Mu-Law and A-Law Companding Using the TMS320C2xx DSP

APPLICATION REPORT: SPRA349

Semiconductor Sales & Marketing

Digital Signal Processing Solutions
October 1997

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Tables

μ-Law and A-Law Companding Using the TMS320C2xx DSP

Abstract

Companding (compression and expansion) is a method commonly used in telephony applications to increase dynamic range while keeping the number of bits used for quantization constant. The compression is lossy, but provides lower quantization error at small amplitude values than at larger values.

μ-law is a companding standard commonly used in the United States. It takes 14-bit data values and compresses them to eight-bit values. A-law is the standard used in Europe. It takes 13-bit data values and compresses them to eight-bit values. For more information on these standards, see the references at the end of this application note.

In these C-callable subroutines, expanded input and output are expected to be in signed left-aligned form. For a-law, this means the least-significant three bits are zero after expansion, and are ignored during compression. For μ-law, the two least significant bits are zero after expansion, and are ignored during compression.

This application note describes subroutines for performing μ-law and A-law companding using Texas Instruments (TI™) TMS320C2xx ('C2xx) DSP.

Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Related Documentation

The following document contains additional information for use with this application.

- *Digital Signal Processing Applications with the TMS320 Family, Volume 1*, Literature number SPRA012A

Obtaining the Software

This program is available for download as a self-extracting executable at the following location:

ftp://ftp.ti.com/pub/tms320bbs/c2xxfiles/COMPAND.EXE

In these C-callable subroutines, expanded input and output are expected to be in signed left-aligned form. For a-law, this means the least-significant three bits are zero after expansion, and are ignored during compression. For μ-law, the two least significant bits are zero after expansion, and are ignored during compression.

Software Contents

The following tables list the contents of the directories containing the subroutines.

Table 1. Main Directory

File	Description
muexpntb.asm	μ-law expand function using 256-word lookup table.
mucompn.asm	μ-law compress function.
aexpntbl.asm	A-law expand function using 256-word lookup table.
aexpn.asm	A-law expand function using no lookup table.
acompn.asm	A-law compress function.
main.c	Sample C program which tests the included functions.
c203.cmd	Linker command file describing memory map and sections for the C203.
compand.doc	This document (in Microsoft Word format).
compand.htm	This document (in HTML format).
compand.txt	This document (in plain text format).
compand.out	Executable file for sample C program to be loaded by C2xx simulator or hardware.
build.bat	Batch file for building compand.out using only the Texas Instruments development tools.
assembly	Directory containing assembly-only versions of the compander routines and example.
obj	Directory containing object files.

Table 2. Assembly Directory

File	Description
muexpntb.asm	μ-law expand function using 256-word lookup table.
mucompn.asm	μ-law compress function.
aexpntbl.asm	A-law expand function using 256-word lookup table.
aexpn.asm	A-law expand function using no lookup table.
acompn.asm	A-law compress function.
main.asm	Sample assembly program that tests the included functions.
c203.cmd	Linker command file describing memory map and sections for the C203.
compand.out	Executable file for sample assembly program to be loaded by C2xx simulator or hardware.
build.bat	Batch file for building compand.out using only the Texas Instruments development tools.
obj	Directory containing object files.

Usage

The following sections describe the correct usage of these algorithms for μ-law and A-law expansion and compression.

μ-law Expansion

To perform μ-law expansion with table lookup:

```
void muexpN_table(int *data, int N)
```

Where `data` is a pointer to an array of integers where each integer contains the encoded value (in bits 0-7). Each of the bits is inverted during μ-law encoding:

Table 3 Input Bit Values for μ-law Expansion

Bit	*15*	*…*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	0
Value	*--*	*…*	*P*	*S2*	*S1*	*S0*	*Q3*	*Q2*	*Q1*	Q0

which consists of:

P - Polarity bit (sign)
S - 3-bit segment number
Q - 4-bit quantization number
"--" represents don't care

Output (written over input array) is a 14-bit signed number stored left aligned in each integer (in bits 2-15):

Table 4 Output Bit Values for μ-law Expansion

Bit	*15*	*14*	*13*	*12*	*…*	*3*	*2*	*1*	*0*
Value	*S*	*X12*	*X11*	*X10*	*…*	*X1*	*X0*	*--*	*--*

which consists of:

S - Sign bit
X - 13-bit magnitude
"--" represents don't care
where $X = (33 + 2Q)\ X\ 2^{S} - 33$

μ-law Compression

To perform μ-law compression:

```
void mucompN(int *data, int N)
```

Where data is a pointer to an array of 14-bit signed number stored left aligned in each integer (in bits 2-15):

Table 5 Input Bit Values for μ-law Compression

Bit	*15*	*14*	*13*	*12*	*…*	*3*	*2*	*1*	*0*
Value	*S*	*X12*	*X11*	*X10*	*…*	*X1*	*X0*	--	--

which consists of:

S - Sign bit
X - 13-bit magnitude
"--" represents don't care
where $X = (33 + 2Q)\ X\ 2^S - 33$

Output (written over input array) consists of integers where each integer contains the encoded value (in bits 0-7). Each of the bits is inverted during μ-law encoding:

Table 6 Output Bit Values for μ-law Compression

Bit	*15*	*…*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	0
Value	*--*	*…*	*P*	*S2*	*S1*	*S0*	*Q3*	*Q2*	*Q1*	Q0

which consists of:

P - Polarity bit (sign)
S - 3-bit segment number
Q - 4-bit quantization number
"--" represents don't care

A-law Expansion

To perform A-law expansion with table lookup:

```
void aexpN_table(int *data, int N)
```

To perform A-law expansion without table lookup:

```
void aexpN(int *data, int N)
```

where data is a pointer to an array of integers where each integer contains the encoded value (in bits 0-7). The even-numbered bits are inverted during a-law encoding:

Table 7 Input Bit Values for A-Law Expansion

Bit	*15*	*…*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	0
Value	*--*	*…*	*P*	*S2*	*S1*	*S0*	*Q3*	*Q2*	*Q1*	Q0

which consists of:

P - Polarity bit (sign)
S - 3-bit segment number
Q - 4-bit quantization number
"--" represents don't care

Output (written over input array) is a 13-bit signed number stored left aligned in each integer (in bits 2-15):

Table 8 Output Bit Values for A-Law Expansion

Bit	*15*	*14*	*13*	*12*	*…*	*3*	*2*	*1*	*0*
Value	*S*	*X11*	*X10*	*X9*	*…*	*X0*	*--*	*--*	*--*

which consists of:

S - Sign bit
X - 13-bit magnitude
"--" represents don't care

A-law Compression

To perform A-law compression:

```
void acompN(int *data, int N)
```

Where data is a pointer to an array of 13-bit signed numbers stored left aligned in each integer (in bits 2-15):

Table 9 Input Bit Values for A-law Compression

Bit	*15*	*14*	*13*	*12*	*…*	*3*	*2*	*1*	*0*
Value	*S*	*X11*	*X10*	*X9*	*…*	*X0*	*--*	*--*	*--*

which consists of:

S - Sign bit
X - 13-bit magnitude
"--" represents don't care

Output (written over input array) consists of integers where each integer contains the encoded value (in bits 0-7). The even-numbered bits are inverted during a-law encoding:

Table 10 Output Bit Values for A-law Compression

Bit	*15*	*…*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	0
Value	*--*	*…*	*P*	*S2*	*S1*	*S0*	*Q3*	*Q2*	*Q1*	Q0

which consists of:

P - Polarity bit (sign)
S - 3-bit segment number
Q - 4-bit quantization number
"--" represents don't care

Performance

The following tables contain performance information for the C-callable routines.

Storage

Table 11 Decode Table Sizes for Routines

Routine	Decode Table Size (in 16-Bit Program Memory Words)
muexpN_table	256
aexpN_table	256
AexpN	0
MucompN	0
AcompN	0

Timing

These timing benchmarks are for the C-callable routines. The assembly-only versions will be several cycles faster, since there is no stack processing. The assembly-only per-sample timing will be nearly identical to the C-callable routines.

Table 12 Timing with Initialization and C function-call overhead (per 256 samples)

Routine	Clock Cycles	Time (20 MHz)	Time (40 MHz)
muexpN_table	2635	132 [µs]	65.9 [µs]
aexpN_table	2635	132 [µs]	65.9 [µs]
AexpN	12938	647 [µs]	323 [µs]
MucompN	10313	516 [µs]	258 [µs]
AcompN	11956	598 [µs]	299 [µs]

Table 13 Approximate Timing per Sample

Routine	Clock Cycles	Time (20 MHz)	Time (40 MHz)
muexpN_table	10.3	.515 [µs]	.258 [µs]
aexpN_table	10.3	.515 [µs]	.258 [µs]
AexpN	50.5	2.53 [µs]	1.26 [µs]
MucompN	40.3	2.02 [µs]	1.01 [µs]
AcompN	46.7	2.34 [µs]	1.17 [µs]

Error Analysis

The signal-to-quantization noise ratio for μ-law encoding is 39.3 dB for a full-amplitude sine wave (Bellamy). For other amplitude signals, see Figure 3 on page 176 (Lin).

Demonstration Programs

C Program Example

Included in this distribution is a C program that shows how to use the package. To build it using the Go DSP Code Composer, add the following files to the project and build the project as documented in the Go DSP documentation: mucompn.asm, muexpntb.asm, acompn.asm, aexpntbl.asm, aexpn.asm, main.c, and c203.cmd.

To build using only the Texas Instruments Code Generation Tools, type build.bat from a DOS prompt from within the directory where you installed this package.

The demonstration program main.c starts with 8-bit compressed samples ramping up from 0 to 255, expands and then re-compresses them using all of the supplied subroutines. To verify correct program execution using the Go DSP tools, display array data[] in a graph window and single step through the C program. You'll see the ramp function after it's generated, then the expansion and the exponential 16-bit version, and then the compression back to the eight-bit ramp function.

NOTE:

The spike at the 128th array element in the μ-law re-compressed ramp function occurs because this represents the value positive zero. This matches the 255th element's value, which is also a zero. Since both positive and negative zero values are represented in μ-law and A-law encoding, only one value can be chosen for compressing the value zero.

Assembly Program Example

An assembly-only demonstration program is included in the assembly subdirectory. This program performs the same function as the C version of the program. To build it using the Go DSP Code Composer, add the following files to the project and build the project as documented in the Go DSP documentation: mucompn.asm, muexpntb.asm, acompn.asm, aexpntbl.asm, aexpn.asm, main.c, c203.cmd.

To build using only the Texas Instruments Code Generation Tools, type build.bat from a DOS prompt from within the assembly directory below where you installed this package.

Each of the assembly versions of the subroutines uses AR3 for holding the pointer to the data and AR4 for holding the number of data points. See the descriptions in the Usage section for more information about the provided functions.

References

Bellamy, J.C. Digital Telephony. John Wiley & Sons, 1982.

Lin, Kun-Shan et al. Digital Signal Processing Applications with the TMS320 Family. Volume 1. Texas Instruments, 1989. (SPRA012A)

ITU (formerly CCITT). General Aspects of Digital Transmission Systems; Terminal Equipment. Recommendations G.700-G.795. Volume III - Fascicle III.4. Ixth Plenary Assembly, Melbourne, 14-25 November 1988. Geneva, 1989. (ISBN 92-61-03341-5)