Companding Routines for the TMS32010/TMS32020

APPLICATION REPORT: SPRA001

Authors: Lou Panucco and Cole Erskine
Digital Signal Processing – Semiconductor Group

Digital Signal Processing Solutions
1989

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

Companding Routines for the TMS32010/TMS32020

Abstract

This report discusses companding routines. Companding is required for applications that use codec devices, such as in public and private telephone networks. With the speed and versatility of the TMS320, companding can be performed in either software or hardware. The report describes both the A-law and μ-law software companding methods. Programs are also provided to show how the software companding can be performed using the computational power of the TMS32010 and the TMS32020. An example of the hardware companding is presented in the report, *Telecommunications Interfacing to the TMS32010* (SPRA128).

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.

INTRODUCTION

In Pulse Code Modulation (PCM) systems, which are commonly used in public and private (PBX) telephone networks, samples of an analog speech waveform are encoded as binary words and transmitted serially usually at a rate of 8000 samples per second. This digitized data is communicated most efficiently if the amplitude of the waveform is compressed to logarithmic scale before transmission (reducing the number of bits required for their representation), and then expanded at the receiver.

The conversion to logarithmic scale insures that low-amplitude signals are digitized with a minimal loss of fidelity. This procedure of first compressing and then expanding the signal is known as "companding" (COMpressing and exPANDING). Figures 1(a) and 1(b) show the procedures involved in companding, typically accomplished by a hardware device called a codec or combo-codec (the combined PCM codec and filter).[1] Since codecs are inexpensive, they have been widely used as input/output (I/O) devices for analog signals in many digital signal processing applications, such as digital telephony.

In a digital signal processing system that incorporates codecs, a reversed companding process is required as shown in Figure 1(c). The compressed PCM data is first converted to linear PCM to be processed by the digital signal processor. After the digital signal processing, the processed linear PCM is then compressed before sending it to the codec to produce an analog output signal.

The TMS320 family of digital signal processors is designed for numeric-intensive applications. Because of the processor's high speed, the companding (actually an expand-and-compress procedure) described in Figure 1(c) can be performed with minimum execution time and program requirements. This allows the processor to dedicate most of its resources for real-time digital signal processing applications, such as filtering, tone generation/detection, transcoding, vocoding, and echo cancellation. Companding can be performed in software in two ways: (1) by calculating the companding algorithm in real-time, or (2) by looking up a table pregenerated using the algorithm. The lookup-table approach naturally requires more storage, but it provides faster execution than the algorithmic approach. The tradeoff must be made by the digital signal processing designer between memory and speed requirements. In addition, companding can be accomplished externally in hardware (see the application report, "Telecommunications Interfacing to the TMS32010").[3]

The main portion of this report presents four TMS32010 programs that implement the standard companding algorithms. The TMS32010 is the first generation of the TMS320 family of digital signal processors. Three programs for companding, which use the TMS32020, the second-generation digital signal processor, are included in the appendix. One of these programs uses the lookup-table approach. Note that no special effort is taken to further optimize these programs for any particular application. The purpose of this report is to show how companding can be

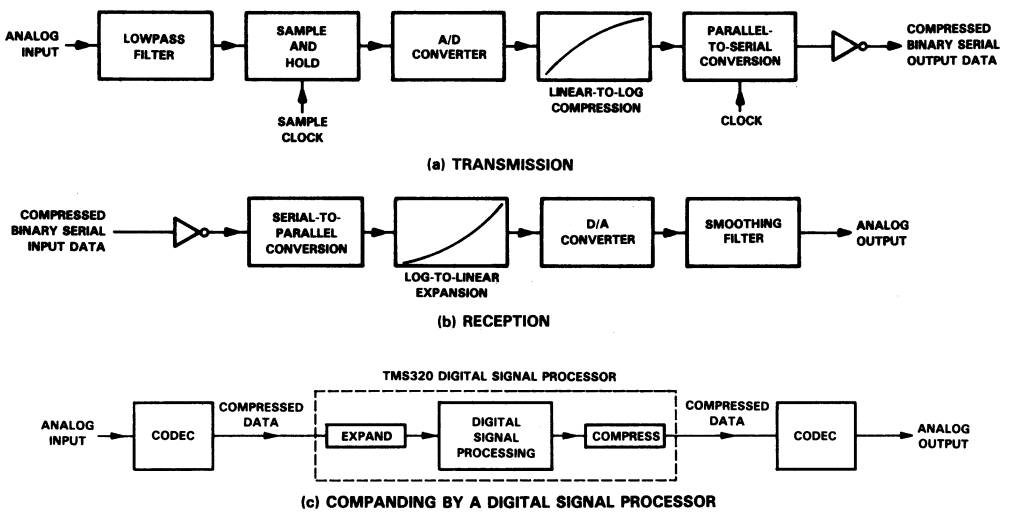


Figure 1. PCM Companding

performed by both generations of digital signal processors. Other application reports are available, which show how the companding routines can be optimized for special applications, such as Adaptive Differential Pulse Code Modulation (ADPCM)[2] and telecommunication interfaces[3] using the TMS32010, and echo cancellation[4] using the TMS32020.

COMPANDING

In any sampled data system, the analog-to-digital (A/D) conversion process introduces quantization noise. For the usual linear A/D encoding scheme, the digitized code word is a truncated binary representation of the analog sample. The effect of this truncation is most pronounced for small signals. For voice transmission, this is undesirable since most information in speech signals resides in the lower amplitudes even though speech signals typically require a wide dynamic range. This can be remedied by adjusting the size of the quantization interval so that it is proportional to the input signal level. In this case, the quantization interval is small for small amplitude signals and larger for larger signals. Consequently, lower amplitudes are represented with more quantization levels and, therefore, with greater resolution.

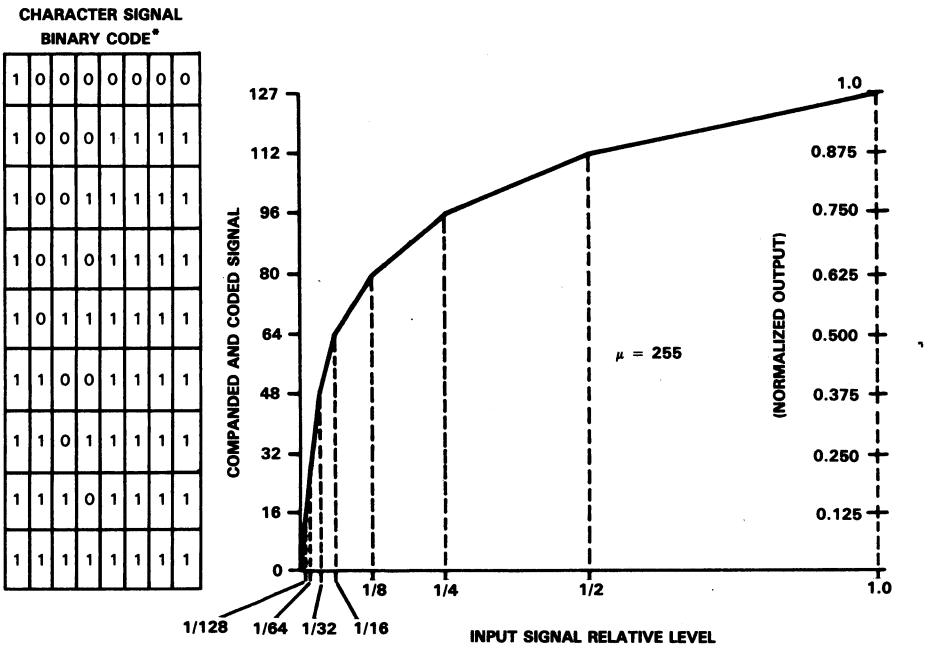
The resulting encoding scheme is logarithmic in nature and has the property of yielding the greatest dynamic range for a given signal-to-noise ratio and word length. Companding is defined by two international standards based on this relation – both compress the equivalent of 13 bits of dynamic range into 7. The standard employed in the United States and Japan is known as the μ-255 law companding characteristic and is given by the equation

$$F(x) = \text{sgn}(x) \frac{\ln(1 + \mu|x|)}{\ln(1 + \mu)}$$

where:

F(x) is the compressed output value
x is the normalized input signal (between −1 and 1)
μ is the compression parameter (= 255 in North America)
sgn(x) is the sign (±) of x

The European standard is referred to as A-law companding and is defined by the equation



* This is the bit pattern transmitted for positive input values. The left-most bit is a 0 for negative input values.

Figure 2. Companding Curve of the μ-Law Compander (from Reference 5)

$$F(x) = \begin{cases} \text{sgn}(x)\ \dfrac{A|x|}{1 + \ln(A)} & \text{for } 0 \le |x| < \dfrac{1}{A} \\ \text{sgn}(x)\ \dfrac{(1 + \ln A|x|)}{1 + \ln(A)} & \text{for } \dfrac{1}{A} \le |x| \le 1 \end{cases}$$

where:

F(x) is the compressed output value
x is the normalized input signal (between −1 and 1)
A is the compression parameter (= 87.6 in Europe)
sgn(x) is the sign (±) of x

In practice, the code word is actually inverted before transmission. Low amplitude signals tend to be more numerous than large amplitude samples. Consequently, inverting the bits increases the density of positive pulses on the transmission line which improves the performance of timing and clock recovery circuits.

μ-255 COMPANDING

Eight-bit sign-magnitude words can represent 255 different code words. This made 255 the most convenient choice for the μ-law companding parameter. This companding characteristic exhibits the valuable property of being closely approximated by a set of eight straight-line segments, as shown in Figure 2.5 This figure illustrates how the input sample values of successively larger intervals are compressed into intervals of uniform size. The slope of each segment is exactly one-half that of the preceding one. The step size between adjacent code words is doubled in each succeeding segment. This property allows the conversion to and from a linear format to be done very efficiently.

TMS32010 Algorithm

Uniformly quantized 14-bit sign-magnitude numbers are compressed into 8-bit signed μ-255 code words by the program 'MULAWCMP' and expanded to their original amplitude by the program 'MULAWEXP', both listed and flowcharted in the Program Listings section. The code word Y, formed by MULAWCMP, has the format Y = PSSSQQQQ composed of:

Polarity bit: P
3-bit segment number: SSS
4-bit quantization bin number: QQQQ

The encoding algorithm is best understood by examining the segment endpoints of Table 1, which begin with the values 31, 95, 223,..., 4063.

Note that

$$\begin{aligned} 31 &= 2^6 - 33 \\ 95 &= 2^7 - 33 \\ 223 &= 2^8 - 33 \\ &\vdots \\ 4063 &= 2^{12} - 33 \end{aligned}$$

so that if 33 is added to each value in the table, the end points become powers of two.

This means that the segment number corresponding to a number N (which is to be encoded) can be determined by finding the most significant '1' bit in the binary representation of N + 33. Furthermore, as Table 2 indicates, the following four bits make up the quantization bin number. The remaining bits are discarded.

On expansion, these lost bits are assumed to have been the median of the possible numbers which these lost bits could have represented — a one followed by zeroes (see Table 3). This rounding limits the loss in accuracy.

Table 1. Encoding/Decoding Table for μ-255 PCM*
(Courtesy of John Wiley & Sons, see Reference 6)

Input Amplitude Range	Step Size	Segment Code S	Quantization Code Q	Code Value	Decoder Amplitude
0-1	1		0000	0	0
1-3			0001	1	2
3-5	2	000	0010	2	4
.			.	.	.
.			.	.	.
.			.	.	.
29-31			1111	15	30
31-35			0000	16	33
.			.	.	.
.	4	001	.	.	.
.			.	.	.
91-95			1111	31	93
95-103			0000	32	99
.			.	.	.
.	8	010	.	.	.
.			.	.	.
215-223			1111	47	219
223-239			0000	48	231
.			.	.	.
.	16	011	.	.	.
.			.	.	.
463-479			1111	63	471
479-511			0000	64	495
.			.	.	.
.	32	100	.	.	.
.			.	.	.
959-991			1111	79	975
991-1055			0000	80	1023
.			.	.	.
.	64	101	.	.	.
.			.	.	.
1951-2015			1111	95	1983
2015-2143			0000	96	2079
.			.	.	.
.	128	110	.	.	.
.			.	.	.
3935-4063			1111	111	3999
4063-4319			0000	112	4191
.			.	.	.
.	256	111	.	.	.
.			.	.	.
7903-8159			1111	127	8031

* This table displays magnitude encoding only. Polarity bits are assigned as "0" for positive and "1" for negative. In transmission, all bits are inverted.

Table 2. μ-255 Binary Encoding Table*

Biased Input Values													Compressed Code Word						
Bit: 12	11	10	9	8	7	6	5	4	3	2	1	0	Bit: 6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	0	0	0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	0	0	1	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	0	1	0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	0	1	1	Q_3	Q_2	Q_1	Q_0
0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	x	1	0	0	Q_3	Q_2	Q_1	Q_0
0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	x	x	1	0	1	Q_3	Q_2	Q_1	Q_0
0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	x	x	x	1	1	0	Q_3	Q_2	Q_1	Q_0
1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	x	x	x	x	1	1	1	Q_3	Q_2	Q_1	Q_0

*The polarity is not shown in this table.
NOTE: The leading bit is the sign bit.
EXAMPLES:
(1) $+865_{10} \xrightarrow{\text{BIAS}} +865_{10}+33_{10} = +898_{10} = +382_{16} = (0)0\ 0011\ 1000\ 0010_2 \rightarrow (0)100\ 1100_2$
(2) $-2513_{10} \xrightarrow{\text{BIAS}} -2513_{10}-33_{10} = -2546_{10} = -9F2_{16} = (1)0\ 1001\ 1111\ 0010_2 \rightarrow (1)110\ 0011_2$

Table 3. μ-255 Binary Decoding Table*

Compressed Code Word							Biased Output Values												
Bit: 6	5	4	3	2	1	0	Bit: 12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	Q_3	Q_2	Q_1	Q_0	0	0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1
0	0	1	Q_3	Q_2	Q_1	Q_0	0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0
0	1	0	Q_3	Q_2	Q_1	Q_0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0
0	1	1	Q_3	Q_2	Q_1	Q_0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0
1	0	0	Q_3	Q_2	Q_1	Q_0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0	0
1	0	1	Q_3	Q_2	Q_1	Q_0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0	0	0
1	1	0	Q_3	Q_2	Q_1	Q_0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0	0	0	0
1	1	1	Q_3	Q_2	Q_1	Q_0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0	0	0	0	0

*The polarity is not shown in this table.
NOTE: The leading bit is the sign bit.
EXAMPLES:
(1) $3C_{16} = (0)011\ 1100_2 \rightarrow (0)0\ 0001\ 1100\ 1000_2 = +01C8_{16} = +456_{10} \xrightarrow{\text{REMOVE BIAS}} 456_{10}-33_{10} = +423_{10}$
(2) $E3_{16} = (1)110\ 0011_2 \rightarrow (1)0\ 1001\ 1100\ 0000_2 = -09C0_{16} = -2496_{10} \xrightarrow{\text{REMOVE BIAS}} -2496_{10}+33_{10} = -2463_{10}$

Performance

Analysis of the PCM μ-255 companding system of Figure 1 shows that the approximated digital values approach the original inputs closely, with a signal-to-quantization noise ratio of 39.3 dB for a full-range sinusoid.[6] In general, voice signals have smaller quantization errors but lower signal power, so μ-255 performance in voice transmission is approximately the same. The signal-to-quantization noise ratio for μ-255 law encoding is given in Figure 3 for sinusoid inputs. The algorithm space and time requirements are given in Table 4.

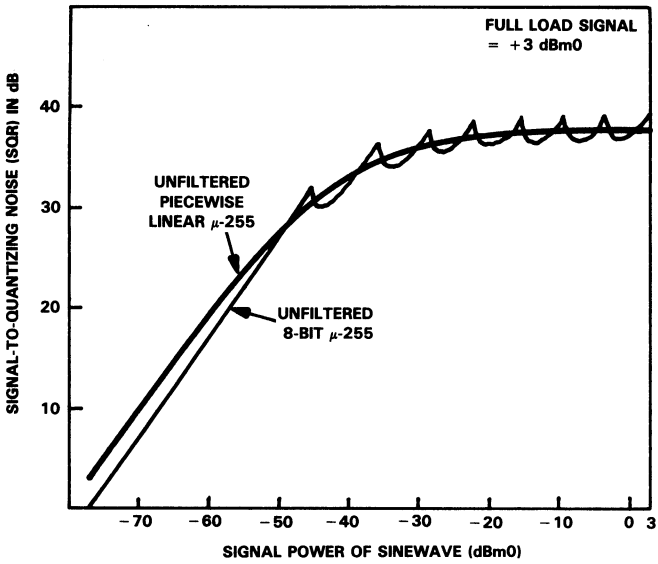


Figure 3. Signal-to-Quantizing Noise of μ-Law Coding with Sinewave Inputs (Courtesy of John Wiley & Sons, see Reference 6)

Table 4. Summary of μ-Law Program Space and Time Requirements

Function	Words of Memory		Program Cycles		Time Required†
	Program	Data	Initialization	Loop‡	μsec
Compress	105	13	17	40	8.0
Expand	46	8	6	23	4.6

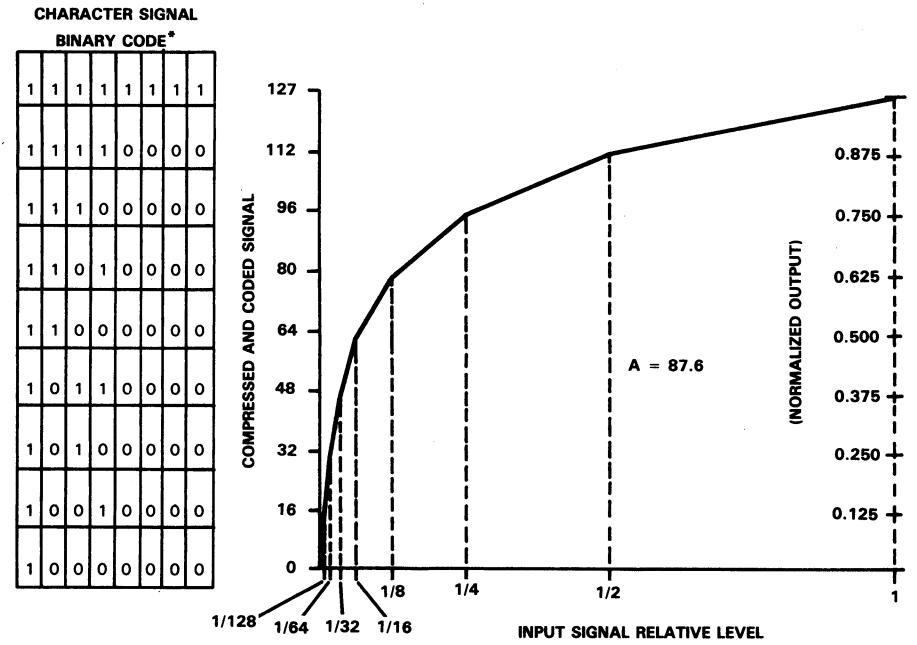
† Assuming initialization
‡ Worst case

A-LAW COMPANDING

The companding characteristic recommended by CCITT and adopted in Europe is the 'A-law' standard. Not only can this characteristic be approximated with linear segments, as with the μ-law approximation, but a portion of the rule is linear by definition.

The A-law programs 'ALAWCOMP' and 'ALAWEXP' are listed and flowcharted in the Program Listings section. They differ from the μ-law routines in the handling of the first segment. This segment is defined to be exactly linear for the A-law. Also, biasing is not required before conversion. The inputs should be scaled to a maximum value of 4096 for representation, as opposed to 8158 for the μ-law case. Since this allows for a minimum step less refined than the μ-law, the A-law characteristic provides less fidelity for small signals but is superior in terms of dynamic range.

Figures 4 and 5 and Tables 5, 6, and 7 presented below are analogous to those given above for μ-law.



* For positive input values. The left-most bit is a 0 for negative input values. Even bits (beginning with 1 at the left) are inverted before transmission.

Figure 4. Companding Curve of the A-Law Compander (from Reference 5)

Table 5. Segmented A-Law Encoding/Decoding Table
(Courtesy of John Wiley & Sons, see Reference 6)

Input Amplitude Range	Step Size	Segment Code S	Quantization Code Q	Code Value	Decoder Amplitude
0-2			0000	0	1
2-4		000	0001	1	3
.			.	.	.
.			.	.	.
.			.	.	.
30-32	2		1111	15	31
32-34			0000	16	33
.			.	.	.
.		001	.	.	.
.			.	.	.
62-64			1111	31	63
64-68			0000	32	66
.			.	.	.
.	4	010	.	.	.
.			.	.	.
124-128			1111	47	126
128-136			0000	48	132
.			.	.	.
.	8	011	.	.	.
.			.	.	.
248-256			1111	63	252
256-272			0000	64	264
.			.	.	.
.	16	100	.	.	.
.			.	.	.
496-512			1111	79	504
512-544			0000	80	528
.			.	.	.
.	32	101	.	.	.
.			.	.	.
992-1024			1111	95	1008
1024-1088			0000	96	1056
.			.	.	.
.	64	110	.	.	.
.			.	.	.
1984-2048			1111	111	2016
2048-2176			0000	112	2112
.			.	.	.
.	128	111	.	.	.
.			.	.	.
3968-4096			1111	127	4032

Table 6. A-Law Binary Encoding Table*

Input Values												Compressed Code Word						
Bit: 11	10	9	8	7	6	5	4	3	2	1	0	Bit: 6	5	4	3	2	1	0
0	0	0	0	0	0	0	Q_3	Q_2	Q_1	Q_0	x	0	0	0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	0	0	1	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	0	1	0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	0	1	1	Q_3	Q_2	Q_1	Q_0
0	0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	1	0	0	Q_3	Q_2	Q_1	Q_0
0	0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	x	1	0	1	Q_3	Q_2	Q_1	Q_0
0	1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	x	x	1	1	0	Q_3	Q_2	Q_1	Q_0
1	Q_3	Q_2	Q_1	Q_0	x	x	x	x	x	x	x	1	1	1	Q_3	Q_2	Q_1	Q_0

*The polarity is not shown in this table.
NOTE: The leading bit is the sign bit.
EXAMPLES:
(1) $+3221_{10} = +C95_{16} = (0)\ 1100\ 1001\ 0101_2 \rightarrow (0)\ 111\ 1001_2$
(2) $-199_{10} = -C7_{16} = (1)\ 0000\ 1100\ 0111_2 \rightarrow (1)\ 011\ 1000_2$

Table 7. A-Law Binary Decoding Table*

Compressed Code Word							Output Values											
Bit: 6	5	4	3	2	1	0	Bit: 11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	Q_3	Q_2	Q_1	Q_0	0	0	0	0	0	0	0	Q_3	Q_2	Q_1	Q_0	1
0	0	1	Q_3	Q_2	Q_1	Q_0	0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1
0	1	0	Q_3	Q_2	Q_1	Q_0	0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0
0	1	1	Q_3	Q_2	Q_1	Q_0	0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0
1	0	0	Q_3	Q_2	Q_1	Q_0	0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0
1	0	1	Q_3	Q_2	Q_1	Q_0	0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0	0
1	1	0	Q_3	Q_2	Q_1	Q_0	0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0	0	0
1	1	1	Q_3	Q_2	Q_1	Q_0	1	Q_3	Q_2	Q_1	Q_0	1	0	0	0	0	0	0

*The polarity is not shown in this table.
NOTE: The leading bit is the sign bit.
EXAMPLES:
(1) $(0)\ 001\ 1101_2 \rightarrow (0)\ 0000\ 0011\ 1011_2 = +3B_{16} = +59_{10}$
(2) $(1)\ 110\ 0100_2 \rightarrow (1)\ 0101\ 0010\ 0000_2 \rightarrow -5201_{6} = -1312_{10}$

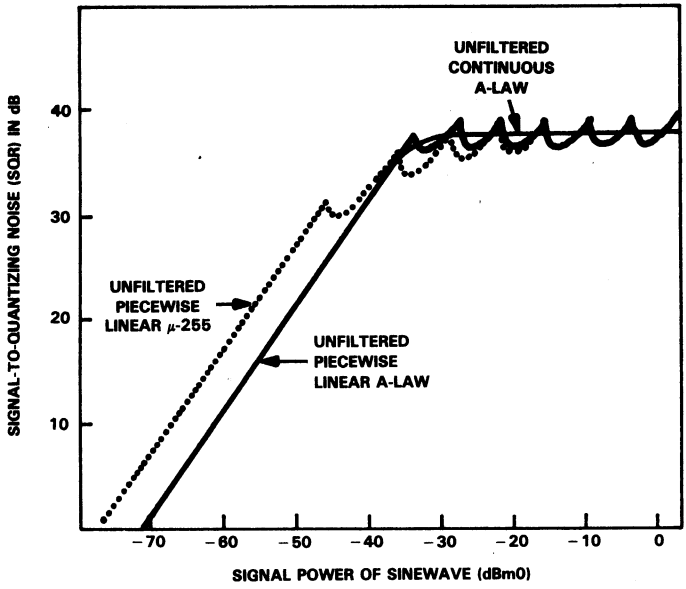


Figure 5. Signal-to-Quantizing Noise of A-Law PCM Coding with Sinewave Inputs (Courtesy of John Wiley & Sons, see Reference 6)

Table 8. Summary of A-Law Program Space and Time Requirements

Function	Words of Memory		Program Cycles		Time Required†
	Program	Data	Initialization	Loop‡	μsec
Compress	97	11	14	36	7.2
Expand	48	7	4	25	5.0

† Assuming initialization
‡ Worst case

SUMMARY

The programs, listed in the next section, have been designed to reduce both memory space used and "loop time," i.e., that time required to complete the calculations after initializations have been made. Of course, other space/time tradeoffs are possible. Dedicated to companding, the TMS32010 can compress 125,000 or expand 200,000 words in a second using these routines. This speed and the versatility of the TMS32010 allow one device to compand a PCM data stream while simultaneously performing related functions such as filtering, vocoding, and tone generation/recognition.

PROGRAM LISTINGS

The following TMS32010 program flowcharts and assembly language routines are listed below:

'MULAWCMP': μ-LAW COMPRESSION
'MULAWEXP': μ-LAW EXPANSION
'ALAWCOMP': A-LAW COMPRESSION
'ALAWEXP': A-LAW EXPANSION

FLOWCHART: MULAWCMP

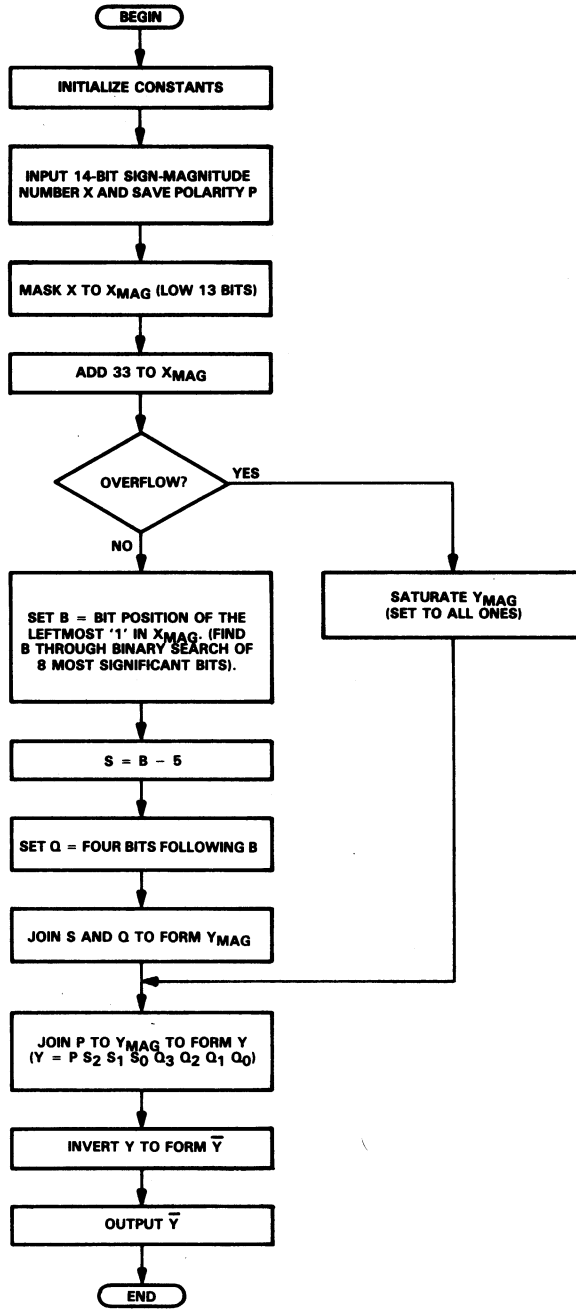


Figure 6. μ-Law Compression

```
0001                    IDT     'MULAWCMP'
0002                ***
0003                *       'MULAWCMP' PERFORMS A MU-255 COMPRESSION. THE
0004                *       14-BIT SIGN-MAGNITUDE INPUT X,
0005                *
0006                *       X = P X12 X11 ... X2 X1 X0
0007                *
0008                *       IS ENCODED AS AN 8-BIT SIGN-MAGNITUDE NUMBER Y,
0009                *
0010                *       Y = P S2 S1 S0 Q3 Q2 Q1 Q0 consisting of
0011                *
0012                *       POLARITY BIT: P,
0013                *       3-BIT SEGMENT NUMBER: S = S2 S1 S0
0014                *       4-BIT QUANTIZATION BIN NUMBER: Q = Q3 Q2 Q1 Q0
0015                *
0016                *       Y IS INVERTED BEFORE TRANSMISSION.
0017                *       PORT 0 IS USED FOR I/O.
0018                *
0019                *       WORST-CASE TIMING IN CYCLES: 17 INIT / 40 LOOP
0020                *       SPACE REQUIREMENTS IN WORDS: 13 DATA / 105 PROGRAM
0021                *
0022                *       CONSTANTS:
0023                *
0024                *
0025      0001      ONE     EQU     1       =1
0026      0002      BIT4    EQU     2       =>0010  (ONE IN BIT 4)
0027      0003      BIT13   EQU     3       =>2000  (ONE IN BIT 13)
0028      0004      MASK13  EQU     4       =>1FFF  (13 ONES)
0029      0005      MASK8   EQU     5       =>00FF  (8 ONES)
0030      0006      MASK4   EQU     6       =>000F  (4 ONES)
0031      0007      MASK2   EQU     7       =>0003  (2 ONES)
0032      0008      BIAS    EQU     8       =33
0033                *
0034                *
0035                *       VARIABLES:
0036                *
0037      0009      X       EQU     9       DATA INPUT (14 BITS)
0038      000A      Y       EQU     10      ENCODED DATA OUTPUT (8 BITS)
0039      000B      P       EQU     11      POLARITY OF DATA (0 FOR POS)
0040      000C      S       EQU     12      3-BIT SEGMENT NUMBER
0041      000D      Q       EQU     13      4-BIT QUANTIZATION BIN NUMBER
0042                *
0043 0000                 AORG    0
0044                *
0045 0000 7E01    INIT    LACK    1
0046 0001 5001            SACL    ONE
0047 0002 2401            LAC     ONE,4
0048 0003 5002            SACL    BIT4
0049 0004 2D01            LAC     ONE,13
0050 0005 5003            SACL    BIT13
0051 0006 1001            SUB     ONE
0052 0007 5004            SACL    MASK13
0053 0008 7EFF            LACK    >00FF
0054 0009 5005            SACL    MASK8
```

```
0055 000A 7E0F          LACK    >000F
0056 000B 5006          SACL    MASK4
0057 000C 7E03          LACK    >0003
0058 000D 5007          SACL    MASK2
0059 000E 7E21          LACK    33
0060 000F 5008          SACL    BIAS
0061              *
0062              *     GET INPUT, SAVE POLARITY, AND MASK TO MAGITUDE
0063              *
0064 0010 4009  START   IN      X,0     INPUT DATA
0065 0011 2003          LAC     BIT13   POLARITY BIT MASK
0066 0012 7909          AND     X
0067 0013 5C0B          SACH    P,4     0 FOR POS; 2 FOR NEG.
0068 0014 2009          LAC     X
0069 0015 7904          AND MASK13
0070              *
0071              *     BIAS MAGNITUDE AND SATURATE IF OVERFLOW OCCURS
0072              *
0073 0016 0008          ADD     BIAS    BIAS INPUT X BY 33
0074 0017 5009          SACL    X
0075 0018 7903          AND     BIT13   CHECK FOR OVERFLOW INTO P
0076 0019 FF00          BZ      LMOST   NO OVERFLOW IF ZERO
     001A 001E
0077              *
0078              *     SATURATION: ENCODE LARGEST CODE WORD
0079              *
0080 001B 7E7F          LACK    >7F     7 ONES
0081 001C F900          B       SIGN
     001D 0063
0082              *
0083              *      COMPUTE THE THREE BITS S(= S2 S1 S0) OF THE
0084              *      COMPRESSED WORD
0085              *
0086              *              S = BP - 5
0087              *
0088              *     WHERE BP IS THE BIT POSITION OF THE LEFTMOST '1'
0089              *     IN X.  THE FOUR FOLLOWING BITS ARE IN THE HIGH
0090              *     HALF OF THE ACCUMULATOR. THESE FOUR BITS ARE
0091              *      Q (=Q3 Q2 Q1 Q0).
0092              *
0093              *     SEARCH BITS X12 THRU X5. (SEE TABLE 1 OF TEXT.)
0094              *
0095              *
0096 001E 2906  LMOST  LAC     MASK4,9         1111 0000
0097 001F 7909          AND     X
0098 0020 FF00          BZ      EEE
     0021 0042
0099 0022 2B07          LAC     MASK2,11        1100 0000
0100 0023 7909          AND     X
0101 0024 FF00          BZ      CC
     0025 0034
0102 0026 2C01          LAC     ONE,12          1000 0000
0103 0027 7909          AND     X
```

```
0104 0028 FF00          BZ      B
     0029 002F
0105 002A 7E07          LACK    7               1... ....
0106 002B 500C          SACL    S
0107 002C 2809          LAC     X,8
0108 002D F900          B       XDONE
     002E 005F
0109 002F 7E06  B       LACK    6               01.. ....
0110 0030 500C          SACL    S
0111 0031 2909          LAC     X,9
0112 0032 F900          B       XDONE
     0033 005F
0113 0034 2A01  CC      LAC     ONE,10          0100 0000
0114 0035 7909          AND     X
0115 0036 FF00          BZ      D
     0037 003D
0116 0038 7E05          LACK    5               001. ....
0117 0039 500C          SACL    S
0118 003A 2A09          LAC     X,10
0119 003B F900          B       XDONE
     003C 005F
0120 003D 7E04  D       LACK    4               0001 ....
0121 003E 500C          SACL    S
0122 003F 2B09          LAC     X,11
0123 0040 F900          B       XDONE
     0041 005F
0124 0042 2707  EEE     LAC     MASK2,7         0000 1100
0125 0043 7909          AND     X
0126 0044 FF00          BZ      GG
     0045 0054
0127 0046 2801          LAC     ONE,8           0000 1000
0128 0047 7909          AND     X
0129 0048 FF00          BZ      F
     0049 004F
0130            *
0131 004A 7E03          LACK    3               0000 1...
0132 004B 500C          SACL    S
0133 004C 2C09          LAC     X,12
0134 004D F900          B       XDONE
     004E 005F
0135            *
0136 004F 7E02  F       LACK    2               0000 01..
0137 0050 500C          SACL    S
0138 0051 2D09          LAC     X,13
0139 0052 F900          B       XDONE
     0053 005F
0140            *
0141 0054 2601  GG      LAC     ONE,6   0000 0010
0142 0055 7909          AND     X
0143 0056 FF00          BZ      H
     0057 005D
0144            *
0145 0058 7E01          LACK    1       0000 001.
0146 0059 500C          SACL    S
```

```
 0147 005A 2E09          LAC     X,14
 0148 005B F900          B       XDONE
      005C 005F
 0149                *
 0150 005D 500C  H       SACL    S       0000 0001 (ACC = 0)
 0151 005E 2F09          LAC     X,15
 0152                *
 0153                * REMOVE LEFTMOST '1' AND STORE Q
 0154                *
 0155 005F 6202  XDONE   SUBH    BIT4
 0156 0060 580D          SACH    Q
 0157                *
 0158                * FORM 8-BIT COMPRESSED WORD FROM Q, S, AND P.
 0159                *
 0160 0061 200D          LAC     Q       Q: BITS 0-3      QQQQ
 0161 0062 040C          ADD     S,4     S: BITS 4-6     SSSQQQQ
 0162 0063 060B  SIGN    ADD     P,6     P: BIT 7       PSSSQQQQ
 0163                *
 0164                * COMPLEMENT FOR TRANSMISSION AND OUTPUT
 0165                *
 0166 0064 7805          XOR     MASK8
 0167 0065 500A          SACL    Y
 0168 0066 480A          OUT     Y,0     PORT 0
 0169 0067 F900  FIN     B       FIN
      0068 0067
 0170                *
 0171                    END
NO ERRORS, NO WARNINGS
```

FLOWCHART: MULAWEXP

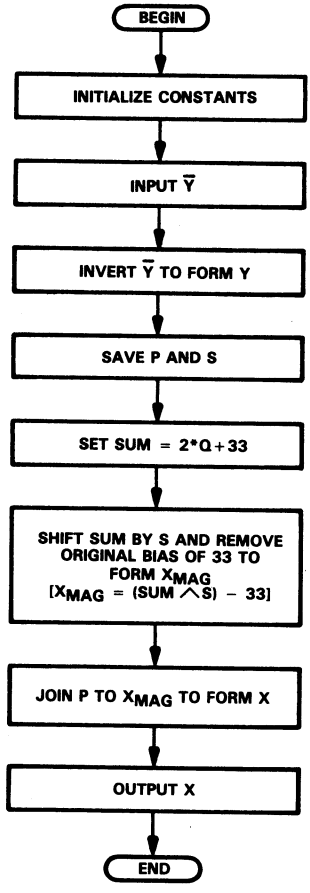


Figure 7. μ-Law Expansion

```
0001                    IDT     'MULAWEXP'
0002              ***
0003              *     'MULAWEXP' PERFORM A MU-LAW EXPANSION. THE
0004              *      8-BIT DATA INPUT IS
0005              *
0006              *      Y = P S2 S1 S0 Q3 Q2 Q1 Q0  WHICH CONSISTS OF
0007              *
0008              *       POLARITY BIT: P
0009              *       3-BIT SEGMENT NUMBER: S = S2 S1 S0
0010              *       4-BIT QUANTIZATION NUMBER: Q = Q3 Q2 Q1 Q0
0011              *
0012              *     THE INPUT Y IS EXPANDED INTO A 14-BIT OUTPUT
0013              *
0014              *      X = P X12 X11 X10 ... X2 X1 X0 CONSISTING OF
0015              *
0016              *       POLARITY BIT:P
0017              *       AND A 13-BIT MAGNITUDE
0018              *                                        S
0019              *       (X12...X0) = (33 + 2Q) X 2  - 33
0020              *
0021              *     PORT 0 IS USED FOR I/O.
0022              *     WORST-CASE TIMING IN CYCLES: 6 INIT / 23 LOOP
0023              *     SPACE REQUIREMENTS IN WORDS: 8 DATA / 46 PROGRAM
0024              *
0025              * CONSTANTS:
0026              *
0027       0001   ONE     EQU     1       = 1
0028       0002   BIT7    EQU     2       = >0080
0029       0003   BIAS    EQU     3       = 33
0030              *
0031              * VARIABLES:
0032              *
0033       0004   Y       EQU     4       MU-LAW COMPRESSED 8-BIT DATA INPUT
0034       0005   X       EQU     5       DECODED (EXPANDED) 14-BIT OUTPUT
0035       0006   P       EQU     6       POLARITY OF DATA (0 FOR POS)
0036       0007   S       EQU     7       3-BIT SEGMENT NUMBER
0037       0008   SUM     EQU     8       VALUE TO BE SHIFTED
0038              *
0039 0000             AORG    0
0040              *
0041 0000 7E01  INIT  LACK    1
0042 0001 5001        SACL    ONE
0043 0002 2701        LAC     ONE,7
0044 0003 5002        SACL    BIT7
0045 0004 7E21        LACK    33
0046 0005 5003        SACL    BIAS
0047              *
0048              * INVERT INPUT
0049              *
0050 0006 4004  START IN      Y,0
0051 0007 7EFF        LACK    >00FF
0052 0008 7804        XOR     Y
0053 0009 5004        SACL    Y
0054              *
```

```
0055                    * SAVE POLARITY AND STRIP TO LOW 7 BITS
0056 000A 7902                  AND     BIT7
0057 000B 5006                  SACL    P       0000 FOR POS; 0080 FOR NEG
0058 000C 7E7F                  LACK    >007F
0059 000D 7904                  AND     Y
0060 000E 5004                  SACL    Y
0061                    *
0062                    * MAGNITUDE IS CORRECT. STRIP Y OF S AND Q.
0063 000F 2C04                  LAC     Y,12    SHIFT S INTO HIGH HALF OF ACC
0064 0010 5807                  SACH    S
0065 0011 2104                  LAC     Y,1     DOUBLE
0066 0012 1507                  SUB     S,5     REMOVE S BITS (STRIP TO 2Q)
0067 0013 0003                  ADD     BIAS
0068 0014 5008                  SACL    SUM     SUM = 2Q + BIAS
0069                    *
0070                    * SHIFT SUM BY S AND REMOVE BIAS
0071 0015 7E1E                  LACK    SBASE   OFFSET FOR SHIFT ROUTINE
0072 0016 0107                  ADD     S,1     DOUBLE S (2 WORDS/SHIFT SEGMENT)
0073 0017 7F8C                  CALA            SHIFT SUM BY S
0074 0018 1003                  SUB     BIAS
0075                    *
0076                    * ACC = MAGNITUDE, ADD POLARITY TO BIT 13
0077 0019 0606                  ADD     P,6     SHIFT P TO BIT 13
0078 001A 5005                  SACL    X
0079 001B 4805                  OUT     X,0     OUTPUT RESULT TO PORT 0
0080 001C F900          FIN     B       FIN
     001D 001C
0081                    *
0082                    *
0083                    * LOAD SUM SHIFTED 0:7
0084 001E 2008          SBASE   LAC     SUM,0
0085 001F 7F8D                  RET
0086 0020 2108                  LAC     SUM,1
0087 0021 7F8D                  RET
0088 0022 2208                  LAC     SUM,2
0089 0023 7F8D                  RET
0090 0024 2308                  LAC     SUM,3
0091 0025 7F8D                  RET
0092 0026 2408                  LAC     SUM,4
0093 0027 7F8D                  RET
0094 0028 2508                  LAC     SUM,5
0095 0029 7F8D                  RET
0096 002A 2608                  LAC     SUM,6
0097 002B 7F8D                  RET
0098 002C 2708                  LAC     SUM,7
0099 002D 7F8D                  RET
0100                    *
0101                            END
NO ERRORS, NO WARNINGS
```

FLOWCHART: ALAWCOMP

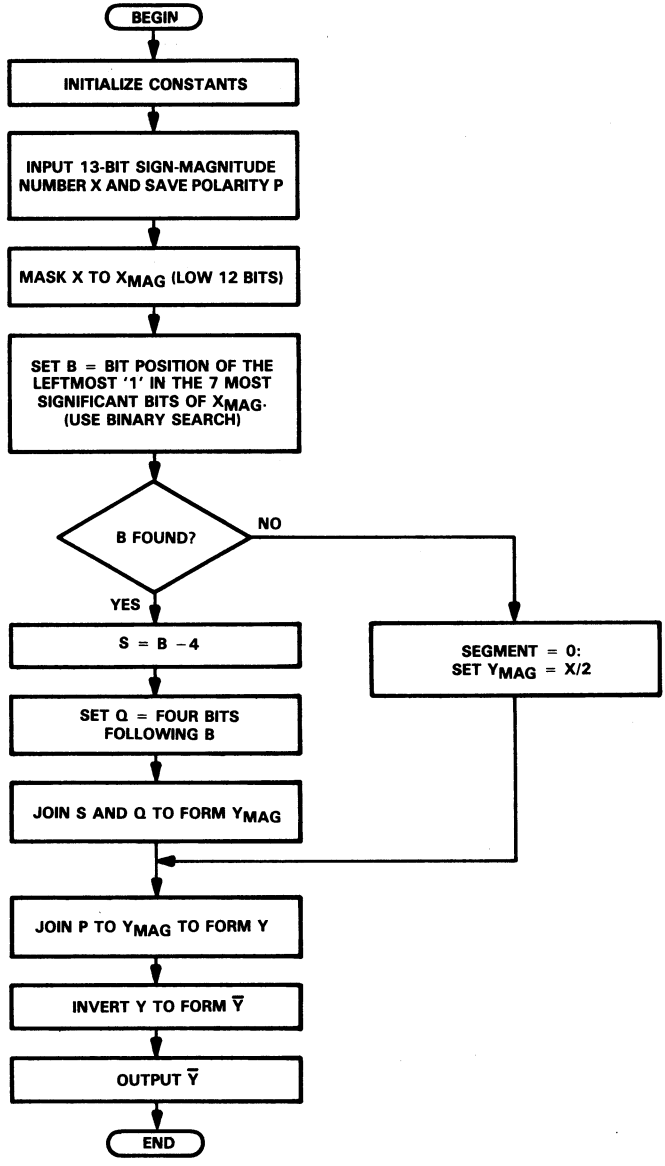


Figure 8. A-Law Compression

```
0001                    IDT     'ALAWCOMP'
0002            ***
0003            *       'ALAWCOMP' PERFORMS AN A-LAW COMPRESSION.
0004            *        THE 13-BIT SIGN-MAGNITUDE INPUT X,
0005            *
0006            *       X = P X11 X10 ... X2 X1 X0
0007            *
0008            *       IS ENCODED AS AN 8-BIT SIGN-MAGNITUDE NUMBER Y,
0009            *
0010            *       Y = P S2 S1 S0 Q3 Q2 Q1 Q0 consisting of
0011            *
0012            *       POLARITY BIT: P,
0013            *       3-BIT SEGMENT NUMBER: S = S2 S1 S0
0014            *       4-BIT QUANTIZATION BIN NUMBER: Q = Q3 Q2 Q1 Q0
0015            *
0016            *       Y IS INVERTED BEFORE TRANSMISSION.
0017            *       PORT 0 IS USED FOR I/O.
0018            *
0019            *       WORST-CASE TIMING IN CYCLES: 14 INIT / 36 LOOP
0020            *       SPACE REQUIREMENTS IN WORDS: 11 DATA / 97 PROG
0021            *
0022            *       CONSTANTS:
0023            *
0024            *
0025      0001  ONE     EQU     1       =1
0026      0002  BIT4    EQU     2       = >0010 (ONE IN BIT 4)
0027      0003  MASK12  EQU     3       = >0FFF (12 ONES)
0028      0004  MASK8   EQU     4       = >00FF ( 8 ONES)
0029      0005  MASK4   EQU     5       = >000F ( 4 ONES)
0030      0006  MASK2   EQU     6       = >0003 ( 2 ONES)
0031            *
0032            *VARIABLES:
0033      0007  X       EQU     7       DATA INPUT (13 BITS)
0034      0008  Y       EQU     8       ENCODED DATA OUTPUT (8 BITS)
0035      0009  P       EQU     9       POLARITY OF DATA (0 FOR POS)
0036      000A  S       EQU     10      3-BIT SEGMENT NUMBER
0037      000B  Q       EQU     11      4-BIT QUANTIZATION BIN NUMBER
0038 0000
0039            *
0040            *
0041 0000               AORG    0
0042            *
0043 0000 7E01  INIT    LACK    1
0044 0001 5001          SACL    ONE
0045 0002 2401          LAC     ONE,4
0046 0003 5002          SACL    BIT4
0047 0004 2C01          LAC     ONE,12
0048 0005 1001          SUB     ONE
0049 0006 5003          SACL    MASK12
0050 0007 7EFF          LACK    >00FF
0051 0008 5004          SACL    MASK8
0052 0009 7E0F          LACK    >000F
0053 000A 5005          SACL    MASK4
0054 000B 7E03          LACK    >0003
```

```
0055 000C 5006          SACL    MASK2
0056                    *
0057                    * GET INPUT AND SAVE POLARITY
0058 000D 4007  START    IN      X,0     INPUT DATA THRU PORT 0
0059 000E 2C01          LAC     ONE,12  POLARITY BIT MASK
0060 000F 7907          AND     X
0061 0010 5C09          SACH    P,4     0 FOR POS; 1 FOR NEG.
0062                    *
0063                    * STRIP TO LOW 12 BITS
0064 0011 2007          LAC     X
0065 0012 7903          AND     MASK12
0066 0013 5007          SACL    X
0067                    *
0068                    * S =BP -4 WHERE BP = BIT POSITION OF THE LEFTMOST '1'
0069                    * IN X. TIND THE '1' THROUGH A BINARY SEARCH OF 8 MSB'S
0070                    * OF X. STORE S AND LOAD X SHIFTED LEFT BY 16-S SO
0071                    * THAT THE '1' AND FOUR FOLLOWING BITS ARE IN THE HIGH
0072                    * HALF OF THE ACCUMULATOR. SEARCH BITS 4 THRU 11.
0073                    *
0074 0014 2805  LMOST    LAC     MASK4,8
0075 0015 7907          AND     X
0076 0016 FF00          BZ      EEE
     0017 0038
0077                    *
0078 0018 2A06          LAC     MASK2,10        1100 0000
0079 0019 7907          AND     X
0080 001A FF00          BZ      CC
     001B 002A
0081                    *
0082 001C 2B01          LAC     ONE,11          1000 0000
0083 001D 7907          AND     X
0084 001E FF00          BZ      B
     001F 0025
0085                    *
0086 0020 7E07          LACK    7               1... ....
0087 0021 500A          SACL    S
0088 0022 2907          LAC     X,9
0089 0023 F900          B       XDONE
     0024 0058
0090                    *
0091 0025 7E06  B       LACK    6               01.. ....
0092 0026 500A          SACL    S
0093 0027 2A07          LAC     X,10
0094 0028 F900          B       XDONE
     0029 0058
0095                    *
0096 002A 2901  CC      LAC     ONE,9           0010 0000
0097 002B 7907          AND     X
0098 002C FF00          BZ      D
     002D 0033
0099                    *
0100 002E 7E05          LACK    5               001. ....
0101 002F 500A          SACL    S
0102 0030 2B07          LAC     X,11
```

```
0103 0031 F900          B       XDONE
     0032 0058
0104                 *
0105 0033 7E04  D       LACK    4               0001 ....
0106 0034 500A          SACL    S
0107 0035 2C07          LAC     X,12
0108 0036 F900          B       XDONE
     0037 0058
0109                 *
0110 0038 2606  EEE     LAC     MASK2,6         0000 1100
0111 0039 7907          AND     X
0112 003A FF00          BZ      GG
     003B 004A
0113                 *
0114 003C 2701          LAC     ONE,7           0000 100
0115 003D 7907          AND     X
0116 003E FF00          BZ      F
     003F 0045
0117                 *
0118 0040 7E03          LACK    3               0000 1...
0119 0041 500A          SACL    S
0120 0042 2D07          LAC     X,13
0121 0043 F900          B       XDONE
     0044 0058
0122                 *
0123 0045 7E02  F       LACK    2               0000 01..
0124 0046 500A          SACL    S
0125 0047 2E07          LAC     X,14
0126 0048 F900          B       XDONE
     0049 0058
0127                 *
0128 004A 2501  GG      LAC     ONE,5           0000 0010
0129 004B 7907          AND     X
0130 004C FF00          BZ      SEGZ
     004D 0053
0131                 *
0132 004E 7E01          LACK    1               0000 001.
0133 004F 500A          SACL    S
0134 0050 2F07          LAC     X,15
0135 0051 F900          B       XDONE
     0052 0058
0136                 * SEGMENT 0: _SSSQQQQ =X/2
0137 0053 2F07  SEGZ    LAC     X,15
0138 0054 5807          SACH    X
0139 0055 2007          LAC     X
0140 0056 F900          B       SIGN
     0057 005C
0141                 *
0142                 * REMOVE LEFTMOST '1' AND STORE Q
0143 0058 6202  XDONE   SUBH    BIT4
0144 0059 580B          SACH    Q
0145                 *
0146                 * FORM 8-BIT COMPRESSED WORS FIR Q, S, AND P.
0147 005A 200B          LAC     Q       Q:BITS 0-3  ____QQQQ
```

```
0148 005B 040A          ADD     S,4     S:BITS 4-6  _SSSQQQQ
0149 005C 0709  SIGN    ADD     P,7     PSSSQQQQ
0150            *
0151            * COMPLEMENT FOR TRANSMISSION AND OUTPUT
0152 005D 7804          XOR     MASK8
0153 005E 5008          SACL    Y
0154 005F 4808          OUT     Y,0     PORT 0
0155 0060 F900  FIN     B       FIN
     0061 0060
0156            *
0157                    END
NO ERRORS, NO WARNINGS
```

FLOW CHART: ALAWEXP

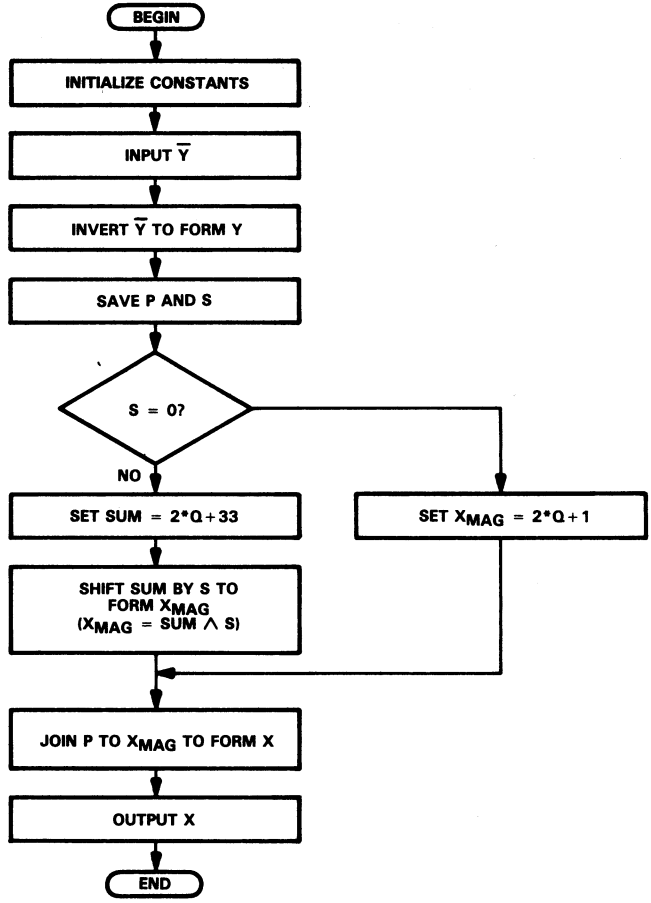


Figure 9. A-Law Expansion

```
0001                    IDT     'ALAWEXP'
0002                ***
0003                *       'ALAWEXP' PERFORM AN A-LAW EXPANSION. THE 8-BIT
0004                *        DATA INPUT IS
0005                *
0006                *        Y = P S2 S1 S0 Q3 Q2 Q1 Q0   WHICH CONSISTS OF
0007                *
0008                *          POLARITY BIT: P
0009                *          3-BIT SEGMENT NUMBER: S = S2 S1 S0
0010                *          4-BIT QUANTIZATION NUMBER: Q = Q3 Q2 Q1 Q0
0011                *
0012                *       THE INPUT Y IS EXPANDED INTO A 13-BIT OUTPUT
0013                *
0014                *        X = P X11 X10 X9 ... X2 X1 X0 CONSISTING OF
0015                *
0016                *          POLARITY BIT:P
0017                *          AND A 13-BIT MAGNITUDE (X12...X0)
0018                *
0019                *       PORT 0 IS USED FOR I/O.
0020                *       WORST-CASE TIMING IN CYCLES: 4 INIT / 25 LOOP
0021                *       SPACE REQUIREMENTS IN WORDS: 7 DATA / 48 LOOP
0022                *
0023                * CONSTANTS:
0024                *
0025      0001      ONE     EQU     1       = 1
0026      0002      BIT7    EQU     2       = >0080 (ONE IN BIT 7)
0027                *
0028                * VARIABLES:
0029                *
0030      0003      Y       EQU     3       A-LAW COMPRESSED 8-BIT DATA INPUT
0031      0004      X       EQU     4       DECODED (EXPANDED) 13-BIT OUTPUT
0032      0005      P       EQU     5       POLARITY OF DATA (0 FOR POS)
0033      0006      S       EQU     6       3-BIT SEGMENT NUMBER
0034      0007      SUM     EQU     7       VALUE TO BE SHIFTED
0035                *
0036 0000                 AORG    0
0037                *
0038 0000 7E01    INIT    LACK    1
0039 0001 5001            SACL    ONE
0040 0002 2701            LAC     ONE,7
0041 0003 5002            SACL    BIT7
0042                *
0043                * INVERT INPUT
0044                *
0045 0004 4003    START   IN      Y,0
0046 0005 7EFF            LACK    >00FF
0047 0006 7803            XOR     Y
0048 0007 5003            SACL    Y
0049                *
0050                * SAVE POLARITY AND STRIP TO LOW 7 BITS
0051 0008 7902            AND     BIT7
0052 0009 5005            SACL    P       0000 FOR POS; 0080 FOR NEG
0053 000A 7E7F            LACK    >007F
0054 000B 7903            AND     Y
```

```
0055 000C 5003          SACL    Y
0056                *
0057                * MAGNITUDE IS CORRECT. STRIP Y OF S AND Q.
0058 000D 2C03          LAC     Y,12    SHIFT S INTO HIGH HALF OF ACC
0059 000E 5806          SACH    S
0060 000F 2006          LAC     S       CHECK FOR SEGMENT 0
0061 0010 FE00          BNZ     SEGNZ
     0011 0016
0062                * SEGMENT 0: EXPAND X TO 2*Q + 1
0063 0012 2103          LAC     Y,1
0064 0013 0001          ADD     ONE
0065 0014 F900          B       SIGN
     0015 001D
0066                * NONZERO SEGMENT: SUM = 2*Q + 33
0067 0016 7E21  SEGNZ   LACK    33
0068 0017 0103          ADD     Y,1
0069 0018 1506          SUB     S,5     REMOVE S BITS
0070 0019 5007          SACL    SUM
0071                *
0072                * SHIFT SUM BY S USING VARIABLE SHIFT ROUTINE AT SBASE
0073 001A 7E20          LACK    SBASE-2 OFFSET (MINUS 0 CASE)
0074 001B 0106          ADD     S,1     DOUBLE S (2 WDS/SHIFT SEGMENT)
0075 001C 7F8C          CALA            SHIFT SUM BY S
0076                *
0077                * ACC = MAGNITUDE. ADD POLARITY TO BIT 12.
0078 001D 0505  SIGN    ADD     P,5     SHIFT P TO BIY 12
0079 001E 5004          SACL    X
0080 001F 4804          OUT     X,0     OUTPUT RESULT TO PORT 0
0081 0020 F900  FIN     B       FIN
     0021 0020
0082                *
0083                * LOAD SUM SHIFTED 0:6
0084 0022 2007  SBASE   LAC     SUM,0
0085 0023 7F8D          RET
0086 0024 2107          LAC     SUM,1
0087 0025 7F8D          RET
0088 0026 2207          LAC     SUM,2
0089 0027 7F8D          RET
0090 0028 2307          LAC     SUM,3
0091 0029 7F8D          RET
0092 002A 2407          LAC     SUM,4
0093 002B 7F8D          RET
0094 002C 2507          LAC     SUM,5
0095 002D 7F8D          RET
0096 002E 2607          LAC     SUM,6
0097 002F 7F8D          RET
0098                *
0099                    END
NO ERRORS, NO WARNINGS
```

Tables 9 and 10 are included to aid in verifying particular implementations of the algorithms that have been presented.

Table 9. Segmented μ-255 Companding*
(Courtesy of John Wiley & Sons, see Reference 6)

	Segment S								Quantization	
	000	001	010	011	100	101	110	111	BIN	Q
Quantization Endpoints	0	31	95	223	479	991	2015	4063	0000	0
	1	35	103	239	511	1055	2143	4319	0001	1
	3	39	111	255	543	1119	2271	4575	0010	2
	5	43	119	271	575	1183	2399	4831	0011	3
	7	47	127	287	607	1247	2527	5087	0100	4
	9	51	135	303	639	1311	2655	5343	0101	5
	11	55	143	319	671	1375	2783	5599	0110	6
	13	59	151	335	703	1439	2911	5855	0111	7
	15	63	159	351	735	1503	3039	6111	1000	8
	17	67	167	367	767	1567	3167	6367	1001	9
	19	71	175	383	799	1631	3295	6623	1010	10
	21	75	183	399	831	1695	3423	6879	1011	11
	23	79	191	415	863	1759	3551	7135	1100	12
	25	83	199	431	895	1823	3679	7391	1101	13
	27	87	207	447	927	1887	3807	7647	1110	14
	29	91	215	463	959	1951	3935	7903	1111	15
	31	95	223	479	991	2015	4063	8159		

* (1) Sample values are referenced to a full-scale value of 8159. (2) Negative samples are encoded in sign-magnitude format with a polarity bit of 1. (3) In actual transmission the codes are inverted to increase the density of 1's when low signal amplitudes are encoded. (4) Analog output samples are decoded as the center of the encoded quantization interval. (5) Quantization error is the difference between the reconstructed output value and the original input sample value.

Table 10. Segmented A-Law Companding
(Courtesy of John Wiley & Sons, see Reference 6)

	Segment S								Quantization	
	000	001	010	011	100	101	110	111	BIN	Q
Quantization Endpoints	0	32	64	128	256	512	1024	2048	0000	0
	2	34	68	136	272	544	1088	2176	0001	1
	4	36	72	144	288	576	1152	2304	0010	2
	6	38	76	152	304	608	1216	2432	0011	3
	8	40	80	160	320	640	1280	2560	0100	4
	10	42	84	168	336	672	1344	2688	0101	5
	12	44	88	176	352	704	1408	2816	0110	6
	14	46	92	184	368	736	1472	2944	0111	7
	16	48	96	192	384	768	1536	3072	1000	8
	18	50	100	200	400	800	1600	3200	1001	9
	20	52	104	208	416	832	1664	3328	1010	10
	22	54	108	216	432	864	1728	3456	1011	11
	24	56	112	224	448	896	1792	3584	1100	12
	26	58	116	232	464	928	1856	3712	1101	13
	28	60	120	240	480	960	1920	3840	1110	14
	30	62	124	248	496	992	1984	3968	1111	15
	32	64	128	256	512	1024	2048	4096		

REFERENCES

1. *TCM2913, TCM2914, TCM2916, TCM2917 Combined Single-Chip PCM Codec and Filter* (Data Sheet SCTS012), Texas Instruments (1983).
2. J.B. Reimer, M.L. McMahan, and M. Arjmand, *32-kbit/s ADPCM with the TMS32010* (Application Report), Texas Instruments (1985).
3. J. Robillard, *Telcommunications Interfacing to the TMS32010* (Application Report), Texas Instruments (1985).
4. D.G. Messerschmitt, D.J. Hedberg, C.R. Cole, A. Haoui, and P. Winship, *Digital Voice Echo Canceller with a TMS32020* (Application Report), Texas Instruments (1985).
5. J.L. Fike and G.E. Friend, *Understanding Telephone Electronics*, Texas Instruments (1983).
6. J.C. Bellamy, *Digital Telephony*, John Wiley & Sons (1982). (Figures and tables reprinted by permission of John Wiley & Sons.)

APPENDIX

COMPANDING ROUTINES FOR THE TMS32020

This appendix provides companding programs for the TMS32020. The basic theory and operation are similar to what is described for the TMS32010 in the major portion of this report.

Programs included in the appendix for μ-law and A-law expansion and compression utilize the serial port of the TMS32020 for 8-bit serial I/O. The routines are interrupt-driven to allow direct interfacing to a codec, such as the Texas Instruments TCM2913 (see the data sheet for further information). The following paragraphs briefly describe each of the programs included in the appendix.

The first program reads a μ-law value from the Data Receive Register (DRR), expands it to a 14-bit linear value, and writes it to location X. This value is then compressed back to an 8-bit μ-law value and written to the Data Transmit Register (DXR). Since μ-law codecs invert all bits of the 8-bit value for transmission, XORK (exclusive-OR immediate with accumulator with shift) instructions are used to perform inversion in the TMS32020 before expansion and after compression. Note also that the LACT (load accumulator with shift specified by T register) instruction is useful for performing the conditional shift implemented in the TMS32020 program by a computed subroutine call (CALA). The compression routine of the program assumes a left-justified 14-bit value within the accumulator. Therefore, the value stored in X is left-shifted twice before the compression routine is entered. The NORM (normalize contents of the accumulator) instruction is then used to find the MSB of the accumulator by performing an in-place accumulator left-shift if the two MSBs are the same. At the same time, the count of the left-shifts is maintained in auxiliary register 0 (AR0) and used to compute the segment number S.

The second program performs A-law expansion and compression, and is similar to the μ-law program. For A-law transmission, however, only the even-order bits of the 8-bit value are inverted. Note that the LACT and NORM instructions are still used to compute the expanded and compressed values.

The third program is an example of a simplified solution to μ-law expansion. The 8-bit μ-law value is used as an index into a 256-word table of 14-bit linear values accessed by the TBLR (table write) instruction. This lookup-table approach may also be utilized for A-law expansion. While this method is obviously the fastest method of performing expansion, it is also the most inefficient in terms of program memory requirements.

```
0001                ***************************************************************
0002                *                          PROGRAM 1                          *
0003                *                                                             *
0004                *   U-LAW EXPAND/COMPRESS CODEC LOOPBACK PROGRAM. UPON        *
0005                *   RECEIVING AN RINT INSTRUCTION, DRR IS READ, EXPANDED,     *
0006                *   COMPRESSED BACK, AND WRITTEN TO DXR, AT WHICH TIME THE    *
0007                *   PROCESSOR IDLES UNTIL ANOTHER RINT INSTRUCTION IS         *
0008                *   GENERATED. THE FIRST WORD TRANSMITTED IS A ZERO.          *
0009                ***************************************************************
0010                *
0011        0000    DRR         EQU   0
0012        0001    DXR         EQU   1
0013        0004    IMR         EQU   4
0014        0060    S           EQU   96              * U-LAW SEGMENT NUMBER
0015        0061    BIAS        EQU   97              * = 33
0016        0062    X           EQU   98
0017        0063    SUM         EQU   99
0018        0064    SIGN        EQU   100
0019        0065    NEG7        EQU   101             * = >FFF9
0020        0066    Q           EQU   102             * U-LAW QUANTIZATION BIN
0021        0067    BIAS2       EQU   103
0022 0000           RSVECT      AORG  0
0023 0000 FF80                  B     INIT
     0001 0400
0024 001A                       AORG  26
0025 001A 9800      EXPAND      BIT   DRR,8           * TEST DRR FOR SIGN
0026 001B 2C00                  LAC   DRR,12
0027 001C D404                  ANDK  >7F00,4         * ZERO SIGN AND OTHER MSBS
     001D 7F00
0028 001E D406                  XORK  >7F00,4         * INVERT ALL BITS
     001F 7F00
0029 0020 6860                  SACH  S
0030 0021 4460                  SUBH  S               * ZERO ACCH
0031 0022 0B61                  ADD   BIAS,11
0032 0023 CE18                  SFL
0033 0024 6C63                  SACH  SUM,4
0034 0025 3C60                  LT    S
0035 0026 4263                  LACT  SUM
0036 0027 1061                  SUB   BIAS
0037 0028 F980                  BBNZ  POSVAL          * POSITIVE IF TC = 1
     0029 002B
0038 002A CE23                  NEG
0039 002B 6062      POSVAL      SACL  X
0040                *
0041 002C 2262      JSTIFY      LAC   X,2             * LEFT-JUSTIFY 14-BIT NUMBER
0042 002D 6062                  SACL  X
0043                *
0044 002E 4062      COMPRS      ZALH  X               * U-LAW COMPRESS # IN ACCH
0045 002F F380                  BLZ   NEGCMP
     0030 0040
0046 0031 4867                  ADDH  BIAS2           * (# ALREADY LEFT-JUSTIFIED)
0047 0032 3065                  LAR   AR0,NEG7
0048 0033 CB06                  RPTK  6               * FIND MSB
0049 0034 CEA2                  NORM
0050 0035 DE04                  ANDK  >F000,14        * ZERO 2 MSBS & ALL LSBS
     0036 F000
0051 0037 6866                  SACH  Q
```

```
0052 0038 7060            SAR   AR0,S
0053 0039 4060            ZALH  S
0054 003A CE1B            ABS
0055 003B 0266            ADD   Q,2
0056 003C D406            XORK  >FF00,4       * INVERT ALL BITS
     003D FF00
0057 003E FF80            B     SATCH
     003F 004E
0058                 *
0059 0040 CE1B  NEGCMP    ABS
0060 0041 4867            ADDH  BIAS2         * (# ALREADY LEFT-JUSTIFIED)
0061 0042 3065            LAR   AR0,NEG7
0062 0043 CB06            RPTK  6             * FIND MSB
0063 0044 CEA2            NORM
0064 0045 DE04            ANDK  >F000,14      * ZERO 2 MSBS & ALL LSBS
     0046 F000
0065 0047 6866            SACH  Q
0066 0048 7060            SAR   AR0,S
0067 0049 4060            ZALH  S
0068 004A CE1B            ABS
0069 004B 0266            ADD   Q,2
0070 004C D406            XORK  >7F00,4       * INVERT ALL BITS IN Q
     004D 7F00
0071                 *                          (P=0 FOR NEGATIVE VALUES)
0072 004E 6C01  SATCH     SACH  DXR,4
0073                 *
0074 004F CE00  WAIT      EINT
0075 0050 CE1F            IDLE                * WAIT FOR RINT
0076                 *
0077                 *
0078                 *         INITIALIZATION ROUTINE
0079                 *
0080                 *
0081 0400                 AORG  1024
0082 0400 C800  INIT      LDPK  0             * POINT DP TO B2 AND MMRS
0083 0401 CA10            LACK  >10
0084 0402 6004            SACL  IMR           * ENABLE RINT BUT DISABLE
0085                 *                          ALL OTHERS
0086 0403 CE0F            FORT  1             * CONFIGURE SERIAL PORT TO
0087                 *                          BYTE MODE
0088 0404 CE03            SOVM
0089 0405 CE07            SSXM
0090 0406 CE08            SPM   0
0091 0407 CA21            LACK  33
0092 0408 6061            SACL  BIAS
0093 0409 2261            LAC   BIAS,2
0094 040A 6067            SACL  BIAS2
0095 040B D001            LALK  -7
     040C FFF9
0096 040D 6065            SACL  NEG7
0097 040E CA00            LACK  0
0098 040F 6000            SACL  DRR           * ZERO MSBS OF DRR
0099 0410 6001            SACL  DXR           * ZERO DXR FOR FIRST
0100                 *                          TRANSMIT OPERATION
0101 0411 5588            LARP  0             * ZERO ARP
0102 0412 5588            LARP  0             *  AND ARB
0103 0413 CE00            EINT
```

```
 0104 0414 CE1F                IDLE                    * WAIT FOR RINT
 0105                          END
NO ERRORS, NO WARNINGS
```

```
0001                ********************************************************
0002                *                      PROGRAM 2                       *
0003                *                                                      *
0004                *   A-LAW EXPAND/COMPRESS CODEC LOOPBACK PROGRAM. UPON  *
0005                *   RECEIVING AN RINT INSTRUCTION, DRR IS READ, EXPANDED,*
0006                *   COMPRESSED BACK, AND WRITTEN TO DXR, AT WHICH TIME THE*
0007                *   PROCESSOR IDLES UNTIL ANOTHER RINT INSTRUCTION IS   *
0008                *   GENERATED. THE FIRST WORD TRANSMITTED IS A ZERO.    *
0009                ********************************************************
0010                *
0011      0000      DRR       EQU   0
0012      0001      DXR       EQU   1
0013      0004      IMR       EQU   4
0014      0060      S         EQU   96            * A-LAW SEGMENT NUMBER
0015      0061      BIAS      EQU   97            * = 33
0016      0062      X         EQU   98
0017      0063      SUM       EQU   99
0018      0064      SIGN      EQU   100
0019      0065      NEG7      EQU   101           * = >FFF9
0020      0066      Q         EQU   102           * A-LAW QUANTIZATION BIN
0021      0067      ONE       EQU   103
0022 0000           RSVECT    AORG  0
0023 0000 FF80      RSVECT    B     INIT
     0001 0400
0024 001A           EXPAND    AORG  26
0025 001A 9800      EXPAND    BIT   DRR,8         * TEST DRR FOR SIGN
0026 001B 2C00                LAC   DRR,12
0027 001C D404                ANDK  >7F00,4       * ZERO SIGN AND OTHER MSBS
     001D 7F00
0028 001E D406                XORK  >5500,4       * INVERT EVEN-ORDER BITS
     001F 5500
0029 0020 6860                SACH  S
0030 0021 4460                SUBH  S             * ZERO ACCH
0031 0022 3060                LAR   AR0,S
0032 0023 FB90                BANZ  SEGNZ,*-      * TEST FOR SEGMENT NUMBER 0
     0024 002D
0033 0025 0B67                ADD   ONE,11
0034 0026 CE18                SFL
0035 0027 F880                BBZ   PSVAL1
     0028 002A
0036 0029 CE23                NEG
0037 002A 6C62      PSVAL1    SACH  X,4
0038 002B FF80                B     JSTIFY
     002C 0037
0039                *
0040 002D 7060      SEGNZ     SAR   AR0,S         * STORE DECREMENTED S
0041 002E 0B61                ADD   BIAS,11
0042 002F CE18                SFL
0043 0030 6C63                SACH  SUM,4
0044 0031 3C60                LT    S
0045 0032 4263                LACT  SUM
0046 0033 F880                BBZ   POSVAL
     0034 0036
0047 0035 CE23                NEG
0048 0036 6062      POSVAL    SACL  X
0049                *
0050 0037 2362      JSTIFY    LAC   X,3
```

```
0051 0038 6062                SACL  X                 * LEFT-JUSTIFY 13-BIT NUMBER
0052                          *
0053 0039 4062  COMPRS        ZALH  X                 * A-LAW COMPRESS # IN ACCH
0054 003A F380                BLZ   NEGCMP
     003B 0052
0055 003C 3065                LAR   AR0,NEG7
0056 003D CB06                RPTK  6                 * FIND MSB
0057 003E CEA2                NORM
0058 003F FB80                BANZ  SEGNZ1,*
     0040 0047
0059 0041 6866                SACH  Q                 * SEGMENT NUMBER = 0
0060 0042 2166                LAC   Q,1
0061 0043 D406                XORK  >5500,4           * INVERT EVEN-ORDER BITS
     0044 5500
0062 0045 FF80                B     SATCH
     0046 0067
0063 0047 DE04  SEGNZ1        ANDK  >F000,14          * ZERO 2 MSBS & ALL LSBS
     0048 F000
0064 0049 6866                SACH  Q
0065 004A 7060                SAR   AR0,S
0066 004B 4060                ZALH  S
0067 004C CE1B                ABS
0068 004D 0266                ADD   Q,2
0069 004E D406                XORK  >5500,4           * INVERT EVEN-ORDER BITS
     004F 5500
0070 0050 FF80                B     SATCH
     0051 0067
0071                          *
0072 0052 CE1B  NEGCMP        ABS
0073 0053 3065                LAR   AR0,NEG7
0074 0054 CB06                RPTK  6                 * FIND MSB
0075 0055 CEA2                NORM
0076 0056 FB80                BANZ  SEGNZ2,*
     0057 005E
0077 0058 6866                SACH  Q                 * SEGMENT NUMBER = 0
0078 0059 2166                LAC   Q,1
0079 005A D406                XORK  >D500,4           * INVERT EVEN-ORDER BITS
     005B D500
0080                          *                         AND SET SIGN BIT TO 1
0081 005C FF80                B     SATCH
     005D 0067
0082 005E DE04  SEGNZ2        ANDK  >F000,14          * ZERO 2 MSBS & ALL LSBS
     005F F000
0083 0060 6866                SACH  Q
0084 0061 7060                SAR   AR0,S
0085 0062 4060                ZALH  S
0086 0063 CE1B                ABS
0087 0064 0266                ADD   Q,2
0088 0065 D406                XORK  >D500,4           * INVERT EVEN-ORDER BITS
     0066 D500
0089                          *                         AND SET SIGN BIT TO 1
0090 0067 6C01  SATCH         SACH  DXR,4
0091                          *
0092 0068 CE00  WAIT          EINT
0093 0069 CE1F                IDLE                    * WAIT FOR RINT
0094                          *
0095                          *
```

```
0096                    *           INITIALIZATION ROUTINE
0097                    *
0098                    *
0099 0400                           AORG  1024
0100 0400 C800  INIT                LDPK  0             * POINT DP TO B2 AND MMRS
0101 0401 CA10                      LACK  >10
0102 0402 6004                      SACL  IMR           * ENABLE RINT BUT DISABLE
0103                    *                                   ALL OTHERS
0104 0403 CE0F                      FORT  1             * CONFIGURE SERIAL PORT TO
0105                    *                                   BYTE MODE
0106 0404 CE03                      SOVM
0107 0405 CE07                      SSXM
0108 0406 CE08                      SPM   0
0109 0407 CA01                      LACK  1
0110 0408 6067                      SACL  ONE
0111 0409 CA21                      LACK  33
0112 040A 6061                      SACL  BIAS
0113 040B D001                      LALK  -7
     040C FFF9
0114 040D 6065                      SACL  NEG7
0115 040E CA00                      LACK  0
0116 040F 6000                      SACL  DRR           * ZERO MSBS OF DRR
0117 0410 6001                      SACL  DXR           * ZERO DXR FOR FIRST
0118                    *                                   TRANSMIT OPERATION
0119 0411 5588                      LARP  0             * ZERO ARP
0120 0412 5588                      LARP  0             *  AND ARB
0121 0413 CE00                      EINT
0122 0414 CE1F                      IDLE                * WAIT FOR RINT
0123                                END
NO ERRORS, NO WARNINGS
```

```
0001                ************************************************************
0002                *                          PROGRAM 3                       *
0003                *                                                          *
0004                *   U-LAW TABLE LOOKUP EXPANSION PROGRAM. UPON RECEIVING   *
0005                *   AN RINT INSTRUCTION, DRR IS READ, EXPANDED, AND        *
0006                *   WRITTEN TO THE STORAGE LOCATION X, AT WHICH TIME THE   *
0007                *   PROCESSOR IDLES UNTIL ANOTHER RINT INSTRUCTION IS      *
0008                *   GENERATED. DXR IS SET TO ZERO SO THAT ALL WORDS        *
0009                *   TRANSMITTED ARE ZEROES.                                *
0010                ************************************************************
0011                *
0012        0000    DRR         EQU   0
0013        0001    DXR         EQU   1
0014        0004    IMR         EQU   4
0015        0060    X           EQU   96
0016        0061    BADDR       EQU   97              * CONTAINS BASE ADDR FOR
0017                *                                   TABLE
0018 0000           RSVECT      AORG  0
0019 0000 FF80      RSVECT      B     INIT
     0001 001F
0020 001A           EXPAND      AORG  26
0021 001A 2000      EXPAND      LAC   DRR
0022 001B 0061                  ADD   BADDR           * ADD TABLE BASE ADDR FOR
0023                *                                   U-LAW EXPANSION TABLE
0024                *                                   LOOKUP
0025 001C 5860                  TBLR  X               * READ INTO LOCATION X
0026                *
0027 001D CE00                  EINT
0028 001E CE1F                  IDLE                  * WAIT FOR RINT
0029                *
0030                *
0031                *           INITIALIZATION ROUTINE
0032                *
0033                *
0034 001F C800      INIT        LDPK  0               * POINT DP TO B2 AND MMRS
0035 0020 CA10                  LACK  >10
0036 0021 6004                  SACL  IMR             * ENABLE RINT BUT DISABLE
0037                *                                   ALL OTHERS
0038 0022 CE0F                  FORT  1               * CONFIGURE SERIAL PORT TO
0039                *                                   BYTE MODE
0040 0023 CE03                  SOVM
0041 0024 CE07                  SSXM
0042 0025 CE08                  SPM   0
0043 0026 D001                  LALK  XTBL
     0027 0300
0044 0028 6061                  SACL  BADDR           * BASE ADDR FOR TBL LOOKUP
0045 0029 CA00                  LACK  0
0046 002A 6000                  SACL  DRR             * ZERO MSBS OF DRR
0047 002B 6001                  SACL  DXR             * ZERO DXR FOR FIRST
0048                *                                   TRANSMIT OPERATION
0049 002C CE00                  EINT
0050 002D CE1F                  IDLE                  * WAIT FOR RINT
0051                *
0052                *
0053                *           TABLE FOR U-LAW EXPANSION TABLE LOOKUP
0054                *
0055                *
```

```
0056 0300                    AORG  768
0057       0300  XTBL        EQU   $
0058                         *
0059 0300 E0A1               DATA  >E0A1          * NEGATIVE VALUES FIRST
0060 0301 E1A1               DATA  >E1A1          *  (FF, FE, ETC.)
0061 0302 E2A1               DATA  >E2A1
0062 0303 E3A1               DATA  >E3A1
0063 0304 E4A1               DATA  >E4A1
0064 0305 E5A1               DATA  >E5A1
0065 0306 E6A1               DATA  >E6A1
0066 0307 E7A1               DATA  >E7A1
0067 0308 E8A1               DATA  >E8A1
0068 0309 E9A1               DATA  >E9A1
0069 030A EAA1               DATA  >EAA1
0070 030B EBA1               DATA  >EBA1
0071 030C ECA1               DATA  >ECA1
0072 030D EDA1               DATA  >EDA1
0073 030E EEA1               DATA  >EEA1
0074 030F EFA1               DATA  >EFA1
0075 0310 F061               DATA  >F061
0076 0311 F0E1               DATA  >F0E1
0077 0312 F161               DATA  >F161
0078 0313 F1E1               DATA  >F1E1
0079 0314 F261               DATA  >F261
0080 0315 F2E1               DATA  >F2E1
0081 0316 F361               DATA  >F361
0082 0317 F3E1               DATA  >F3E1
0083 0318 F461               DATA  >F461
0084 0319 F4E1               DATA  >F4E1
0085 031A F561               DATA  >F561
0086 031B F5E1               DATA  >F5E1
0087 031C F661               DATA  >F661
0088 031D F6E1               DATA  >F6E1
0089 031E F761               DATA  >F761
0090 031F F7E1               DATA  >F7E1
0091 0320 F841               DATA  >F841
0092 0321 F881               DATA  >F881
0093 0322 F8C1               DATA  >F8C1
0094 0323 F901               DATA  >F901
0095 0324 F941               DATA  >F941
0096 0325 F981               DATA  >F981
0097 0326 F9C1               DATA  >F9C1
0098 0327 FA01               DATA  >FA01
0099 0328 FA41               DATA  >FA41
0100 0329 FA81               DATA  >FA81
0101 032A FAC1               DATA  >FAC1
0102 032B FB01               DATA  >FB01
0103 032C FB41               DATA  >FB41
0104 032D FB81               DATA  >FB81
0105 032E FBC1               DATA  >FBC1
0106 032F FC01               DATA  >FC01
0107 0330 FC31               DATA  >FC31
0108 0331 FC51               DATA  >FC51
0109 0332 FC71               DATA  >FC71
0110 0333 FC91               DATA  >FC91
0111 0334 FCB1               DATA  >FCB1
0112 0335 FCD1               DATA  >FCD1
```

```
0113 0336 FCF1          DATA  >FCF1
0114 0337 FD11          DATA  >FD11
0115 0338 FD31          DATA  >FD31
0116 0339 FD51          DATA  >FD51
0117 033A FD71          DATA  >FD71
0118 033B FD91          DATA  >FD91
0119 033C FDB1          DATA  >FDB1
0120 033D FDD1          DATA  >FDD1
0121 033E FDF1          DATA  >FDF1
0122 033F FE11          DATA  >FE11
0123 0340 FE29          DATA  >FE29
0124 0341 FE39          DATA  >FE39
0125 0342 FE49          DATA  >FE49
0126 0343 FE59          DATA  >FE59
0127 0344 FE69          DATA  >FE69
0128 0345 FE79          DATA  >FE79
0129 0346 FE89          DATA  >FE89
0130 0347 FE99          DATA  >FE99
0131 0348 FEA9          DATA  >FEA9
0132 0349 FEB9          DATA  >FEB9
0133 034A FEC9          DATA  >FEC9
0134 034B FED9          DATA  >FED9
0135 034C FEE9          DATA  >FEE9
0136 034D FEF9          DATA  >FEF9
0137 034E FF09          DATA  >FF09
0138 034F FF19          DATA  >FF19
0139 0350 FF25          DATA  >FF25
0140 0351 FF2D          DATA  >FF2D
0141 0352 FF35          DATA  >FF35
0142 0353 FF3D          DATA  >FF3D
0143 0354 FF45          DATA  >FF45
0144 0355 FF4D          DATA  >FF4D
0145 0356 FF55          DATA  >FF55
0146 0357 FF5D          DATA  >FF5D
0147 0358 FF65          DATA  >FF65
0148 0359 FF6D          DATA  >FF6D
0149 035A FF75          DATA  >FF75
0150 035B FF7D          DATA  >FF7D
0151 035C FF85          DATA  >FF85
0152 035D FF8D          DATA  >FF8D
0153 035E FF95          DATA  >FF95
0154 035F FF9D          DATA  >FF9D
0155 0360 FFA3          DATA  >FFA3
0156 0361 FFA7          DATA  >FFA7
0157 0362 FFAB          DATA  >FFAB
0158 0363 FFAF          DATA  >FFAF
0159 0364 FFB3          DATA  >FFB3
0160 0365 FFB7          DATA  >FFB7
0161 0366 FFBB          DATA  >FFBB
0162 0367 FFBF          DATA  >FFBF
0163 0368 FFC3          DATA  >FFC3
0164 0369 FFC7          DATA  >FFC7
0165 036A FFCB          DATA  >FFCB
0166 036B FFCF          DATA  >FFCF
0167 036C FFD3          DATA  >FFD3
0168 036D FFD7          DATA  >FFD7
0169 036E FFDB          DATA  >FFDB
```

```
0170 036F FFDF              DATA  >FFDF
0171 0370 FFE2              DATA  >FFE2
0172 0371 FFE4              DATA  >FFE4
0173 0372 FFE6              DATA  >FFE6
0174 0373 FFE8              DATA  >FFE8
0175 0374 FFEA              DATA  >FFEA
0176 0375 FFEC              DATA  >FFEC
0177 0376 FFEE              DATA  >FFEE
0178 0377 FFF0              DATA  >FFF0
0179 0378 FFF2              DATA  >FFF2
0180 0379 FFF4              DATA  >FFF4
0181 037A FFF6              DATA  >FFF6
0182 037B FFF8              DATA  >FFF8
0183 037C FFFA              DATA  >FFFA
0184 037D FFFC              DATA  >FFFC
0185 037E FFFE              DATA  >FFFE
0186 037F 0000              DATA  >0
0187                *
0188 0380 1F5F              DATA  >1F5F         * POSITIVE VALUES NEXT
0189 0381 1E5F              DATA  >1E5F         *   (POLARITY BIT = 1)
0190 0382 1D5F              DATA  >1D5F
0191 0383 1C5F              DATA  >1C5F
0192 0384 1B5F              DATA  >1B5F
0193 0385 1A5F              DATA  >1A5F
0194 0386 195F              DATA  >195F
0195 0387 185F              DATA  >185F
0196 0388 175F              DATA  >175F
0197 0389 165F              DATA  >165F
0198 038A 155F              DATA  >155F
0199 038B 145F              DATA  >145F
0200 038C 135F              DATA  >135F
0201 038D 125F              DATA  >125F
0202 038E 115F              DATA  >115F
0203 038F 105F              DATA  >105F
0204 0390 0F9F              DATA  >F9F
0205 0391 0F1F              DATA  >F1F
0206 0392 0E9F              DATA  >E9F
0207 0393 0E1F              DATA  >E1F
0208 0394 0D9F              DATA  >D9F
0209 0395 0D1F              DATA  >D1F
0210 0396 0C9F              DATA  >C9F
0211 0397 0C1F              DATA  >C1F
0212 0398 0B9F              DATA  >B9F
0213 0399 0B1F              DATA  >B1F
0214 039A 0A9F              DATA  >A9F
0215 039B 0A1F              DATA  >A1F
0216 039C 099F              DATA  >99F
0217 039D 091F              DATA  >91F
0218 039E 089F              DATA  >89F
0219 039F 081F              DATA  >81F
0220 03A0 07BF              DATA  >7BF
0221 03A1 077F              DATA  >77F
0222 03A2 073F              DATA  >73F
0223 03A3 06FF              DATA  >6FF
0224 03A4 06BF              DATA  >6BF
0225 03A5 067F              DATA  >67F
0226 03A6 063F              DATA  >63F
```

```
0227 03A7 05FF              DATA  >5FF
0228 03A8 05BF              DATA  >5BF
0229 03A9 057F              DATA  >57F
0230 03AA 053F              DATA  >53F
0231 03AB 04FF              DATA  >4FF
0232 03AC 04BF              DATA  >4BF
0233 03AD 047F              DATA  >47F
0234 03AE 043F              DATA  >43F
0235 03AF 03FF              DATA  >3FF
0236 03B0 03CF              DATA  >3CF
0237 03B1 03AF              DATA  >3AF
0238 03B2 038F              DATA  >38F
0239 03B3 036F              DATA  >36F
0240 03B4 034F              DATA  >34F
0241 03B5 032F              DATA  >32F
0242 03B6 030F              DATA  >30F
0243 03B7 02EF              DATA  >2EF
0244 03B8 02CF              DATA  >2CF
0245 03B9 02AF              DATA  >2AF
0246 03BA 028F              DATA  >28F
0247 03BB 026F              DATA  >26F
0248 03BC 024F              DATA  >24F
0249 03BD 022F              DATA  >22F
0250 03BE 020F              DATA  >20F
0251 03BF 01EF              DATA  >1EF
0252 03C0 01D7              DATA  >1D7
0253 03C1 01C7              DATA  >1C7
0254 03C2 01B7              DATA  >1B7
0255 03C3 01A7              DATA  >1A7
0256 03C4 0197              DATA  >197
0257 03C5 0187              DATA  >187
0258 03C6 0177              DATA  >177
0259 03C7 0167              DATA  >167
0260 03C8 0157              DATA  >157
0261 03C9 0147              DATA  >147
0262 03CA 0137              DATA  >137
0263 03CB 0127              DATA  >127
0264 03CC 0117              DATA  >117
0265 03CD 0107              DATA  >107
0266 03CE 00F7              DATA  >F7
0267 03CF 00E7              DATA  >E7
0268 03D0 00DB              DATA  >DB
0269 03D1 00D3              DATA  >D3
0270 03D2 00CB              DATA  >CB
0271 03D3 00C3              DATA  >C3
0272 03D4 00BB              DATA  >BB
0273 03D5 00B3              DATA  >B3
0274 03D6 00AB              DATA  >AB
0275 03D7 00A3              DATA  >A3
0276 03D8 009B              DATA  >9B
0277 03D9 0093              DATA  >93
0278 03DA 008B              DATA  >8B
0279 03DB 0083              DATA  >83
0280 03DC 007B              DATA  >7B
0281 03DD 0073              DATA  >73
0282 03DE 006B              DATA  >6B
0283 03DF 0063              DATA  >63
```

```
 0284 03E0 005D              DATA  >5D
 0285 03E1 0059              DATA  >59
 0286 03E2 0055              DATA  >55
 0287 03E3 0051              DATA  >51
 0288 03E4 004D              DATA  >4D
 0289 03E5 0049              DATA  >49
 0290 03E6 0045              DATA  >45
 0291 03E7 0041              DATA  >41
 0292 03E8 003D              DATA  >3D
 0293 03E9 0039              DATA  >39
 0294 03EA 0035              DATA  >35
 0295 03EB 0031              DATA  >31
 0296 03EC 002D              DATA  >2D
 0297 03ED 0029              DATA  >29
 0298 03EE 0025              DATA  >25
 0299 03EF 0021              DATA  >21
 0300 03F0 001E              DATA  >1E
 0301 03F1 001C              DATA  >1C
 0302 03F2 001A              DATA  >1A
 0303 03F3 0018              DATA  >18
 0304 03F4 0016              DATA  >16
 0305 03F5 0014              DATA  >14
 0306 03F6 0012              DATA  >12
 0307 03F7 0010              DATA  >10
 0308 03F8 000E              DATA  >E
 0309 03F9 000C              DATA  >C
 0310 03FA 000A              DATA  >A
 0311 03FB 0008              DATA  >8
 0312 03FC 0006              DATA  >6
 0313 03FD 0004              DATA  >4
 0314 03FE 0002              DATA  >2
 0315 03FF 0000              DATA  >0
 0316                *
 0317                        END
NO ERRORS, NO WARNINGS
```