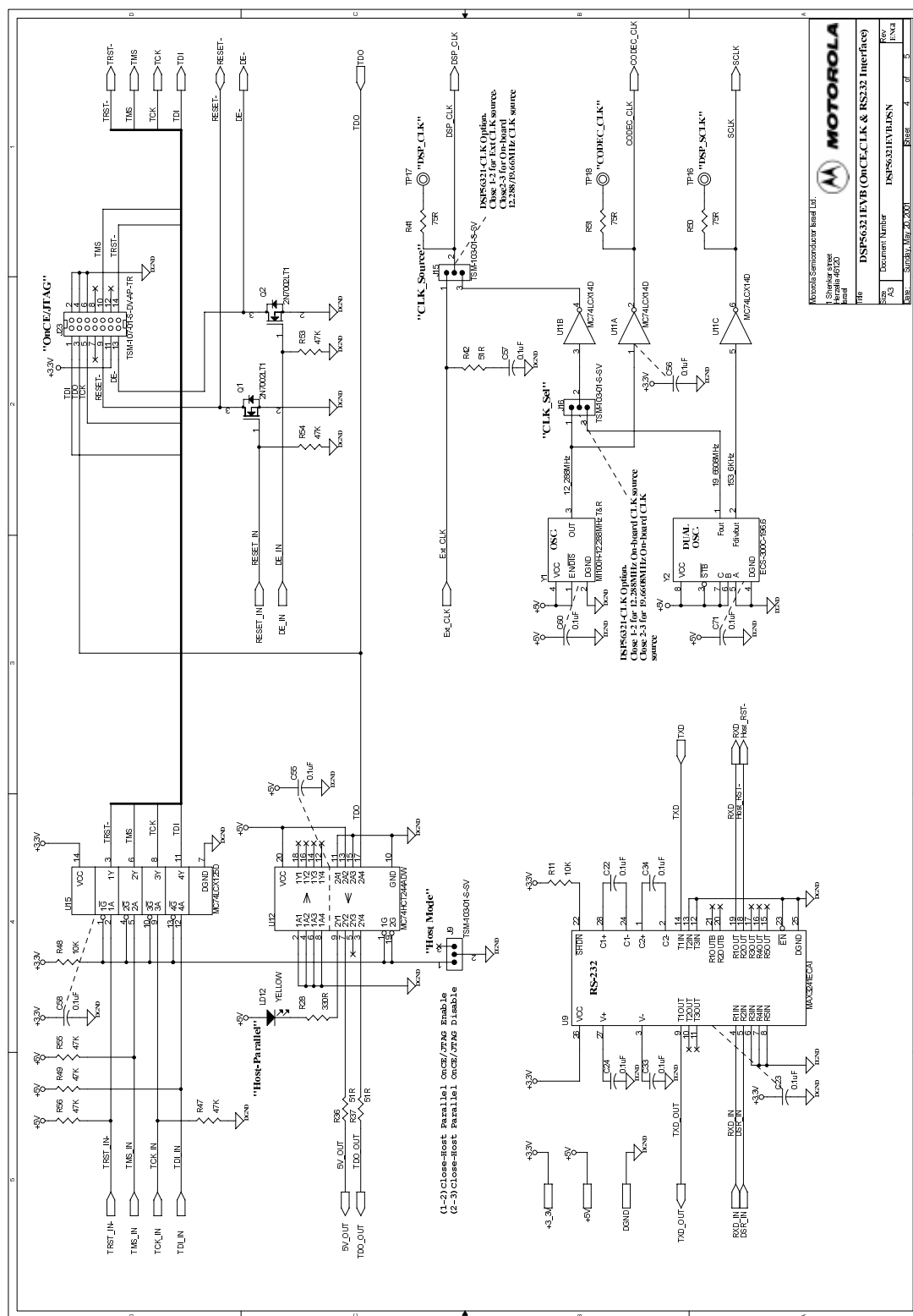
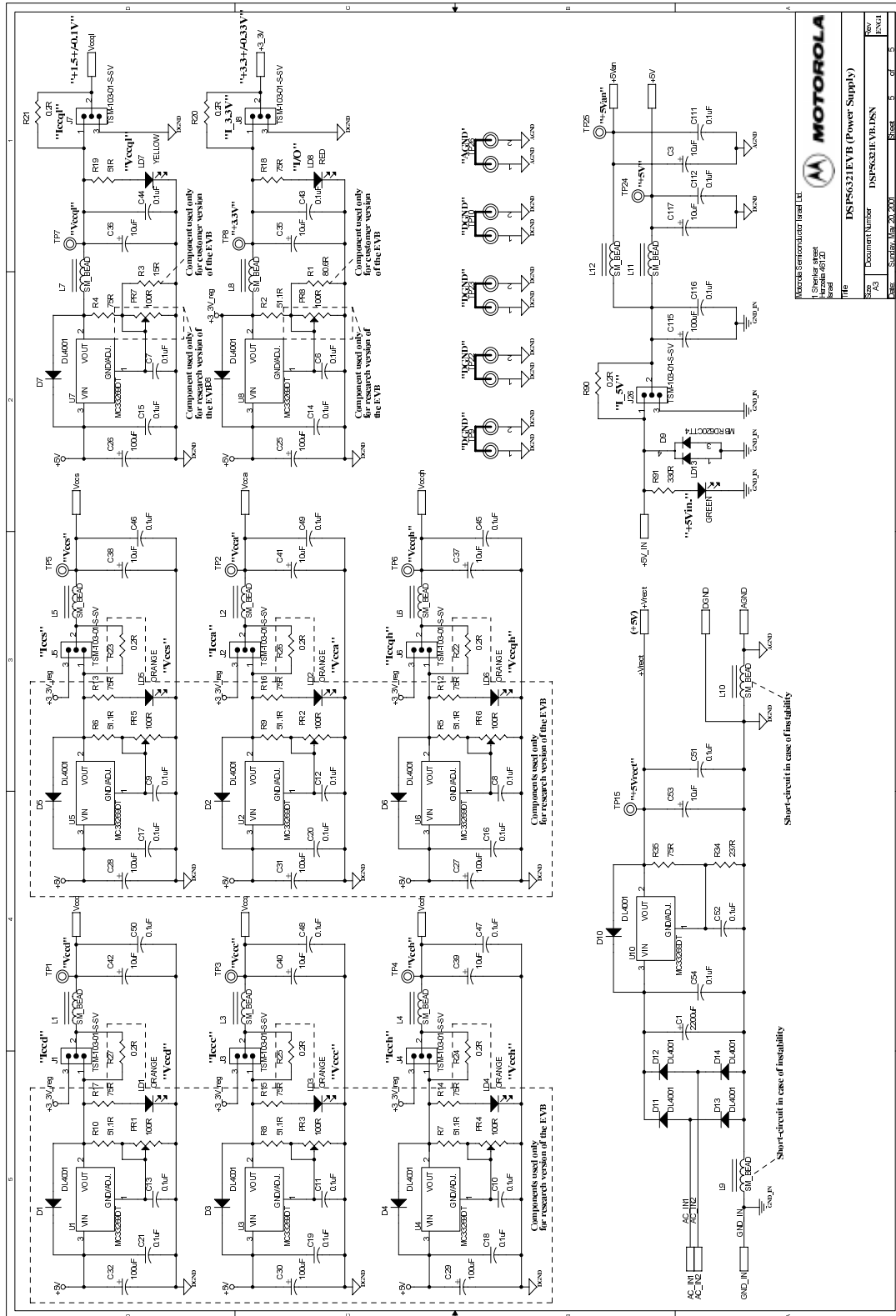


Figure A-5. Power Supply



# Appendix B

## DSP56321EVM Parts List

### B.1 Parts Listing

The following table contains information on the parts and devices on the DSP56321EVM.

**Table B-1. DSP56321EVM Parts List**

Qty	Designator	Manufacturer	Part Number	Description
3	U7, U8, U10	ON Semi.	MC33269DT	Adj Regulator
1	U9	Maxim	MAX3241ECAI	RS-232 Transceiver
1	U11	ON Semi.	MC74LCX14D	Hex Inverter
1	U12	ON Semi.	MC74HCT244ADW	Octal Buffer
1	U13	AMP	822498-1	PLCC32 Socket
1	U14	3M	2196-9306-0A-2401 Type0	BGA Open-Top Socket
1	U15	ON Semi.	MC74LCX125D	Multiplexer
1	U16	Dallas Semi.	DS1818R-10	Power-On-Reset
1	U17	GSI	GS71024T-8	FSRAM
1	U18	ON Semi.	MC74LCX157D	QUAD 2in. Multiplexer
1	U19	Crystal Semi.	CS4218-KQ	Audio Codec
1	U20	National Semi.	LM4880M	Audio Amplifier
1	U22	Motorola	DSP56321	DSP
1	U23	Atmel	AT29LV010A-15TC	Flash
2	Q1 Q2	ON Semi.	2N7002LT1	Transistor
7	D7, D8, D10...D14	Diodes Inc.	DL4001	Diode
1	D9	ON Semi.	MBRD620CTT4	Power Rectifier
2	D15, D16	Rohm	RLS4148	Switching Diode

**Table B-1. DSP56321EVM Parts List (Continued)**

Qty	Designator	Manufacturer	Part Number	Description
3	LD7, LD11, LD12	Infineon	LYT670	Yellow LED
2	LD8, LD9	Infineon	LST676-P1	Red LED
2	LD10, LD13	Infineon	LGT670	Green LED
1	Y1	MMD Components	MI100H-12.288MHz	12.288 MHz Oscillator
1	Y2	ECS Inc.	ECS-300C-196.6	19.6608 MHz /153.6 KHz Oscillator
1	SW1	Grayhill	90HBW03PR	DIP-Switch
1	SW2	C&K	E101MD1ABE	Toggle Switch
1	SW3	Grayhill	90HBW04PR	DIP-Switch
3	SW4, SW5, SW6	C&K	KT11P2SM	Push Switch
1	P1	KCC_Keltron	DNR-09SCB-SG	DB-9 Female Connector
1	P3	KCC_Keltron	DNR-25PCB-SG	DB-25 Male Connector
1	P4	Switchcraft	RAPC-722	2.1 mm DC Power Jack
3	P6, P7, P8	Switchcraft	35RAPC4BHN2	3.5 mm Miniature Stereo Jack
13	J1...J9, J12, J15, J16, J26	Samtec	TSM-103-01-S-SV	Header 3 pin single row
1	J10	Samtec	TSM-108-01-S-DV-AP-TR	Header 16 pin double row
1	J13	Samtec	TSM-111-01-S-DV-AP-TR	Header 22 pin double row
1	J14	Samtec	TSM-110-01-S-DV-AP-TR	Header 20 pin double row
1	J20	Samtec	TSM-104-01-S-DV-AP-TR	Header 8 pin double row
1	J21	Samtec	TSM-114-01-S-DV-AP-TR	Header 28 pin double row
2	J22, J24	Samtec	TSM-106-01-S-DV-AP-TR	Header 12 pin double row
1	J23	Samtec	TSM-107-01-S-DV-AP-TR	Header 14 pin double row
1	J25	Samtec	TSM-103-01-S-DV-AP-TR	Header 6 pin double row

**Table B-1. DSP56321EVM Parts List (Continued)**

Qty	Designator	Manufacturer	Part Number	Description
1	F1	Raychem	SMD150/33-2	Polyswitch
1	C1	Panasonic	EEXFC1E222N	2200 $\mu$ F Capacitor, 25VDC Electrolytic
12	C2,C3,C35...C42,C53, C117	Matsushita El.	10 $\mu$ F/25V	Capacitor, Tantalum
4	C4,C5,C115	Matsushita El.	100 $\mu$ F/10V	Capacitor, Tantalum
60	C20,C22...C24,C33,C34, C43...C52,C54...C56,C58, C60...C83,C86...C89, C93...C95,C99,C101, C102,C109...C112,C114, C116	Philips	0.1 $\mu$ F/50V	Capacitor, Ceramic
2	C84,C91	Philips	330pF, 5%, 50V NPO	Capacitor, Ceramic
2	C85, C90	Philips	0.33 $\mu$ F/16V	Capacitor, Ceramic
7	C92,C100,C103, C105...C107,C118	Philips	1.0 $\mu$ F/16V	Capacitor, Ceramic
3	C96...C98	Philips	0.47 $\mu$ F/50V, 10%	Capacitor, Ceramic
2	C104,C108	Philips	2200 pF/50V, 10%	Capacitor, Ceramic
1	C113	Philips	220 pF/50V, 5%	Capacitor, Ceramic
12	L1...L12	Fair-Rite Co.	2743021447	Ferrite SM-Bead
1	R1	Philips	80.6 $\Omega$ , 1%	Resistor
7	R2,R4,R18,R35,R41,R50, R51	Philips	75 $\Omega$ , 1%	Resistor
1	R3	Philips	15 $\Omega$ , 1%	Resistor
32	R11,R32,R33,R43...R46, R48,R52,R57,R60...R62, R65...R67,R69...R73, R75...R78,R83...R89	Philips	10 K $\Omega$ , 5%	Resistor
3	R19,R36,R37	Philips	51 $\Omega$ , 1%	Resistor
3	R20,R21,R90	Philips	0.2 $\Omega$ , 1%	Resistor
5	R28...R31,R91	Philips	330 $\Omega$ , 1%	Resistor
1	R34	Philips	237 $\Omega$ , 1%	Resistor
9	R38...R40,R47,R49, R53...R56	Philips	47K $\Omega$ , 5%	Resistor

**Table B-1. DSP56321EVM Parts List (Continued)**

Qty	Designator	Manufacturer	Part Number	Description
5	R58,R59,R63,R64,R82	Philips	5.62K $\Omega$ , 1%	Resistor
2	R68,R74	Philips	39.2K $\Omega$ , 1%	Resistor
4	R79...R81,R92	Philips	20K $\Omega$ , 5%	Resistor



# INDEX

## Symbols

" C-5  
 # C-9  
 #< C-9  
 #> C-9  
 % C-4  
 \* C-6  
 ++ C-6, C-7  
 ; C-1  
 ;; C-2  
 < C-7  
 << C-7  
 > C-8  
 ? C-3  
 @ C-6  
 \ C-2  
 ^ C-4

## A

A/D converter 3-8  
 AAR0  
   programming 3-5  
 Address Attribute Pin Polarity Bit, BAAP 3-6  
 Address Attribute Pin, AA1 3-7  
 Address Attribute Register, AAR0 3-5  
 Address Muxing Bit, BAM 3-6  
 Address Pins, A(0:17) 3-7  
 Address Priority Disable Bit, APD 3-5  
 Address to Compare Bits, BAC(11:0) 3-7  
 Addressing  
   I/O short C-7  
   immediate C-9  
   long C-8  
   long immediate C-9  
   short C-7  
   short immediate C-9  
 Analog Input/Output 3-9  
 Assembler 2-12  
   mode C-33  
   option C-34  
   warning C-40  
 assembler 2-1  
   control 2-8  
   data definition/storage allocation 2-8, 2-9  
   directives 2-8  
   listing control and options 2-10

macros and conditional assembly 2-11  
 object file control 2-10  
 options 2-6  
 significant characters 2-8  
 structured programming 2-11  
 symbol definition 2-8, 2-9

assembler control 2-8  
 assembler directives 2-8  
 assembler options 2-6  
 assembling the example program 2-12  
 assembling the program 2-5  
 assembly programming 2-1  
 AT29LV010A 3-7  
 audio codec 3-1, 3-8  
 audio interface cable 1-2  
 audio source 1-2

## B

Buffer  
   address C-10  
   end C-23

## C

Checksum C-37  
 codec 3-8  
   digital interface 3-9  
   digital interface connections 3-10  
 Codec Control Data Chip Select Pin, MF4/CCS 3-11  
 Codec Control Data Clock Pin, MF3/CCLK 3-11  
 Codec Control Data Input Pin, MF2/CDIN 3-11  
 Codec Digital Interface 3-9  
 Codec Left Channel Output Pin, LOUT 3-9  
 Codec Left Input #2 Pin, LIN2 3-9  
 codec Master Clock Pin, CLKIN 3-8  
 codec modes of serial operation 3-10  
 Codec Reset Pin, RESET 3-11  
 Codec Right Channel Output Pin, ROUT 3-9  
 Codec Right Input #2 Pin, RIN2 3-9  
 Codec Serial Port Clock Pin, SCLK 3-11  
 Codec Serial Port Data In Pin, SDIN 3-11  
 Codec Serial Port Data Out Pin, SDOUT 3-11  
 Codec Serial Sync Signal Pin, SSYNC 3-11  
 command converter 3-1, 3-11  
 command format  
   assembler 2-5  
 Comment C-14

- delimiter C-1
- object file C-13
- unreported C-2
- comment field 2-3
- Conditional assembly C-28, C-37
- Constant
  - define C-14, C-15
  - storage C-11
- Crystal Semiconductor CS4215 3-8
- CS4218 3-8
- Cycle count C-37

## D

- D/A converter 3-8
- Data Pins, D(0:23) 3-7
- data transfer fields 2-3
- Debugger 2-1, 2-17
  - running the 2-19
- Debugger software 2-17
- Debugger window display 2-18
- development process flow 2-1
- Directive C-10
  - .BREAK C-54
  - .CONTINUE C-55
  - .FOR C-55
  - .IF C-56
  - .LOOP C-57
  - .REPEAT C-57
  - .WHILE C-57
  - BADDR C-10
  - BSB C-11
  - BSC C-11
  - BSM C-12
  - BUFFER C-12
  - COBJ C-13
  - COMMENT C-14
  - DC C-14
  - DCB C-15
  - DEFINE C-5, C-16, C-37
  - DS C-17
  - DSM C-17
  - DSR C-18
  - DUP C-18
  - DUPA C-19
  - DUPC C-20
  - DUPF C-21
  - END C-22
  - ENDBUF C-23
  - ENDIF C-23
  - ENDM C-23
  - ENDSEC C-24
  - EQU C-24
  - EXITM C-25

- FAIL C-25
- FORCE C-26
- GLOBAL C-26
- GSET C-26
- HIMEM C-27
- IDENT C-27
- IF C-28
- in loop C-37
- INCLUDE C-29
- LIST C-29
- LOCAL C-30
- LOMEM C-30
- LSTCOL C-31
- MACLIB C-31
- MACRO C-32
- MODE C-33
- MSG C-33
- NOLIST C-34
- OPT C-34
- ORG C-41
- PAGE C-44
- PMACRO C-44
- PRCTL C-45
- RADIX C-45
- RDIRECT C-46
- SCSJMP C-46
- SCSREG C-47
- SECTION C-47
- SET C-50
- STITLE C-50
- SYMOBJ C-50
- TABS C-51
- TITLE C-51
- UNDEF C-51
- WARN C-51
- XDEF C-52
- XREF C-52
- Domain Technologies Debugger 1-1, 2-17
- DSP development tools 2-1
- DSP linker 2-12
- DSP56002 3-11
- DSP56300 Family Manual 3-1
- DSP56307 2-1
  - Chip Errata 3-2
  - Product Specification 1-1
  - Product Specification, Revision 1.02 3-1
  - Technical Data 1-1, 3-1
  - User's Manual 3-1
- DSP56307 Features 3-1
- DSP56307EVM
  - additional requirements 1-2
  - Component Layout 3-2
  - connecting to the PC 1-4
  - contents 1-1

- description 3-1
- features 3-1
- Flash PEROM 3-2
- functional block diagram 3-3
- installation procedure 1-2
- interconnection diagram 1-4
- memory 3-2
- power connection 1-4
- Product Information 1-1
- SRAM 3-2
- User's Manual 1-1

## E

- Enhanced Synchronous Serial Port 0 (ESSIO) 3-15
- Enhanced Synchronous Serial Port 1 (ESSI1) 3-15
- ESSIO 3-8
- ESSI1 3-8
- example
  - assembling the 2-12
- example program 2-3
- Expansion Bus Control 3-16
- Expression
  - address C-37
  - compound C-60
  - condition code C-58
  - formatting C-60
  - operand comparison C-59
  - radix C-45
  - simple C-58
- External Access Type Bits, BAT(1:0) 3-6

## F

- field
  - comment 2-3
  - data transfer 2-3
  - label 2-2
  - operand 2-3
  - operation 2-2
  - X data transfer 2-3
  - Y data transfer 2-3
- File
  - include C-29
  - listing C-38
- Flash 3-2
- Flash Address Pins, A(0:16) 3-7
- Flash Chip Enable Pin, CE 3-7
- Flash Data Pins, I/O(0:7) 3-7
- Flash Output Enable Pin, OE 3-7
- Flash PEROM 3-2, 3-7
  - connections 3-7
  - stand-alone operation 3-7
- Flash Write Enable Pin, WE 3-7

- format
  - assembler command 2-5
  - source statement 2-2
- Function C-6

## G

- GS71024T-10 3-3

## H

- headphones 1-2
- host PC 3-11
- host PC requirements 1-2
- Host Port (HI08) 3-16

## I

- Include file C-29

## J

- J1 1-3
- J4 1-3, 3-8
- J5 1-3, 3-8
- J6 3-13
- J7 3-8
- J8 1-3
- J9 3-4, 3-8
- JTAG 3-11

## K

- kit contents 1-1

## L

- Label
  - local C-38, C-40
- label field 2-3
- Line continuation C-2
- linker 2-1, 2-12
  - directives 2-16
  - options 2-13
- linker directives 2-16
- Listing file C-38
  - format C-31, C-37, C-38, C-40, C-44, C-51
  - sub-title C-50
  - title C-51
- LM4880 3-9
- Location counter C-6, C-41

## M

- Macro

- call C-38
  - comment C-37
  - definition C-32, C-38
  - directive C-32
  - end C-23
  - exit C-25
  - expansion C-38
  - library C-31, C-38
  - purge C-44
  - Macro argument
    - concatenation operator C-2
    - local label override operator C-4
    - return hex value operator C-4
    - return value operator C-3
  - Memory
    - limit C-27, C-30
    - utilization C-38
  - Memory space C-38, C-41
  - Mode Selection 3-19
  - Motorola
    - DSP linker 2-12
- N**
- Number of Bits to Compare Bits, BCN(3:0) 3-7
- O**
- Object file
    - comment C-13
    - identification C-27
    - symbol C-40, C-50
  - object files 2-1
  - OnCE commands 3-11
  - OnCE/JTAG conversion 3-11
  - operand field 2-3
  - operand fields 2-3
  - Operating Mode, DSP56307 3-7
  - operation field 2-3
  - Option
    - AE C-35, C-37
    - assembler operation C-36
    - CC C-36, C-37
    - CEX C-35, C-37
    - CK C-36, C-37
    - CL C-35, C-37
    - CM C-36, C-37
    - CONST C-36, C-37
    - CONTC C-37
    - CONTCK C-36, C-37
    - CRE C-35, C-37
    - DEX C-36, C-37
    - DLD C-36, C-37
    - DXL C-35, C-37
    - FC C-35, C-37
    - FF C-35, C-37
    - FM C-35, C-37
    - GL C-36, C-38
    - GS C-36, C-38
    - HDR C-35, C-38
    - IC C-36, C-38
    - IL C-35, C-38
    - INTR C-36, C-38
    - LB C-36, C-38
    - LDB C-36, C-38
    - listing format C-35
    - LOC C-35, C-38
    - MC C-35, C-38
    - MD C-35, C-38
    - message C-35
    - MEX C-35, C-38
    - MI C-36, C-38
    - MSW C-35, C-38
    - MU C-35, C-38
    - NL C-35, C-38
    - NOAE C-38
    - NOCC C-38
    - NOCEX C-38
    - NOCK C-39
    - NOCL C-39
    - NOCM C-39
    - NODEX C-39
    - NODLD C-39
    - NODXL C-39
    - NOFC C-39
    - NOFF C-39
    - NOFM C-39
    - NOGS C-39
    - NOHDR C-39
    - NONINTR C-39
    - NOMC C-39
    - NOMD C-39
    - NOMEX C-39
    - NOMI C-39
    - NOMSW C-39
    - NONL C-39
    - NONS C-39
    - NOPP C-39
    - NOPS C-39
    - NORC C-39
    - NORP C-39
    - NOSCL C-39
    - NOU C-39
    - NOUR C-39
    - NOW C-39
    - NS C-36, C-39
    - PP C-35, C-40
    - PS C-36, C-40

PSM C-36  
 RC C-35, C-40  
 reporting C-35  
 RP C-36, C-40  
 RSV C-36  
 S C-35, C-40  
 SCL C-36, C-40  
 SCO C-36, C-40  
 SI C-36  
 SO C-36, C-40  
 SVO C-36  
 symbol C-36  
 U C-35, C-40  
 UR C-35, C-40  
 W C-35, C-40  
 WEX C-40  
 XLL C-36, C-40  
 XR C-36, C-41

## P

P Space Enable Bit, BPEN 3-7  
 Packing Enable Bit, BPAC 3-6  
 PC requirements 1-2  
 PEROM 3-7  
     stand-alone operation 3-7  
 power supply, external 1-2, 1-4  
 program  
     assembling the 2-5  
     example 2-3  
     writing the 2-2  
 Program counter C-6, C-41  
 programming  
     AAR0 3-5  
     assembly 2-1  
     development 2-1  
     example 2-1

## Q

Quick Start Guide 1-1

## R

Read Enable Pin, RD 3-7  
 Reset, DSP56307 3-8  
 RS-232 cable connection 1-4  
 RS-232 interface 3-11  
 RS-232 interface cable 1-2  
 RS-232 serial interface 3-11  
 running the Debugger program 2-19

## S

Sampling frequency 3-8

Section C-47  
     end C-24  
     global C-26, C-38, C-48  
     local C-30, C-48  
     nested C-39  
     static C-38, C-48  
 Serial Clock Pin, SCK0 3-11  
 Serial Communication Interface Port (SCI) 3-14  
 Serial Control Pin 0, SC00 3-11  
 Serial Control Pin 0, SC10 3-11  
 Serial Control Pin 1, SC01 3-11  
 Serial Control Pin 1, SC11 3-11  
 Serial Control Pin 2, SC02 3-11  
 Serial Control Pin 2, SC12 3-11  
 serial interface 3-11  
 Serial Receive Data Pin, SRD0 3-11  
 Serial Transmit Data Pin, STD0 3-11  
 Source file  
     end C-22  
 source statement format 2-2  
 SRAM 3-2, 3-3  
     connections 3-4  
 SRAM memory map 3-5  
 stand-alone operation 3-7  
 Stereo Headphones 3-9  
 Stereo Input 3-9  
 Stereo Output 3-9  
 String  
     concatenation C-6, C-7  
     delimiter C-5  
     packed C-40  
 Symbol  
     case C-38  
     cross-reference C-37  
     equate C-24, C-37  
     global C-38  
     listing C-40  
     set C-26, C-50  
     undefined C-40

## W

Warning C-40  
 Write Enable Pin, WR 3-7

## X

X data transfer field 2-3  
 X Space Enable Bit, BXEN 3-7

## Y

Y data transfer field 2-3  
 Y Space Enable Bit, BYEN 3-7



Quick Start Guide	<b>1</b>
Example Program	<b>2</b>
DSP56321EVM Technical Summary	<b>3</b>
DSP56321EVM Schematics	<b>A</b>
DSP56321EVM Parts List	<b>B</b>
Index	<b>I</b>

<b>1</b>	Quick Start Guide
<b>2</b>	Example Program
<b>3</b>	DSP56321EVM Technical Summary
<b>A</b>	DSP56321EVM Schematics
<b>B</b>	DSP56321EVM Parts List
<b>I</b>	Index