MDA

MACDONALD DETTWILER & ASSOCIATES LTD.

DIAGNOSTIC RHYME TEST
EVALUATION OF RELP CODED
SPEECH IN MECHANICAL NOISE
ENVIRONMENT USING A REAL-TIME
HARDWARE SIMULATOR
00-3058-R00
NOVEMBER, 1982

MDA

DOC-CR-SP-82-068

MDA

MACDONALD DETTWILER & ASSOCIATES LTD.

VANCOUVER, CANADA

DIAGNOSTIC RHYME TEST EVALUATION OF RELP CODED SPEECH IN MECHANICAL NOISE ENVIRONMENT USING A REAL-TIME HARDWARE SIMULATOR

00-3058-R00

Prepared by:

J.A. MADELEY and P.F. LEE

MacDonald, Dettwiler and Associates Ltd.
3751 Shell Road, Richmond, B.C., Canada, V6X 2Z9
Telephone No.: (604) 278-3411

NOVEMBER, 1982

S/N 02

DEPARTMENT OF COMMUNICATIONS

COMMUNICATIONS RESEARCH CENTRE

RELEASABLE

DOC-CR-SP-82-068

ORIGINAL DOCUMENT REVIEW AND PUBLICATION RECORD

SECTOR DGSTA	BRANCH DSS	DATE 1982

PURPOSE This form is for use during review of the DOC-CR contractor reports.
It is designed to: record decisions for classification,
record reasons for classification and cautionary marking
provide for indexing requirements.

INSTRUCTIONS * 1 copy of the completed form must accompany the contractor report package submitted to the CRC Library.

* Complete the following items as applicable.

1. DOC-CR NO. DOC-CR-SP-82-068	2. DSS CONTRACT NO. unavailable in book	
3. TITLE: Diagnostic rhyme test evaluation of RELP coded speech in mechanical noise environment using a real-time hardware simulator		4. DATE November 1982
5. CONTRACTOR Macdonald Dettwiler & Associates Ltd.		
6. SCIENTIFIC AUTHORITY B. Bryden	7. LOCATION DSS/CRC	8. TEL. NO. 998-2515

9. CONTRACTOR REPORT CLASSIFICATION:

RELEASABLE [XX] CONDITIONALLY RELEASABLE □ NON-RELEASABLE □

* REASONS FOR CLASSIFICATION:

10. NO. OF COPIES SUBMITTED TO LIBRARY:

EXECUTIVE SUMMARY □ FINAL REPORT [XX] 3 copies

....
Scientific Authority's Signature

....
Date

This form is not official therefore it is not signed.

TABLE OF CONTENTS

TABLE OF APPENDICES

TABLE OF ILLUSTRATIONS

* Actual Page Number

TABLE OF ILLUSTRATIONS

* Actual Page Number

SECTION 1

1. INTRODUCTION

A Residual Excited Linear Prediction (RELP) codec (coder/decoder) real-time hardware simulator has been developed recently with funding from the Communications Research Centre (CRC). It is believed to be the first of its kind built in Canada. The simulator permits real-time evaluation of RELP coded speech at various data rates, in particular, 4.8 and 9.6 KBPS. It also provides flexibility to fine tune the original INRS (Institut National de la Recherche Scientifique) developed RELP algorithm through simple firmware changes. Preliminary laboratory testing of RELP coded speech at 4.8 KBPS has revealed that speech quality is judged to be very intelligible and of good quality. In addition, the robustness of the RELP algorithm to background noise makes it particularly attractive for military and law enforcement applications.

In the military environment, mechanical noise in the background is frequently encountered. The popular Linear Prediction Codecs (LPCs) performs poorly in the presence of periodic noise because it interfers with the pitch extraction process. This report attempts to measure the intelligibility of RELP coded speech and its robustness to periodic noise quantitatively through the help of Diagnostic Rhyme Test (DRT). Real-time simulation of alternate high frequency regeneration methods on the hardware simulator are also included in this study to realize a more economical implementation of the RELP codec.

SECTION 2

2. RELP REAL-TIME SIMULATIONS

2.1 Residual Regeneration Through Sample Position Perturbation

The primary source of error in the 9.6 KBPS RELP codec is the reconstruction of the fullband residual from the baseband residual. This can be demonstrated by using unquantized residual and predictor, which gives very little improvements in speech quality over 9.6 KBPS RELP.

A paper presented by R. Viswanathan [1], claimed that a spectral folding high frequency regeneration technique produced higher speech quality than the rectification method used in the RELP. In its simplest form this method involves inserting four zeros after each baseband signal. This causes aliasing terms up to 4 kHz to be generated (see Figure 2.1-1). This simple technique generates objectionable tonal noises.

A perturbation method was described in the paper which masked the tonal noises. This involved randomly perturbing the baseband residual sample to the left or right by one sample position. Small samples were perturbed with a greater probability than large samples to reduce the chance of perturbing pitch pulses. The high pass filtered perturbed signal was added to the low pass unperturbed signal to regenerate the fullband residual (see Figure 2.1-2). The random perturbation was accomplished on the hardware simulator by comparing the absolute value of the sample to be perturbed with a random signed number between -1/8 and 1/8. The random number is extracted from bits truncated in an arithmetic operations. If the sample was greater than the random number, no perturbation was performed, otherwise, the sample was perturbed to the left or right based on a random bit. This generates a conditional probability of perturbation as a function of input sample amplitude given in Figure 2.1-3. The perceived quality of the processed speech was about the same as the rectification method. A 30% reduction in processing resource may be obtained in the synthesis processor using this method. If the perturbed residual was fed into the predictor directly without the low and high frequency paths suggested in the paper (Figure 2.1-4), a further 40% or a total of 70% of the processing resource may be saved in the synthesizer without affecting the subjective quality of the RELP coded speech.

A small number of diagnostic rhyme tests were performed to see if this algorithm had the same intelligibility as RELP. The perturbation algorithm in Figure 2.1-4 is referred to as "PERT" in the rest of this report.

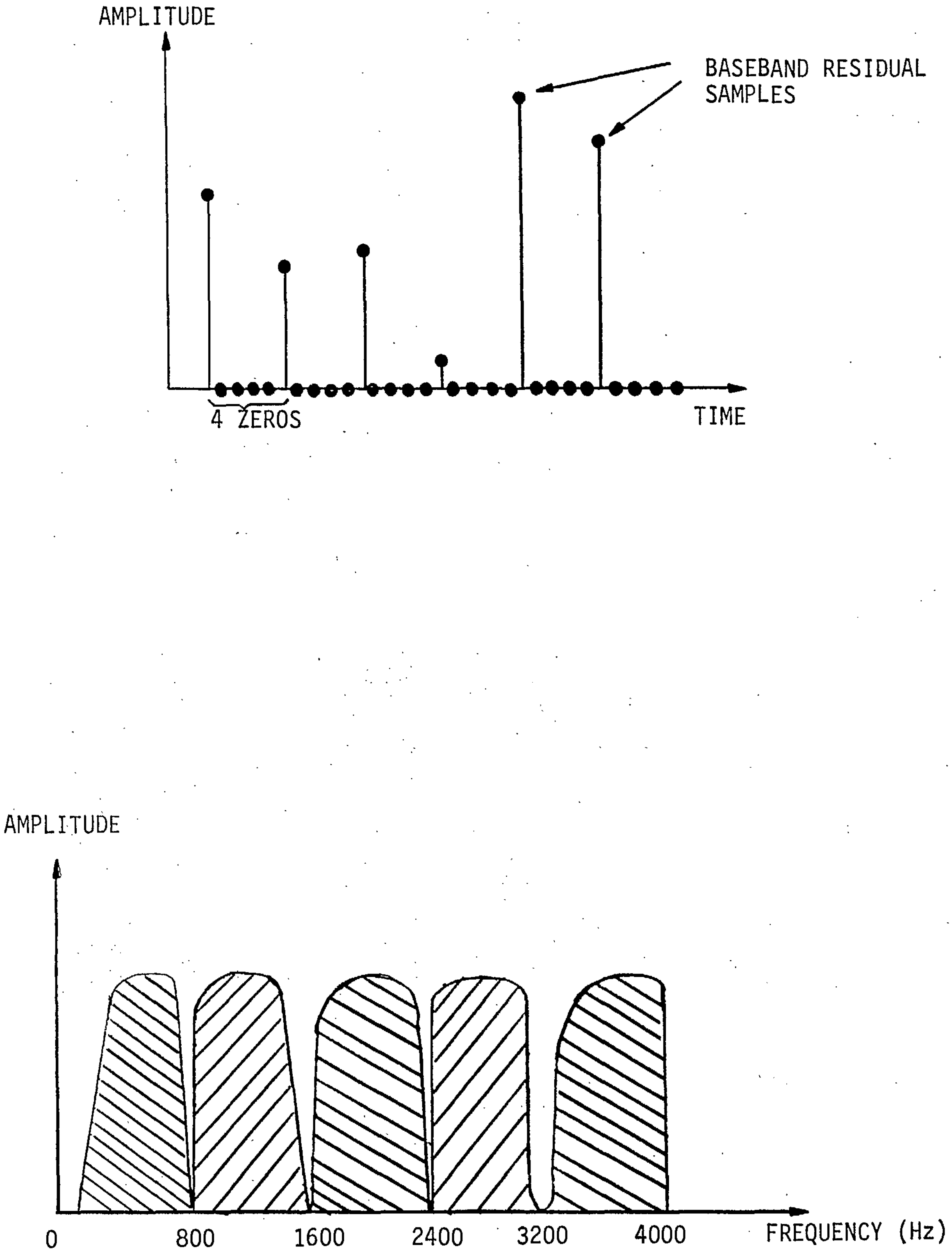


FIGURE 2.1-1 ILLUSTRATION OF SPECTRAL FOLDING

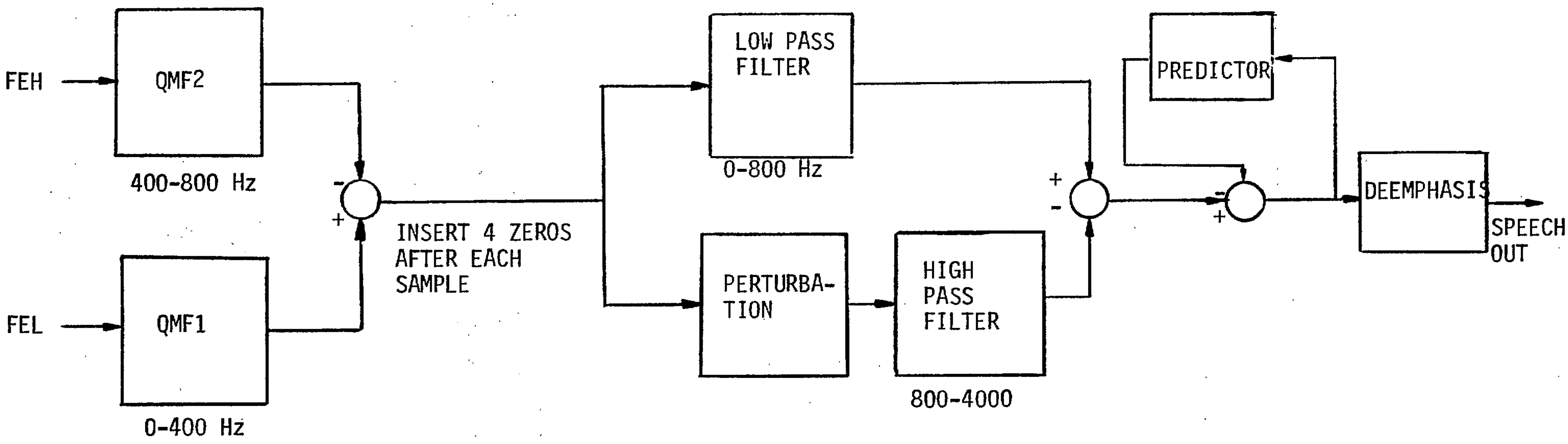
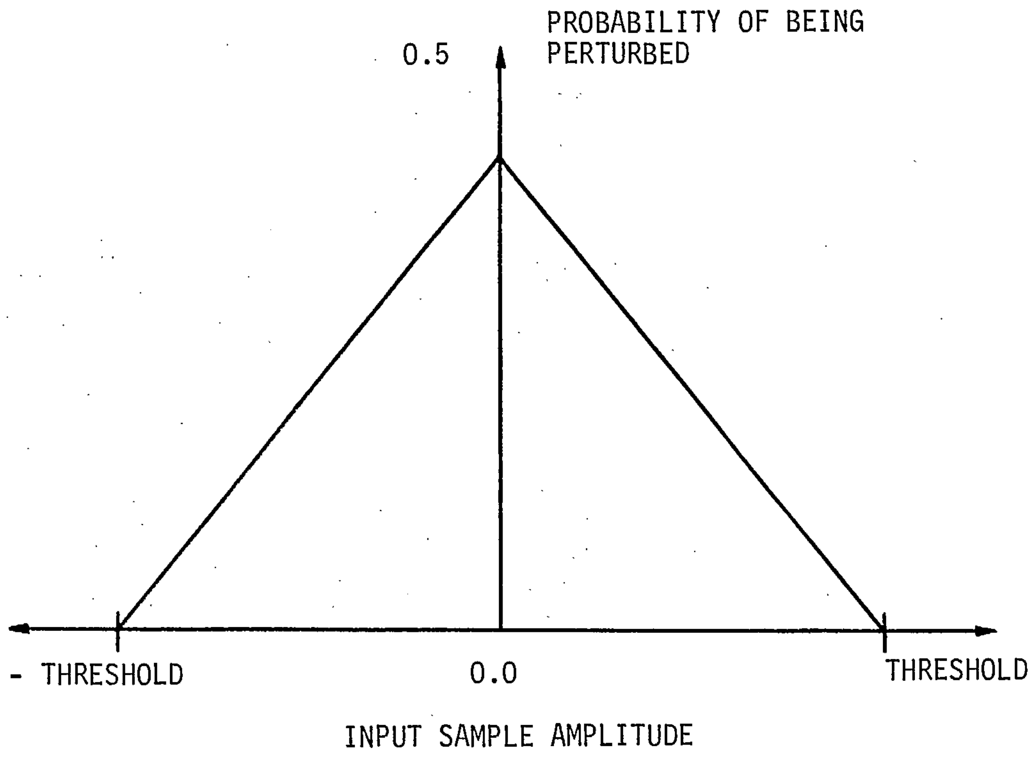


FIGURE 2.1-2 HIGH FREQUENCY REGENERATION WITH AN UNPERTURBED BASEBAND



THRESHOLD = 1/8

FIGURE 2.1-3 CONDITIONAL PERTURBATION PROBABILITY AS A FUNCTION OF INPUT SAMPLE AMPLITUDE

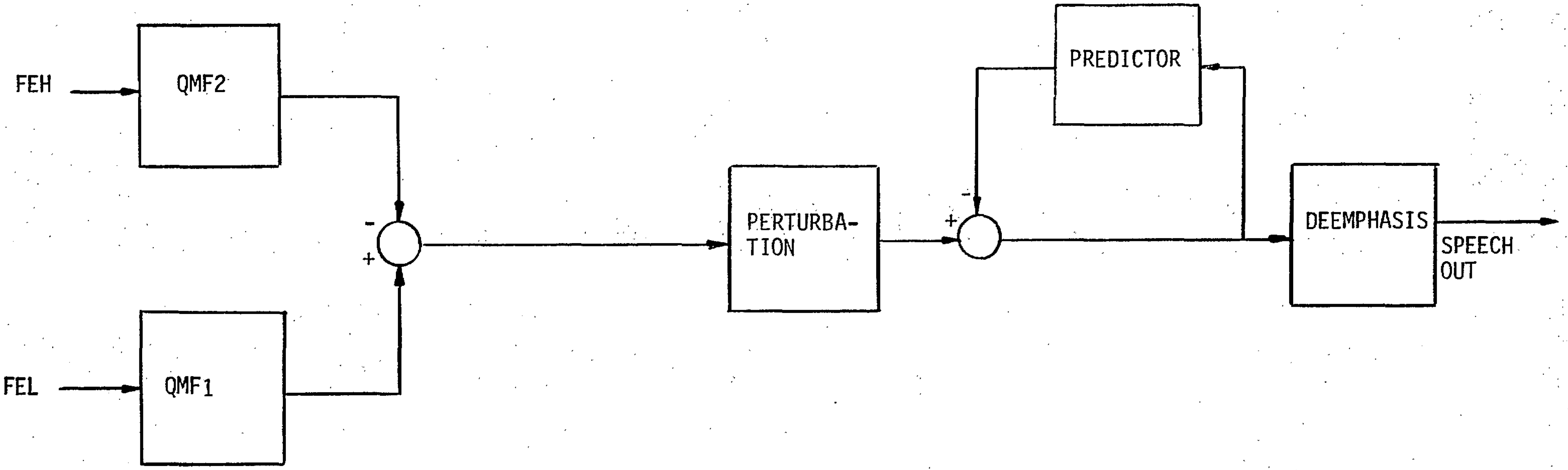


FIGURE 2.1-4 HIGH FREQUENCY REGENERATION WITH A PERTURBED BASEBAND (PERT)

2.2 5-Bit Adaptive Residual Quantizer for 9.6 KBPS Simulation

The existing muldem simulator firmware implements a 2-bit adaptive residual quanitzer which generates 64 bits of residual data for each 20 msec speech frame. These residual data together with the quantized reflection coefficients (22 bits) and housekeeping overhead (10 bits) produce 96 bits of information per speech frame at a rate of 4.8 KBPS. Through the use of a 5-bit adaptive residual quantizer, the information rate will be doubled to 9.6 KBPS. It is expected that the quality of RELP coded speech with 5-bit residual quantization will be perceptibly indistinguishable from that of RELP coded speech without residual quantization.

An adaptive quantizer is one whose step size Δ_n is updated every time an input sample X_n is received. The step size is updated according to the following equation:

$$\Delta_n = \begin{cases} \Delta_{n-1} \cdot \mathrm{QMLT}(i+1); & i\Delta \leqq X_n < (i+1)\Delta \text{ and} \\ & 0 \leqq i \leqq N-2 \\ \Delta_{n-1} \cdot \mathrm{QMLT}(N); & (N-1)\Delta \leqq X_n \end{cases} \qquad [2.1]$$

where N is half the number of quantizer levels, n is the iteration number and QMLT(i) is a set of N postive multiplier constants which are selected to maximize the signal-to-quantization noise ratio based on the statistical property of the quantizer input samples. In addition, the step size is also bounded between Δ_{min} and Δ_{max}. A mid-tread/mid-rise adaptive quantizer characteristic [2] is employed according to Figures 2.2-1 and 2.2-2. The mid-tread quantizer performs like a squelch circuit which attempts to eliminate any residual noise in the absence of speech energy. A step size threshold determines which of two quantizer characters (mid-tread or mid-rise) should be employed to quantize the current input sample. The values 1 through N-1 in units of step size are called the quantizer breakpoints, while the values 1/2 through (2N-1)/2 are the quantizer output levels in units of step size.

In the actual implementation of the 5-bit adaptive quantizer, all signed numbers have to be scaled down to between -1 and +1 as a result of integer arithmetic performed by the 8086 subsystem. The breakpoints and output levels are therefore normalized by N or 16 resulting in numbers between -1 and +1. This calls for the establishment of a new step size parameter FSV which is related to the former step size Δ as follows:

$$\mathrm{FSV} = 16 \cdot \Delta \qquad [2.2]$$

The FSV value is updated in pretty much the same way as Δ defined in Equation 2.1. The FSVMN, FSVMX and FSVTH values are chosen to be 10, 1000 and 15 respectively, identical to those used in the 2-bit quantizer. Other parameters of the 5-bit adaptive quantizer is given in Table 2.2-1. The set of multiplier constants QMLT was adopted from Jayant [3] and was

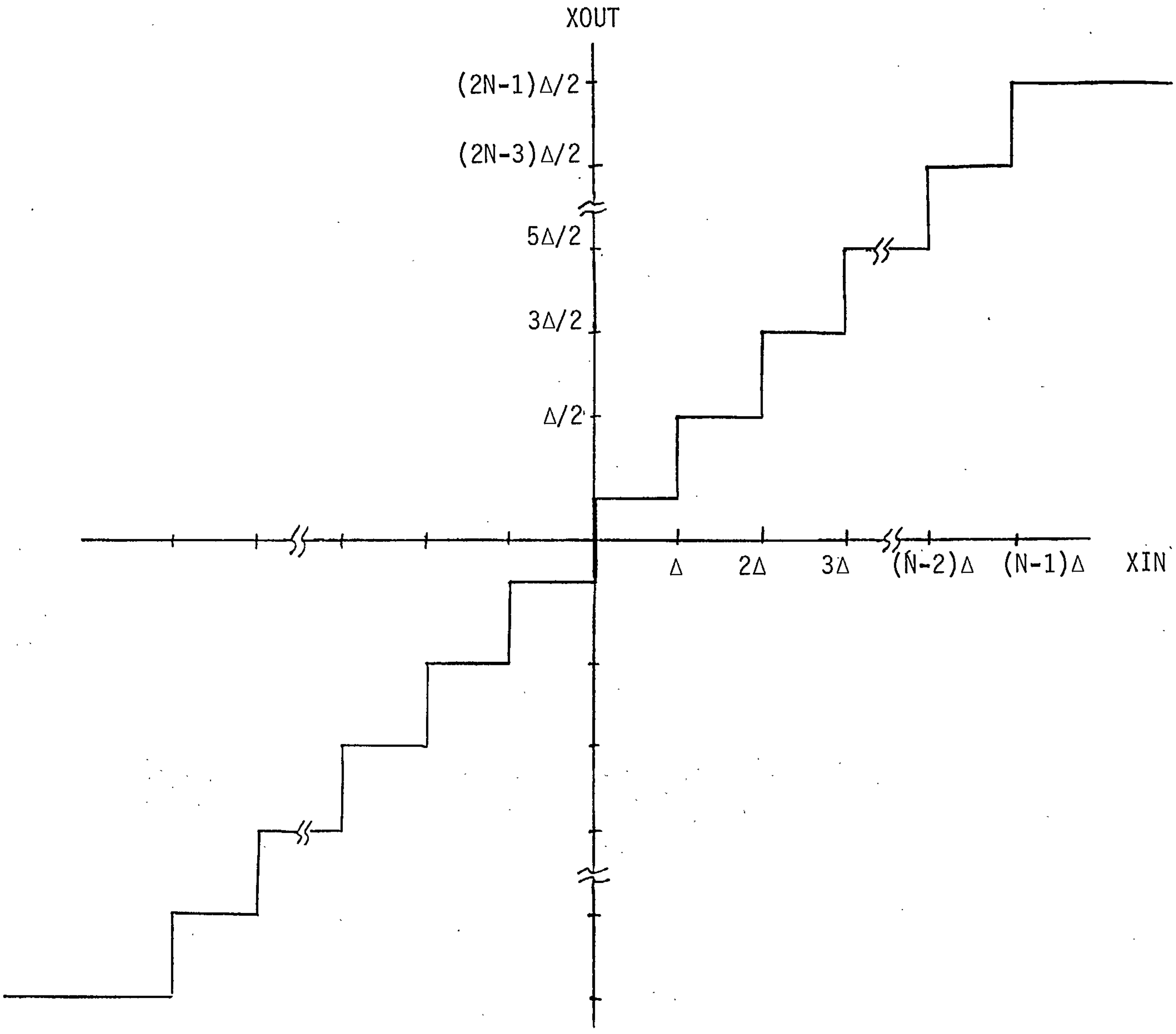


FIGURE 2.2-1 MID-RISE QUANTIZER CHARACTERISTIC FOR STEP SIZE GREATER THAN OR EQUAL TO A PRESET THRESHOLD

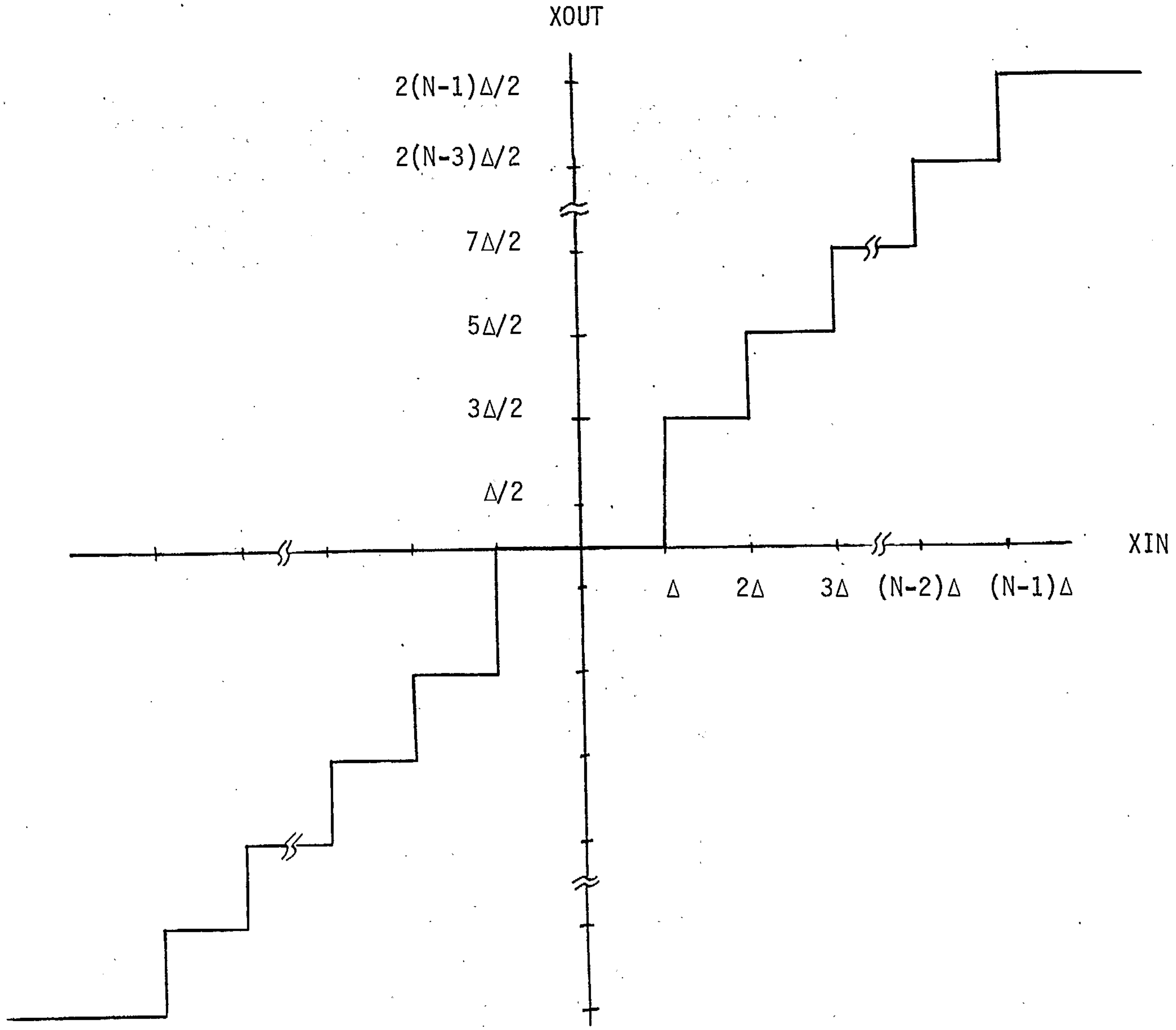


FIGURE 2.2-2 MID-TREAD QUANTIZER CHARACTERISTIC FOR STEP SIZE LESS THAN THE PRESET THRESHOLD

TABLE 2.2-1 5-BIT ADAPTIVE RESIDUAL QUANTIZER PARAMETERS

MULTIPLIERS		NORMALIZED BREAKPOINTS		NORMALIZED OUTPUT LEVELS	
QMLT(1)	0.85	XQ(1)	1/16	Y/Q(1)	1/32
QMLT(2)	0.85	XQ(2)	2/16	Y/Q(2)	3/32
QMLT(3)	0.85	XQ(3)	3/16	Y/Q(3)	5/32
QMLT(4)	0.85	XQ(4)	4/16	Y/Q(4)	7/32
QMLT(5)	0.85	XQ(5)	5/16	Y/Q(5)	9/32
QMLT(6)	0.85	XQ(6)	6/16	Y/Q(6)	11/32
QMLT(7)	0.85	XQ(7)	7/16	Y/Q(7)	13/32
QMLT(8)	0.85	XQ(8)	8/16	Y/Q(8)	15/32
QMLT(9)	1.2	XQ(9)	9/16	Y/Q(9)	17/32
QMLT(10)	1.4	XQ(10)	10/16	Y/Q(10)	19/32
QMLT(11)	1.6	XQ(11)	11/16	Y/Q(11)	21/32
QMLT(12)	1.8	XQ(12)	12/16	Y/Q(12)	23/32
QMLT(13)	2.0	XQ(13)	13/16	Y/Q(13)	25/32
QMLT(14)	2.2	XQ(14)	14/16	Y/Q(14)	27/32
QMLT(15)	2.4	XQ(15)	15/16	Y/Q(15)	29/32
QMLT(16)	2.6			Y/Q(16)	31/32

optimized for PCM speech. No perceptible differences in the quality of RELP coded speech were observed between 5-bit residual quantization and no residual quantization. A listing of the muldem simulator firmware with a 5-bit quantizer is given in Appendix B2.

In the process of modifying the muldem simulator firmware to accommodate the 5-bit adaptive residual quantizer, the software error handling routines were also altered. Up-to-date listings of the muldem simulator firmware with 2-bit residual quantization and no residual quantization are given in Appendix B. The muldem simulator firmware periodically checks software counter and buffer pointer values against their expected values and detects arithmetic overflow conditions to ensure proper firmware operation. In the existing firmware, the residual buffer service counter is checked against a range of values every time the buffer requires service, i.e., read a residual sample and store its quantized value back to the buffer. If the service counter value exceeds a maximum; i.e., too many residual samples in the buffer are not quantized, it is possible for the 8086 CPU to output a residual sample which has not been quantized. Such an error condition will cause the CPU to stop processing indefinitely and flash the double HEX digit LED display (alternating between "00" and "FF"). In the new firmware, such an error condition will increment the most significant digit of the HEX display and reinitialize the CPU such that processing may continue. The existing I/O buffer pointer errors cause the CPU to reset the pointer values and increment the HEX display. As the result of the modification, only the least significant HEX digit is incremented. Overflow error handling remains the same as before and will output "FF" to the HEX display. Under normal operation, the above error conditions will never occur. However, static discharge or edge connector contact problems can occasionally cause a RAM error or noise on the interrupt lines which will in turn produce counter and/or pointer errors.

SECTION 3

3. DIAGNOSTIC RHYME TESTS

3.1 Introduction

The Diagnostic Rhyme Test (DRT) is a method for measuring speech intelligibility. The DRT not only measures total intelligibility but also gives diagnostic scores which measure how well a system preserves six phonetic features. In this test, listeners have to recognize the correct word spoken out of a pair of words given on the answer sheet. These paired words differ only in the presence or absence of one of the six phonetic attributes in the initial consonent.

Diagnostic rhyme tests were performed to measure the effects of RELP coding at 4.8 KBPS and 9.6 KBPS on speech intelligibility. These tests were performed with and without the presence of armoured personnel carrier or helicopter noise.

3.2 Acoustic Signal to Noise Ratio Measurement

Acoustic Signal to Noise Ratio (ASNR) is defined as the ratio of the average peak power of all the words recorded on the DRT tape (four tests) to the average power of the background noise in dB. Speech is first band passed (70-3500 Hz) and the peak power of a spoken word is measured using a fast responding true RMS (Root Mean Square) to DC converter circuit as shown in Figure 3.2-1 and 3.2-2. The measured peak RMS voltages of all words on the DRT tape are listed in Appendix C. The average peak power is obtained through summing the square of all peak RMS voltages and dividing by the total number of words. The type of background noises used are those of M113 armoured personnel carrier and the CH147 Chinook helicopter recorded on CRC supplied tape. The power of these continuous noises are also measured with the true RMS to DC converter. The power levels were observed to fluctuate slightly with time (about 2 dB peak-to-peak). Therefore, five short-term power measurements were made at different intervals on each noise tape. The average noise power was computed in a similar fashion as the average word power.

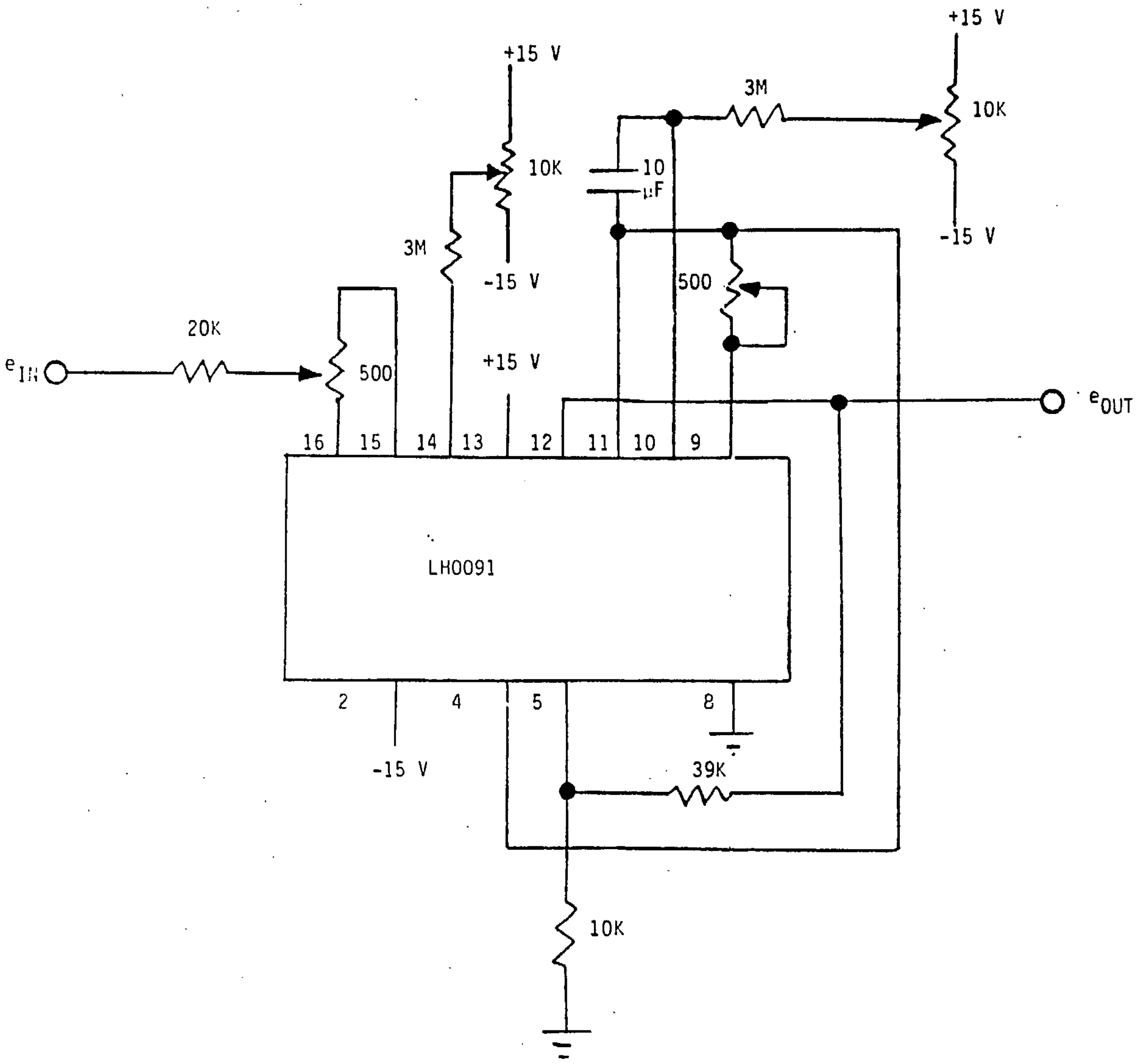


FIGURE 3.2-1 SCHEMATIC OF TRUE RMS TO DC CONVERTER

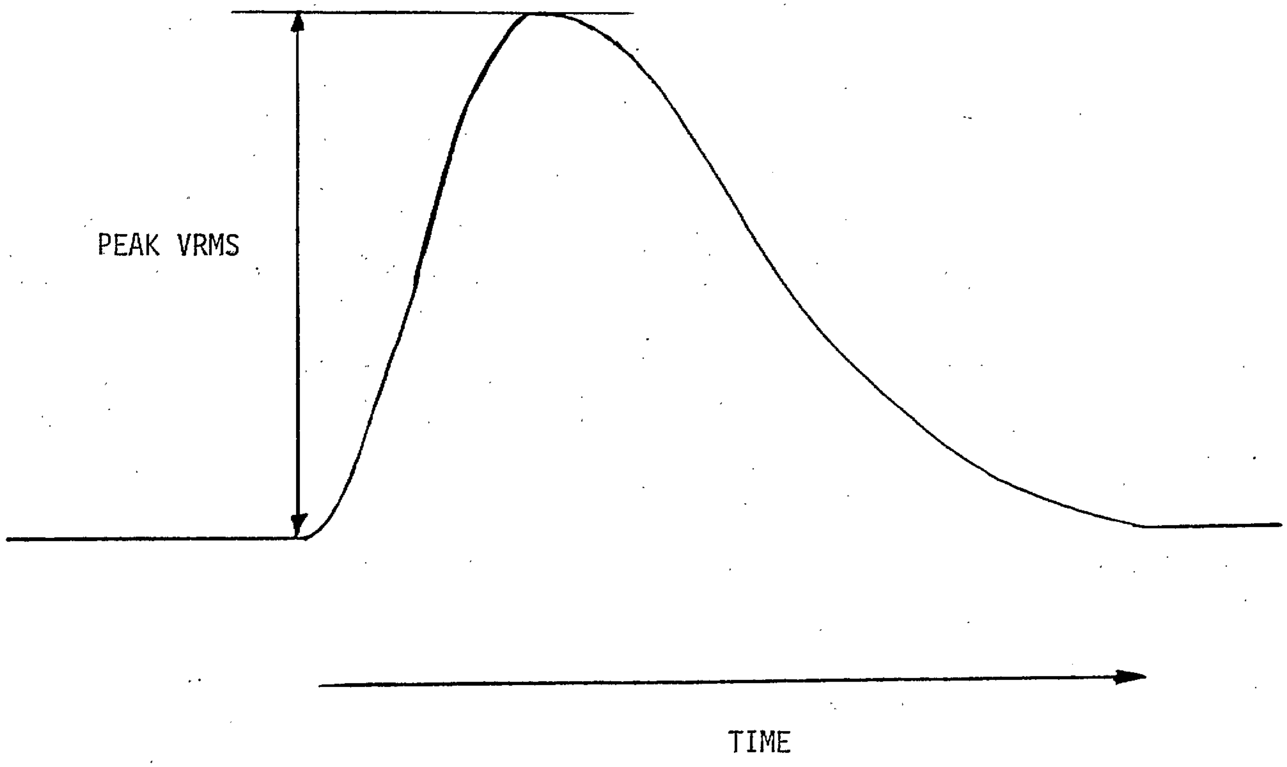


FIGURE 3.2-2 TRUE RMS WAVEFORM OF A TYPICAL SPOKEN WORD OBSERVED ON AN OSCILLOSCOPE

3.3 Preparation of the DRT Tapes

The noise tape and the DRT tape were played by two cassette recorders (Hitachi DE95). The outputs of the recorders were fed to a summing amplifier (Realistic 32-1200) whose output was recorded on a third cassette recorder (Technics M240X). The average noise power bandlimited to 700-3500 Hz was adjusted to 10 dB below the average peak word power (166 mV RMS). Two recordings being DRT in the presence of helicopter noise and DRT in the presence of armoured personnel carrier noise were produced. These noisy DRT recordings and the original high quality DRT recording were played through the RELP codec real-time simulator whose output was again recorded. A total of 40 DRTs with different coding rates and background noise types were produced and the order of presentation was randomized according to Table 3.3-1. These DRTs were then broken down into five separate recordings, each consisting of eight different DRTs. One recording was to be presented to the test subjects each day to minimize the effects of fatigue, and possibly, adaptation.

3.4 Test Procedure

The five recordings were presented to four test subjects over a period of five days. The tests were performed in a quiet room, using a Hitachi DE95 tape deck and a Vector Research model VR-5000 receiver which drives a KEF speaker. Two male subjects and two female subjects with English as their first language were selected. None of them were familiar with RELP coded speech nor with diagnostic rhyme testing.

TABLE 3.3-1 SCHEDULE OF TESTS

DAY	PART	TEST	CODING	BACKGROUND NOISE
1	1	1	RELP 9.6	Helicopter
1	2	2	RELP 4.8	Helicopter
1	3	3	RELP 4.8	None
1	4	4	RELP 4.8	Helicopter
1	5	3	None	Armoured Car
1	6	2	RELP 4.8	Armoured Car
1	7	1	RELP 9.6	None
1	8	4	PERT 4.8	None
2	1	3	RELP 4.8	Helicopter
2	2	2	RELP 4.8	None
2	3	1	None	None
2	4	4	None	Armoured Car
2	5	3	RELP 4.8	Armoured Car
2	6	4	RELP 9.6	Armoured Car
2	7	1	None	Helicopter
2	8	2	PERT 4.8	None
3	1	2	None	Helicopter
3	2	1	RELP 4.8	Armoured Car
3	3	4	RELP 4.8	Helicopter
3	4	3	RELP 9.6	None
3	5	2	None	None
3	6	1	RELP 4.8	None
3	7	4	RELP 9.6	Helicopter
3	8	3	RELP 9.6	Armoured Car
4	1	4	None	None
4	2	3	RELP 4.8	None
4	3	2	RELP 9.6	Helicopter
4	4	1	None	Armoured Car
4	5	2	RELP 9.6	None
4	6	3	PERT 9.6	None
4	7	4	None	Helicopter
4	8	1	RELP 9.6	Armoured Car
5	1	3	None	None
5	2	4	RELP 9.6	None
5	3	1	PERT 9.6	None
5	4	2	None	Armoured Car
5	5	3	RELP 9.6	Helicopter
5	6	2	RELP 9.6	Armoured Car
5	7	1	RELP 4.8	Helicopter
5	8	4	RELP 4.8	None

3.5 Analysis of Test Results

3.5.1 Introduction

The answers given for all the DRTs were entered into a computer. The use of a computer allowed fast, reliable marking of the tests and permitted summaries of the scores based on the person taking the test, coding, background noise and attribute. All scores are given in unadjusted percent correct to allow comparison with INRS results.

$$Pc = 100 * (Total - Wrong)/(Total)$$

The adjusted percent correct score may be obtained from the unadjusted score as follows:

$$Pa = 2 * Pc - 100$$

3.5.2 DRT Scores and Observations

A summary of the results of the DRTs is given in Table 3.5.2-1. The DRT scores over the five days of testing showed no trend which would indicate that the subjects were learning the tests or becoming better at understanding RELP speech.

Table 3.5.2-2 gives a summary of the DRT results for the various coding methods and background noises. RELP coding at 9.6 Kbps results in a 3 percent drop in DRT score over raw speech. Dropping the data rate to 4.8 Kbps decreases the DRT scores by a further 3 percent. The perturbation algorithm does not perform as well as RELP. The RELP scores are about 2 to 3 percent higher than the PERT scores. The perturbation algorithm perturbs the low frequencies as well as the high frequencies in the residual. This may have caused the lower score in the graveness attribute which is mostly responsible for the score differences. In the presence of armoured personnel carrier noise, 9.6 Kbps RELP coding is only marginally affected (-2%) while 4.8 Kbps RELP has a 5 percent reduction in score (Figure 3.5.2-2). Figure 3.5.2-3 shows that RELP coded speech at 4.8 Kbps is just about as intelligible as that at 9.6 Kbps in the presence of helicopter noise. This would suggest that 4.8 Kbps RELP coding may be adequate for noisy environment applications in spite of the fact that 9.6 Kbps RELP coding is judged to be more acceptable.

Of the six speech attributes of speech measured in the DRTs, sustention and graveness have the lowest scores for the various coding and noise conditions. RELP coding has the greatest difficulty with the sustention attribute, often causing no sustention attribute to be heard in the coded speech when it was actually present in the original speech. This causes words like "sheet" to be coded like "cheat". The bias added to speech is summarized in Table 3.5.2-3.

Since the words in the DRTs varied by greater than 12 dB in amplitude, an attempt was made to see if this affected the scores. Table 3.5.2-4 shows that there is less than one dB difference among the average powers of words in each attribute. This suggests that it is unlikely that the difference in scores among attributes is caused by different signal to noise ratios. There is no trend of decreasing score with increasing noise levels, Table 3.5.2-5. This is likely due to the fact that the difference in difficulty among the word groups has a greater effect on the score.

Table 3.5.2-6 shows the standard errors of the DRT scores among the four test subjects. In all cases, the standard errors are less than about 3 percent. This may suggest that there is no great variation in test scores among subjects.

TABLE 3.5.2-1 DRT SCORES BY PERSON AND TIME OF TEST

PERCENT CORRECT DIAGNOSTIC RHYME SCORES
Score as a function of person and time of test

DAY	PART	TEST	CODING	BACKGROUND NOISE	ALVA	LYNNE	PETER	WOLF	AVERAGE
1	1	1	RELP 9.6	Helicopter	85	94	88	88	89
1	2	2	RELP 4.8	Helicopter	83	81	83	92	85
1	3	3	RELP 4.8	None	90	90	90	90	90
1	4	4	RELP 4.8	Helicopter	77	85	81	92	84
1	5	3	None	Armoured Car	96	92	98	96	95
1	6	2	RELP 4.8	Armoured Car	85	81	85	90	85
1	7	1	RELP 9.6	None	96	96	98	98	97
1	8	4	PERT 4.8	None	94	88	92	94	92
2	1	3	RELP 4.8	Helicopter	88	79	81	90	84
2	2	2	RELP 4.8	None	94	92	94	100	95
2	3	1	None	None	100	100	100	100	100
2	4	4	None	Armoured Car	98	98	98	96	97
2	5	3	RELP 4.8	Armoured Car	85	90	83	94	88
2	6	4	RELP 9.6	Armoured Car	96	96	90	96	94
2	7	1	None	Helicopter	100	98	98	100	99
2	8	2	PERT 4.8	None	92	90	88	90	90
3	1	2	None	Helicopter	94	98	100	98	97
3	2	1	RELP 4.8	Armoured Car	90	90	94	94	92
3	3	4	RELP 4.8	Helicopter	90	85	90	81	86
3	4	3	RELP 9.6	None	96	92	96	94	94
3	5	2	None	None	98	100	100	100	99
3	6	1	RELP 4.8	None	94	92	94	94	93
3	7	4	RELP 9.6	Helicopter	92	79	94	88	88
3	8	3	RELP 9.6	Armoured Car	81	90	94	96	90
4	1	4	None	None	100	100	98	100	99
4	2	3	RELP 4.8	None	85	88	92	94	90
4	3	2	RELP 9.6	Helicopter	88	88	88	92	89
4	4	1	None	Armoured Car	100	96	100	98	98
4	5	2	RELP 9.6	None	96	96	96	98	96
4	6	3	PERT 9.6	None	90	90	94	96	92
4	7	4	None	Helicopter	100	100	96	94	97
4	8	1	RELP 9.6	Armoured Car	100	98	100	96	98
5	1	3	None	None	100	96	100	98	98
5	2	4	RELP 9.6	None	98	94	96	98	96
5	3	1	PERT 9.6	None	94	90	96	98	94
5	4	2	None	Armoured Car	96	100	98	100	98
5	5	3	RELP 9.6	Helicopter	88	83	88	90	87
5	6	2	RELP 9.6	Armoured Car	92	96	92	92	93
5	7	1	RELP 4.8	Helicopter	96	88	94	96	93
5	8	4	RELP 4.8	None	94	100	96	94	96

TABLE 3.5.2-2 DRT SCORES BY ATTRIBUTE

PERCENT CORRECT DIAGNOSTIC RHYME TEST SCORES
Score as a function of attribute

CODING	BACKGROUND NOISE	VOICING	NASALITY	SUSTENTION	SIBILATION	GRAVENESS	COMPACTNESS	TOTAL
None	None	99	100	99	100	98	100	99
None	Armoured Car	98	100	94	100	95	98	97
None	Helicopter	100	100	96	100	92	100	98
RELP 4.8	None	95	100	83	93	86	99	93
RELP 4.8	Armoured Car	91	96	82	83	83	95	88
RELP 4.8	Helicopter	93	95	68	90	81	93	87
RELP 9.6	None	97	100	91	95	94	99	96
RELP 9.6	Armoured Car	98	100	83	93	92	97	94
RELP 9.6	Helicopter	95	97	71	86	84	95	88
PERT 4.8	None	97	98	88	91	77	94	91
PERT 9.6	None	98	100	86	91	84	100	93

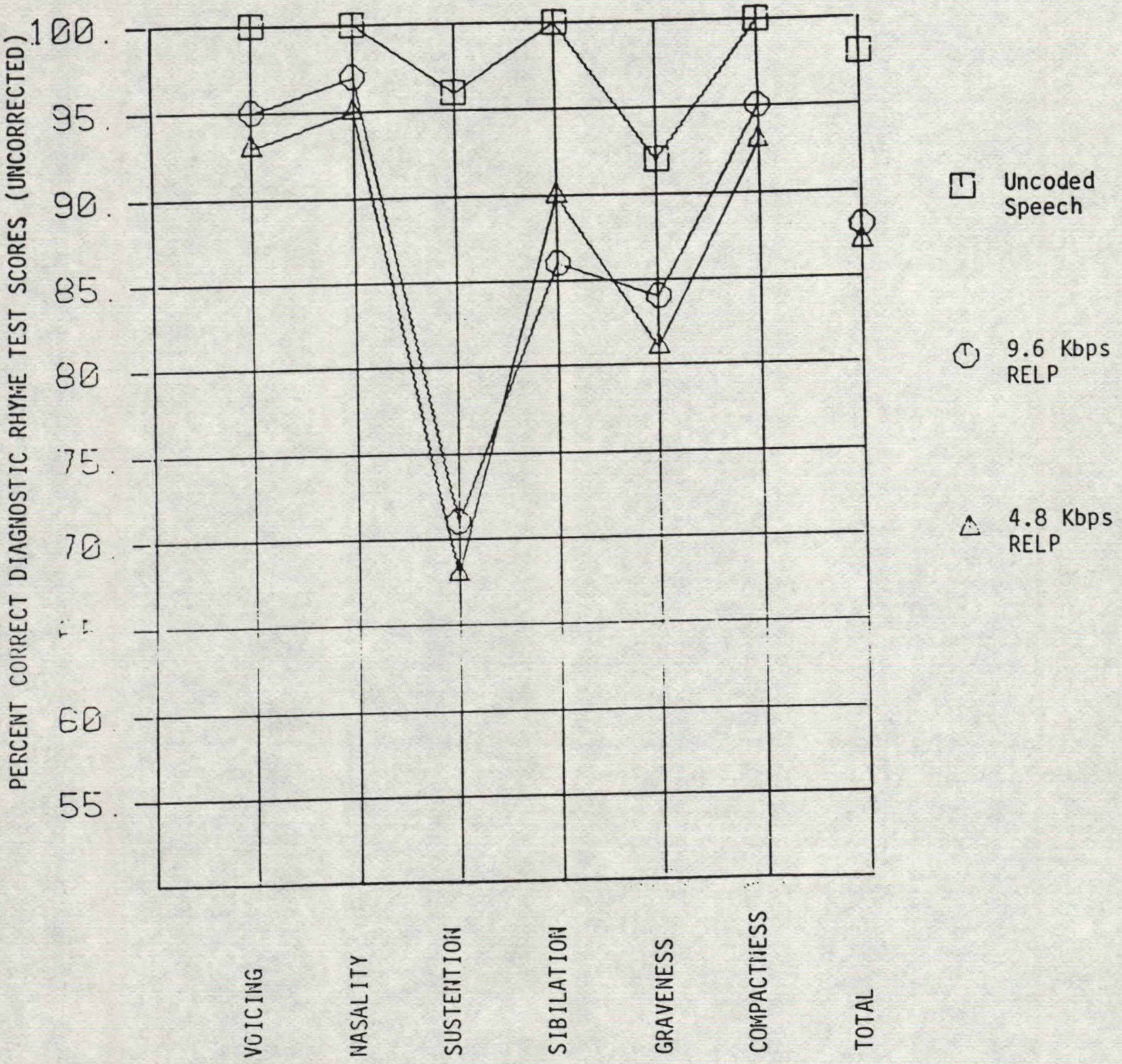


FIGURE 3.5.2-3 DIAGNOSTIC RHYME TEST RESULTS FOR RELP CODING WITH HELICOPTER BACKGROUND NOISE

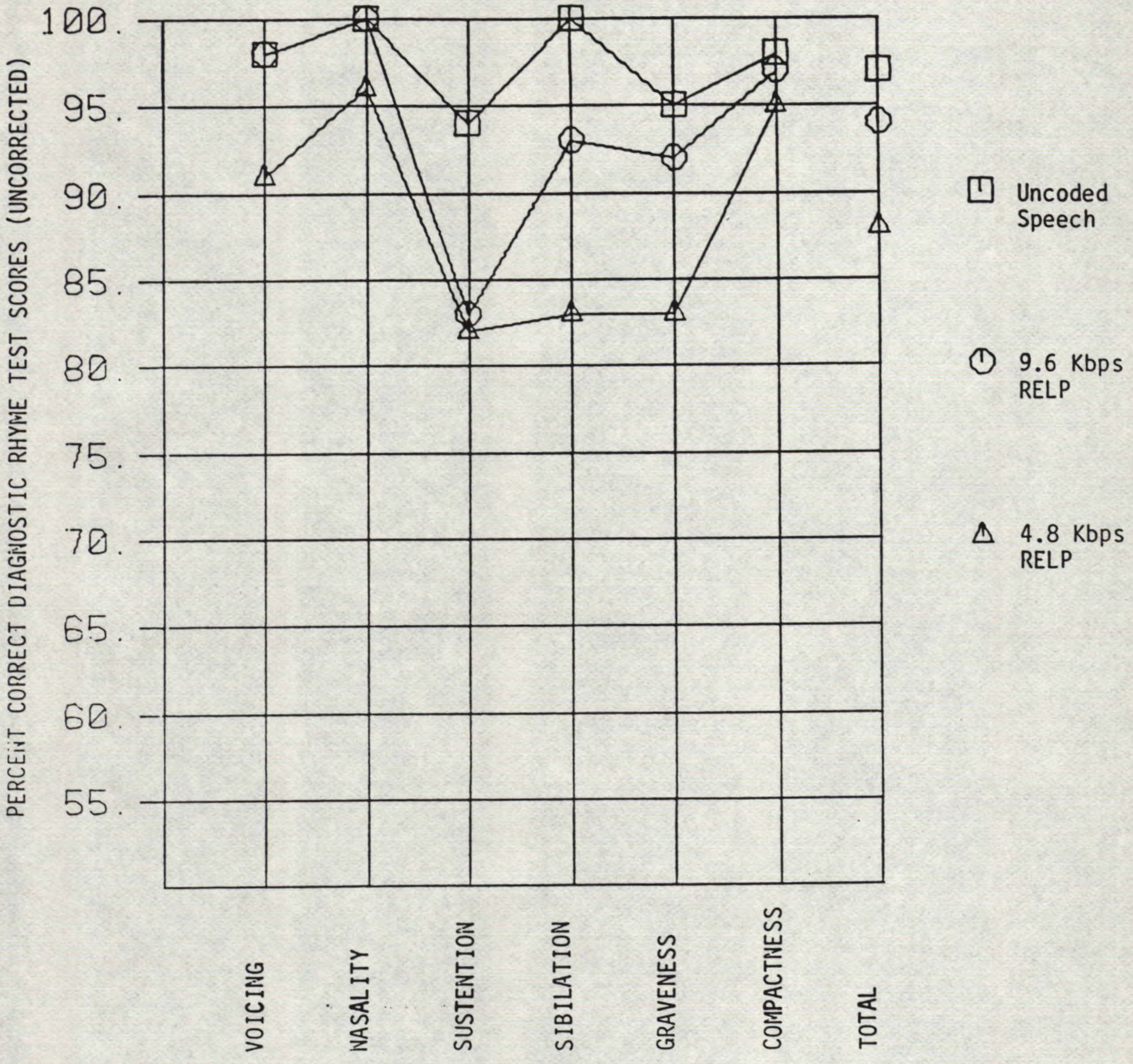


FIGURE 3.5.2-2 DIAGNOSTIC RHYME TEST RESULTS FOR RELP CODING WITH ARMOURED PERSONEL CARRIER NOISE

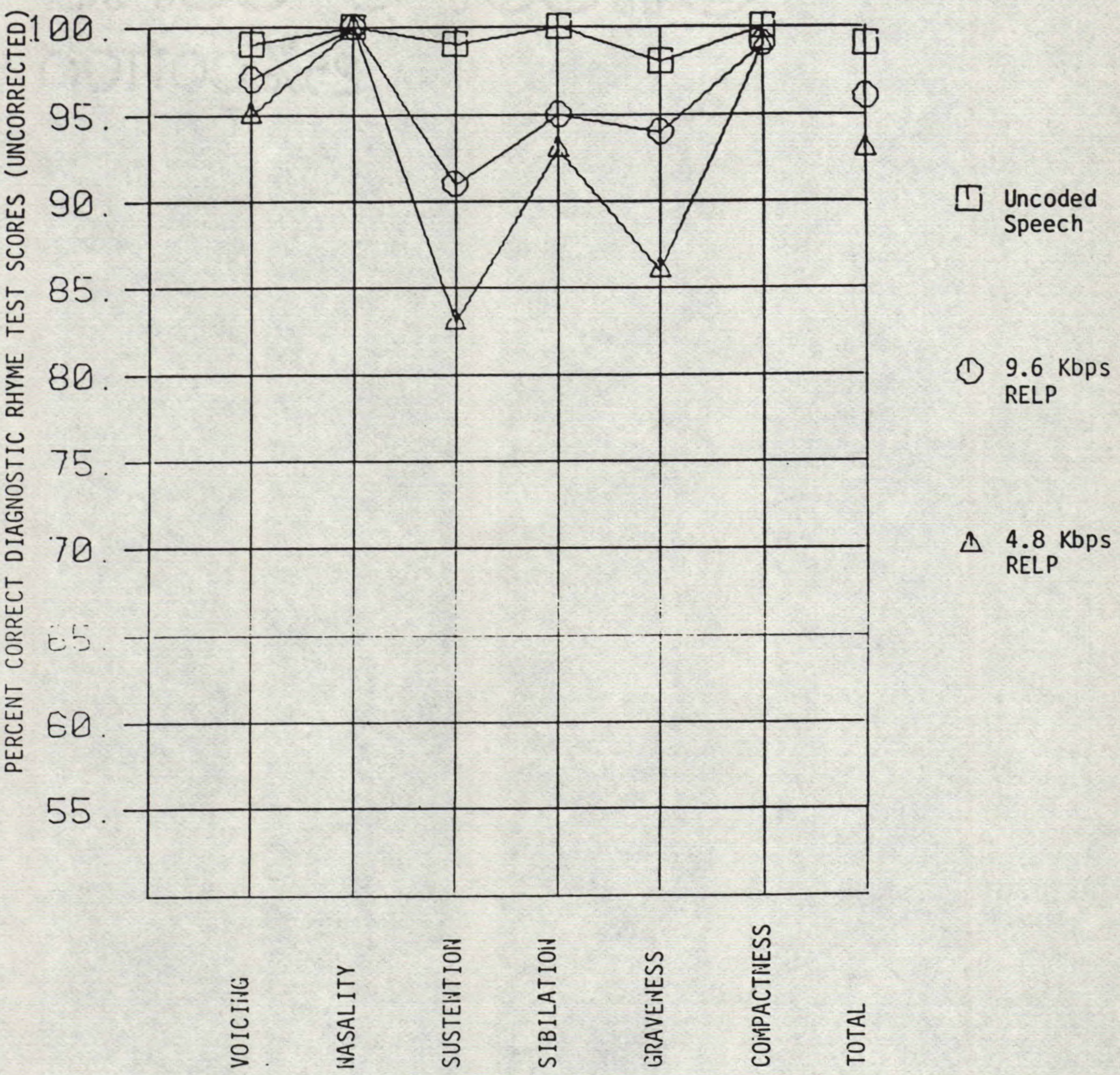


FIGURE 3.5.2-1 DIAGNOSTIC RHYMES TEST RESULTS FOR RELP CODING WITHOUT NOISE

TABLE 3.5.2-3 BIAS IN DRT SCORES

CODING	BACKGROUND NOISE	VOICING	NASALITY	SUSTENTION	SIBILATION	GRAVENESS	COMPACTNESS
None	None	2	0	-2	0	-5	0
None	Armoured Car	5	0	-3	0	0	5
None	Helicopter	0	0	-1	0	0	0
RELP 4.8	None	10	0	-18	-8	-6	2
RELP 4.8	Armoured Car	18	-5	-17	-13	0	-3
RELP 4.8	Helicopter	6	-7	-18	-8	5	0
RELP 9.6	None	7	0	-16	-6	3	2
RELP 9.6	Armoured Car	4	0	-28	-15	0	0
RELP 9.6	Helicopter	3	-3	-5	-9	4	-3
PERT 4.8	None	4	6	-19	-10	-13	-4
PERT 9.6	None	-4	0	-21	-9	-6	0

NOTE: The above scores are the difference between the percent correct with the attribute present and the percent correct with the attribute absent. A positive number indicates that the attribute is heard when it is not present.

TABLE 3.5.2-4 AVERAGE WORD POWER VERSUS ATTRIBUTE

ATTRIBUTE	AVERAGE RMS VOLTAGE	POWER DEVIATION (dB)
VOICING	163.94	-0.15
NASALITY	161.29	-0.29
SUSTENTION	167.22	0.03
SIBILATION	168.92	0.11
GRAVENESS	160.09	-0.35
COMPACTNESS	178.19	0.58
OVERALL AVERAGE	166.72	0.00

TABLE 3.5.2-5 DRT SCORES AS A FUNCTION OF SIGNAL TO NOISE RATIO

CODING	BACKGROUND NOISE	1 dB	4 dB	7 dB	10 dB	13 dB	16 dB	19 dB	NO NOISE
None	None								99
None	Armoured Car		100	97	97	97	?100		
None	Helicopter		100	94	99	98	?100		
RELP 4.8	None								93
RELP 4.8	Armoured Car		93	84	89	89			
RELP 4.8	Helicopter		92	78	90	88	? 94		
RELP 9.6	None								96
RELP 9.6	Armoured Car		98	91	96	94	? 92		
RELP 9.6	Helicopter		89	85	90	87	?100		
PERT 4.8	None								91
PERT 9.6	None								93
PERT 9.6	None								93

NOTE: No entry in table if less than 10 sample points. A '?' precedes scores with less than 48 samples.

TABLE 3.5.2-6 DRT SCORES AND STANDARD ERRORS

CODING	BACKGROUND NOISE	AVERAGE SCORE (%)	STANDARD ERROR (%)
None	None	99.35	0.23
None	Armoured Car	97.40	0.73
None	Helicopter	97.91	0.46
RELP 4.8	None	92.60	1.08
RELP 4.8	Armoured Car	88.38	2.31
RELP 4.8	Helicopter	86.56	2.25
RELP 9.6	None	95.96	1.00
RELP 9.6	Armoured Car	93.50	1.71
RELP 9.6	Helicopter	88.02	1.27
PERT 4.8	None	90.63	1.65
PERT 9.6	None	93.23	2.81

3.5.3 Comparison With The INRS Results

Nakatsui et al [4] gives results of DRTs performed on an earlier version of the RELP Algorithm developed at INRS. The INRS RELP algorithm was simulated on a PDP-11 minicomputer. It has several differences from the current hardware implementation, such as the use of floating point instead of integer arithmetic; interpolation of the predictor coefficients between frames, to avoid abrupt amplitude changes at the frame boundary; use of two residual gain parameters and several other small differences. It is not known if the same speaker was used for the INRS DRTs. Despite these differences, the DRT scores for the two implementations were remarkably similar with three exceptions.

The hardware implementation has lower scores on the sustention attribute (Figure 3.5.3-1). This could be due to the changes in the algorithm such as the removal of predictor coefficient interpolation, or the use of a different speaker on DRT recording.

The overall scores in the presence of noise were higher for the hardware implementation than the INRS software simulation (Figure 3.5.3-2). One explanation for the difference is attributed to the fact that white noise was used by INRS instead of periodic noise used in this study. The high peak-to-average power ratio in white noise could have impaired normal hearing to a greater extent than the periodic noises of equal power levels. Such a phenomenon manifested itself in the score discrepancy between helicopter noise and armoured personnel carrier noise. The DRT scores in the presence of helicopter noise were generally lower as a result of a higher peak-to-average noise power ratio. Subjectively, noise with a higher peak-to-average power ratio sounds more abrupt and annoying.

The scores for the uncoded speech were higher in this report than those described in INRS (Figures 3.5.3-3 and 3.5.3-4). The uncoded speech data base used by INRS was band-limited to 3.2 KHz but not in this study. Another reason for the inconsistency could be due to speaker dependence.

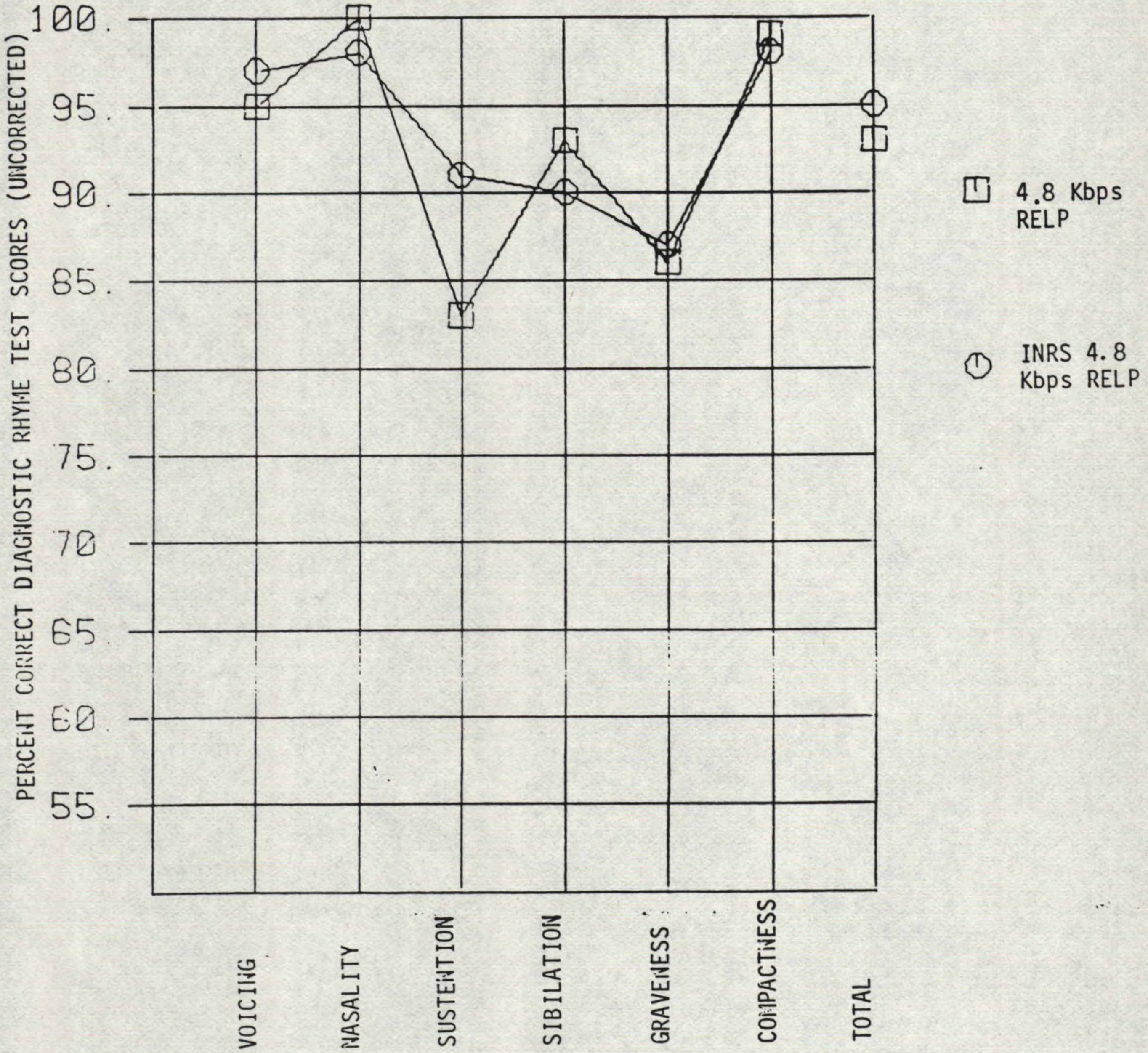


FIGURE 3.5.3-1 COMPARISON WITH INRS DIAGNOSTIC RHYME TEST SCORES FOR 4.8 KBPS RELP

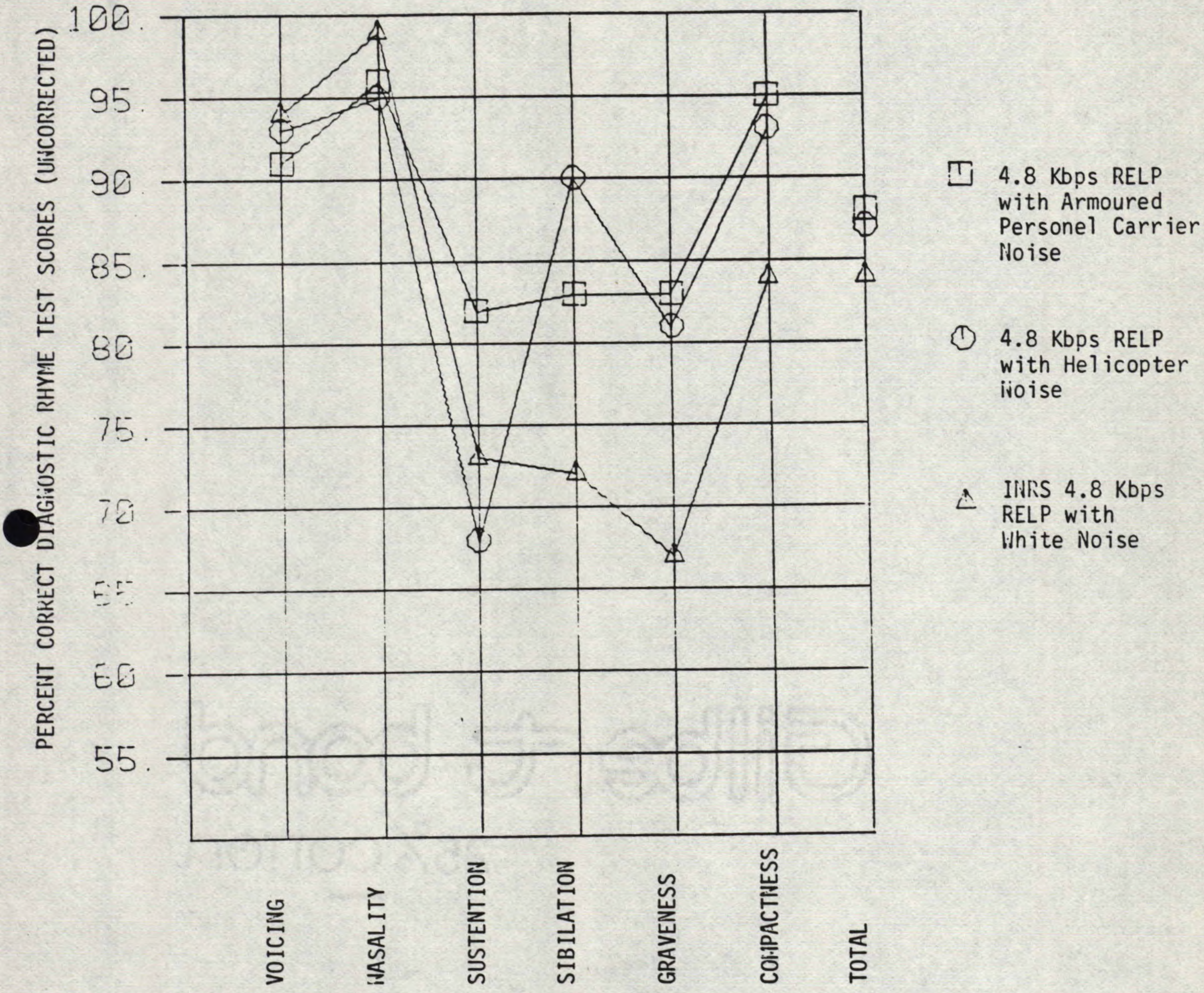


FIGURE 3.5.3-2 COMPARISON WITH INRS DIAGNOSTIC RHYME TEST SCORES FOR 4.8 KBPS RELP WITH 10 dB SIGNAL TO NOISE RATIO

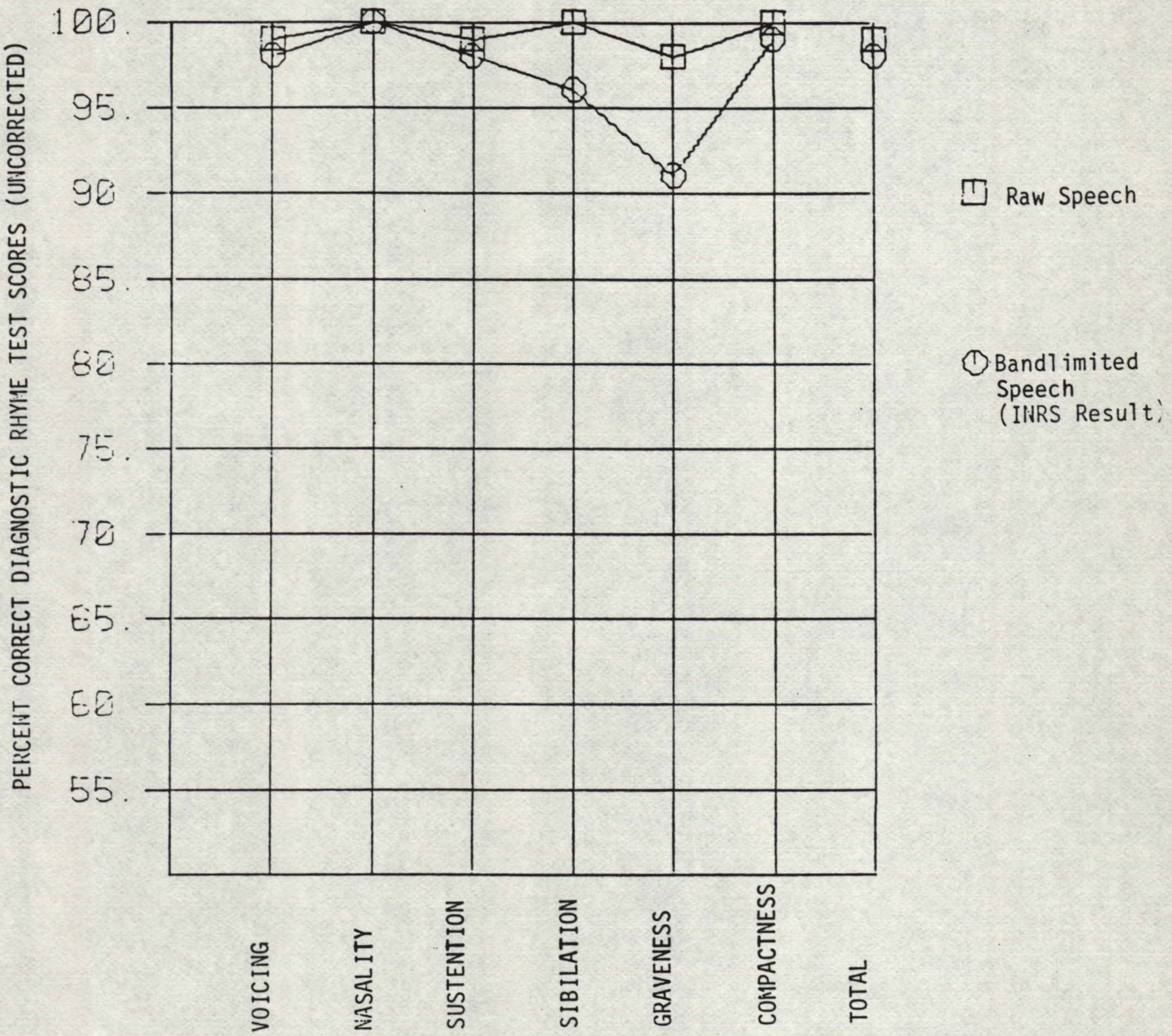


FIGURE 3.5.3-3 DRT SCORES FOR UNCODED SPEECH WITHOUT NOISE

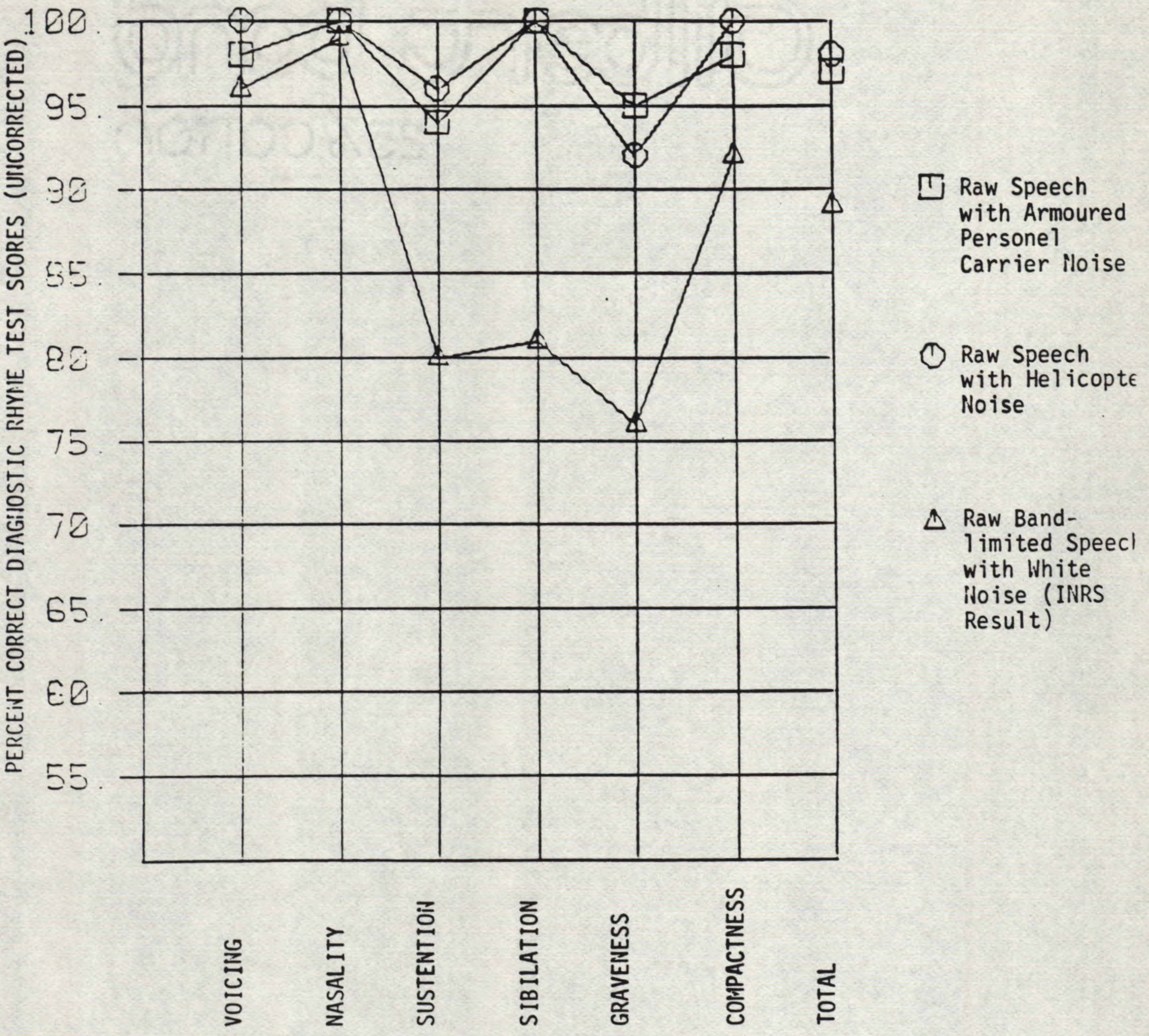


FIGURE 3.5.3-4 DRT SCORES FOR UNCODED SPEECH AT 10 dB SIGNAL TO NOISE RATIO

SECTION 4

4. CONCLUSIONS

Diagnostic Rhyme Tests have been performed on the RELP coded speech using the RELP codec real-time hardware simulator. RELP coding at 4.8 Kbps and 9.6 Kbps together with an alternate full-band residual regeneration method have been simulated. The effects of helicopter noise and armoured personnel carrier noise on RELP coded speech are also measured.

Intelligibility as measured by the Diagnostic Rhyme Tests is 96% for RELP coded speech at 9.6 Kbps and 93% for RELP coded speech at 4.8 Kbps. These represent, respectively, a three and six percent degradation over uncoded speech. Further, 9.6 Kbps RELP coding is only degraded marginally by two percent in the presence of armoured personnel carrier noise while 4.8 Kbps RELP coding is degraded by as much as five percent. RELP coding at 9.6 Kbps and 4.8 Kbps have almost equal intelligibility in the presence of helicopter noise. This may suggest that 4.8 Kbps RELP coding is adequate for applications in noisy environment.

The new full-band residual regeneration method (PERT) scores two to three percent lower than the rectification method proposed by INRS although speech quality is judged to be slightly more acceptable. It is therefore possible to tradeoff minor performance degradation for a seventy percent reduction in the amount of synthesizer processing.

In spite of the various simplifications in the hardware simulation, results are remarkably similar to the INRS software simulation. A 1.5 percent reduction in DRT score has been observed on the hardware RELP simulation at 4.8 Kbps.

SECTION 5

5. REFERENCES

[1] Higgins, A. and R. Viswanathan, "New High-Frequency Regeneration Techniques for Voice-Excited Speech Coders", Proceedings of the 98th Meeting of the Acoustic Society of America, Salt Lake City, Utah, November 26-30, 1979.

[2] Crochiere, R.E., "A Mid-Rise/Mid-Tread Quantizer Switch for Improved Idle-Channel Performance in Adaptive Coders", B.S.T.J, October, 1978, pp. 2953-2955.

[3] Jayant, N.S., "Adaptive Quantization with a One-Word Memory", B.S.T.J, September, 1973, pp. 1118-1144.

[4] Nakatsui, M., D. Stevenson and P. Mermelstein, "Subjective Evaluation of a 4.8 KBPS Residual-Excited Linear Prediction Coder", A Manuscript prepared for IEEE Transactions on Communications, 1981.

APPENDIX A

APPENDIX A

SYNTHESIS SIGNAL PROCESSOR FIRMWARE LISTINGS

APPENDIX A

SYNTHESIS SIGNAL PROCESSOR FIRMWARE LISTINGS

A1. RESIDUAL REGENERATION THROUGH INTERPOLATION OF LOW PASS COMPONENT BY ZERO AMPLITUDE INSERTION, AND SAMPLE POSITION PERTURBATION OF HIGH FREQUENCY COMPONENT

```
00000 TITLE SYNTHESIS PROCESSOR ASSEMBLER SOURCE FILE. SEPT 10,1982
00000       LIST X
00000 ;**************************************************************
00000 ;*        PROJECT:    472A, RELP CODEC                        *
00000 ;*        BOARD:      SYNTHESIS PROCESSOR                     *
00000 ;*        DEVICE:     6 MICROCODE PROMS, 3636 or 3628         *
00000 ;*        LOCATION:   G34,P1,P15,P29,P43,P57                  *
00000 ;*        PROG.DEV#:  60-0690  -  60-0695                     *
00000 ;*        ASSEMBLER:  META, April 82                          *
00000 ;*        FILE:       SYNTHESIS.ASM                           *
00000 ;*        REVISION:   SEPT 10,1982                            *
00000 ;*        UPDATE TABLE:                                       *
00000 ;*             SEPT 9:   Initial folding exp, hopeful         *
00000 ;*                       Higgins method implemented.          *
00000 ;*             AUG 3:    Aliasing high freq regeneration      *
00000 ;*             JULY 7:   Each filter assigned gain in PROM    *
00000 ;*             JULY 2:   More precision maintained.           *
00000 ;*             JUNE 28:  Filter Gains Adjusted.               *
00000 ;*             JUNE 14:  Mods made to use subtract inst corr  *
00000 ;*             APRIL 23 - MAY 23: Initial progam entry JM     *
00000 ;*                                                            *
00000 ;**************************************************************
```

```
00000  ;**********************************************************
00000  ;*            TABLE OF CONTENTS                           *
00000  ;*                                                        *
00000  ;*     PAGE      DESCRIPTION                              *
00000  ;*    ----------------------------------------------      *
00000  ;*      1        File Header and Update Table             *
00000  ;*      2        Table of Contents                        *
00000  ;*      3        Filter size specifications               *
00000  ;*      4        Ring Buffer memory map                   *
00000  ;*      5        Scratch Pad memory map                   *
00000  ;*      6        Coefficient From memory map              *
00000  ;*      7        Input/Output definitions                 *
00000  ;*      8        PROCEDURE start_up;                      *
00000  ;*      9        PROCEDURE main;                          *
00000  ;*     14        PROCEDURE operations_done_EVERY_sample   *
00000  ;*     17        PROCEDURE Quadrature_Mirror_Filter_High  *
00000  ;*     20        PROCEDURE Quadrature_Mirror_Filter_Low   *
00000  ;*     23        PROCEDURE Low_Pass_Filter                *
00000  ;*     25        PROCEDURE Double_Difference_Filter       *
00000  ;*     26        PROCEDURE High_Pass_Filter               *
00000  ;*     28        PROCEDURE Predict                        *
00000  ;*     29        PROCEDURE CoPY_PREDictor_Coefficients    *
00000  ;*     30        PROCEDURE DE-EMphasize                   *
00000  ;*     31        Cross Reference Table                    *
00000  ;**********************************************************
```

```
00000 ;*************************************************************
00000 ;*           FILTER SIZE SPECIFICATIONS                       *
00000 ;*                                                            *
00000 ;*************************************************************
00000 J:      EQU    9      ; Predictor filter order.
00000 QMFSZ:  EQU   36      ; Quadrature Mirror Filter Size
00000 LPFSZ:  EQU   31      ; Low pass filter size
00000 PREDSZ: EQU    9      ; Predictor Filter Size
00000 DEMSZ:  EQU    2      ; Deemphasis Filter Size
00000 DLY:    EQU   82      ; Delay so predictors line up with frame
00000 ; The following constants define filter sizes in the rectification
00000 ; RELP implementation.  They are used here only to specify the location
00000 ; the filters in the coefficient PROM, which is the same for both
00000 ; the perturbation and rectication implementations.
00000 PDFSZ:  EQU    3
00000 ABSZ:   EQU    1
00000 HPFSZ:  EQU   33
```

```
  68   00000  ;*********************************************************
  69   00000  ;*           RING BUFFER MEMORY MAP                      *
  70   00000  ;*                                                       *
  71   00000  ;*********************************************************
  72   00000  QMFBS:  EQU   10(0):% ; Base of quadrature mirror filter
  73   00000  DLYBS:  EQU   10(QMFBS+5*QMFSZ+10):% ; Compensating delay for PERT
  74   00000  LPFBS:  EQU   10(DLYBS+D#3):%        ; Low pass filter base
  75   00000  SUMBS:  EQU   10(LPFBS+LPFSZ+D#1):%  ; summing node for HPF and LPF path
  76   00000  PREDBS: EQU   10(SUMBS+DLY):%        ; Predictor base(101 IS MATCHING DE
  77   00000  PREDOUT:EQU   10(PREDBS+PREDSZ)   ; End of predictor
  78   00000  DEMBS:  EQU   10(PREDBS+PREDSZ):% ; Deemphasis base
  79   00000  PERTBS: EQU   10(DEMBS+D#5):%     ; Base of compensating delay buffer
  80   00000  HPFBS:  EQU   10(PERTBS+D#3):%    ; End of compensating delay
  81   00000  HFROUT: EQU   10(HPFBS+LPFSZ):%   ; Output of High Freq Regeneration
  82   00000  S:      EQU   10(DEMBS+2):%       ; Synthesized speech location
  83   00000  LEAST:  EQU   10(D#1023):%        ; Temp location used in double prec
  84   00000                                  ; arithmatic
  85   00000  MOST:   EQU   10(D#1022):%        ; As above
```

```
00000 ;*****************************************************
00000 ;*    SCRATCH PAD MEMORY MAP                           *
00000 ;*****************************************************
00000 P0:     EQU   8(0):%               ; base of predictor storage
00000 PTMP0:  EQU   8(P0+J):%            ; Base of temporary predictor storage
00000 P:      EQU   8(PTMP0+J):%         ; Last predictor coef. input
00000 FE:     EQU   8(P+1):%             ; FE(H,L) storage
00000 LSTP:   EQU   8(FE+1):%            ; Last P used
00000 LSTFEL: EQU   8(LSTP+1):%          ; Last FEL input
00000 LSTFEH: EQU   8(LSTFEL+1):%        ; Last FEH input
00000 LSTFE:  EQU   8(LSTFEH+1):%        ; Last FE calculated
00000 TEMP:   EQU   8(LSTFE+1):%         ; Temporary storage location
```

```
 100   00000  ;**************************************************************
 101   00000  ;*      COEFFICIENT PROM MEMORY MAP                             *
 102   00000  ;**************************************************************
 103   00000  CON0:    EQU   8(0);%                ; constant zero
 104   00000  CON1:    EQU   8(1);%                ; constant one
 105   00000  CONM1:   EQU   8(2);%                ; constant minus one
 106   00000  A:       EQU   8(3);%                ; Pre-emphasis constant/2
 107   00000  CLPF:    EQU   8(A+1);%              ; Base of low pass filter coef
 108   00000  CQMF:    EQU   8(CLPF+LPFSZ);%       ; Base of quadrature mirror filter
 109   00000  CDDF:    EQU   8(CQMF+QMFSZ);%       ; Base oF double difference filter
 110   00000  CHPF:    EQU   8(CDDF+DDFSZ);%       ; Base of high pass filter
 111   00000  PREDSCL:EQU    8(CHPF+HPFSZ);%       ; Scale factor for predictor
 112   00000   CONQRTR: EQU 8(PREDSCL);%           ; Scale facter is 1/4
 113   00000  CON2:    EQU   8(PREDSCL+1);%
 114   00000  CON4:    EQU   8(CON2+1);%
 115   00000  CON8:    EQU   8(CON4+1);%
 116   00000  CON128:  EQU   8(CON8+1);%
 117   00000  GNINPUT:EQU    8(CON128+1);%
 118   00000  GNQMF:   EQU   8(GNINPUT+1);%        ; Filter gains follow
 119   00000  GNLPF:   EQU   8(GNQMF+1);%
 120   00000  GNDDF:   EQU   8(GNLPF+1);%
 121   00000  GNHPF:   EQU   8(GNDDF+1);%
 122   00000  GNPRED:  EQU   8(GNHPF+1);%
 123   00000  GNOLT:   EQU   8(GNPRED+1);%
 124   00000  THRESH:  EQU   8(CLPF+1);%           ; = 526, USED TO AVOID BURNING
 125   00000                                       ; PROM BEFORE FINDING OPTIMUM VALUE
 127   00000  ;*********************************************************
 128   00000  ;*   INPUT/OUTPUT DEFINITIONS                             *
 129   00000  ;*********************************************************
 130   00000  TEST:   EQU B#01       ; Analog test port address
 131   00000  PPRT:   EQU B#10       ; Predictor input port
 132   00000  FEPRT:  EQU B#11       ; FE input port
 133   00000  SPRT:   EQU B#10       ; Synthesized speech output port
 134   00000  .SENBL: EQU 4H#5       ; S enable test input address

 136   00000  ; Test switch input definitions
 137   00000  TSTFEL: EQU .TEST0                  ; Test point select 0 = FEL
 138   00000  TSTFEH: EQU .TEST1                  ; Test point select 1 = FEH
 139   00000  TSTFE:  EQU .TEST2                  ; Test point select 2 = FE
 140   00000  TSTE:   EQU .TEST3                  ; Test point select 3 = E
 141.  00000  TSTHPF: EQU .TEST4                  ; Test point select 4 = HPF
 142   00000  TSTSUM: EQU .TEST5                  ; Test point select 5 = SUM HPF & E
 143   00000  TSTSPE: EQU .TEST6                  ; Test point select 6 = SPE
 144   00000  TSTS:   EQU .TEST7                  ; Test point select 7 = S
 146   00000  ;*********************************************************
 147   00000  ;*   PROCEDURE start_up;                                  *
 148   00000  ;** This procedure is executed on power up.  It initializes*
 149   00000  ;* the signal processor and waits a sufficient time to    *
 150   00000  ;* allow the rest of the system to start functioning before*
 151   00000  ;* processing.  This prevents noise from being output     *
 152   00000  ;* during the power up sequence.                         }*
 153   00000  ;*                                                        *
 154   00000  ;*   BEGIN                                                *
 155   00000  ;*     Load the Accumulator with zero                     *
 156   00000  ;*     Zero sample output port so no signal output during *
 157   00000  ;*             power up sequence.                         *
 158   00000  ;*     FOR i := 0 TO 4095 DO                              *
 159   00000  ;*       shift data in ring buffer                        *
 160   00000  ;*       Store Accumulator in ring buffer location zero   *
 161   00000  ;*     ENDFOR                                             *
 162   00000  ;*     Start normal processing loop at start_of_frame entry*
 163   00000  ;*   END;                                                 *
 164   00000  ;*                                                        *
 165   00000  ;*********************************************************
 166   00000         ORG     0
 167   00000         FILTER: COEFB,CONO,0:%  ; Zero the Accumulator
              11100000 01010000 00000000 00000000 11100000 01000100
 168   00001         FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 169   00002         OUTPUT  SPRT,SPRT        ; Zero the sample output port
              11100000 01100100 00000110 01000000 11000000 00100011
 170   00003         OUTPUT  TEST,TEST        ; Zero the analog test point
              11100000 01100010 00000110 00100000 11000000 00100011
 171   00004         PUSH    (D#4095):%       ; FOR i := 0 TO 4095 DO
              01000000 11111111 11110110 11111111 11000000 00000000
 172   00005         SHL     ,,,,.SHIFT       ;   shift data in ring buffer
              11100000 01101111 11110110 11111111 11001000 00000000
 173   00006         SPL                      ;   NOP, wait for shift to complete
              11100000 01101111 11110110 11111111 11000000 00000000
 174   00007         STMR    ,0:%             ;   store zero in ring buf
              11100000 00111111 11110000 00000000 11000000 00101010
 175   00008         REPLP                    ; ENDFOR
              10000000 01101111 11110110 11111111 11000000 00000000
 176   00009         JMP     ,(SOFENT):%      ; Jump into main prodedure at
              00110000 00000000 10100110 11111111 11000000 00000000
 177   0000A                                  ; Start_of_Frame entry point
```

```
179   0000A  ;*****************************************************************
180   0000A  ;*      PROCEDURE main;                                           *
181   0000A  ;* { This procedure does the calls for all filtering              *
182   0000A  ;* operations in the synthesis processor, as well as all the*
183   0000A  ;* input.  All input and output operations are done after    *
184   0000A  ;* S enable goes low but before S clock goes high. Sample     *
185   0000A  ;* output takes place after S enable goes high and all        *
186   0000A  ;* calculations for the sample interval are complete
187   0000A  ;*****************************************************************
188   0000A  ;  Start of sample interval 0
189   0000A  SOFENT:  JMP  .SOF,SOFENT:%       ; Wait for start of frame
             00110111 00000000 10100110 11111111 11000000 00000000
190   0000B           CALL ,EVERY:%           ; Do the stuff done every sample
             00010000 00000101 01100110 11111111 11000000 00000000

192   0000C  ;  Start of sample interval 1
193   0000C           CALL ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

195   0000D  ;  Start of sample interval 2
196   0000D           CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

198   0000E  ;  Start of sample interval 3, get FEL
199   0000E           CALL  ,EVERY:%         ; Do stuff done every frame
             00010000 00000101 01100110 11111111 11000000 00000000
200   0000F           CALL  ,QMFL:%          ; Do Low QMF filter stuff
             00010000 00001010 01110110 11111111 11000000 00000000

202   00010  ;  Start of sample interval 4, get P(0)
203   00010           CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
204   00011           LOAD  P,P               ; Input P(0)
             11100000 01000001 00100100 00010010 11000001 10000100
205   00012           STMS  PTMP0,PTMP0       ; Store in PTMP0
             11100000 01000000 10010100 00001001 11000000 00101010

207   00013  ;  Start of sample interval 5, get P(1)
208   00013           CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
209   00014           LOAD  P,P               ; Input P(1)
             11100000 01000001 00100100 00010010 11000001 10000100
210   00015           STMS  (PTMP0+D#1):%,(PTMP0+D#1):% ; Store in Ptmp1
             11100000 01000000 10100100 00001010 11000000 00101010

212   00016  ;  Start of sample interval 6, get P(2)
213   00016           CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
214   00017           LOAD  P,P               ; Input P(2)
             11100000 01000001 00100100 00010010 11000001 10000100
215   00018           STMS (PTMP0+D#2):%,(PTMP0+D#2):% ; Store in Ptmp2
             11100000 01000000 10110100 00001011 11000000 00101010
```

```
 217   00019  ;  Start of sample interval 7
 218   00019          CALL  ,EVERY!%
             00010000 00000101 01100110 11111111 11000000 00000000
 219   0001A          LOAD PTMP0,PTMP0          ; Get P(0) for the test port
             11100000 01000000 10010100 00001001 11000001 10000100
 220   0001B          STMS LSTP,LSTP            ; Save it
             11100000 01000001 01000100 00010100 11000000 00101010

 222   0001C  ;  Start of sample interval 8, Get FEH
 223   0001C          CALL  ,EVERY!%,
             00010000 00000101 01100110 11111111 11000000 00000000
 224   0001D          CALL  ,QMFH!%                ; Do QMF for FEH input
             00010000 00000111 11100110 11111111 11000000 00000000
 225   0001E          LOAD (PTMP0+D#0)!%,(PTMP0+D#0)!% ; Get P(0) for test point
             11100000 01000000 10010100 00001001 11000001 10000100
 226   0001F          STMS LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010

 228   00020  ;  Start of sample interval 9, set P(3)
 229   00020          CALL  ,EVERY!%
             00010000 00000101 01100110 11111111 11000000 00000000
 230   00021          LOAD  P,P                  ; Input P(3)
             11100000 01000001 00100100 00010010 11000001 10000100
 231   00022          STMS (PTMP0+D#3)!%,(PTMP0+D#3)!% ; Store in Ptmp3
             11100000 01000000 11000100 00001100 11000000 00101010
 232   00023          LOAD (PTMP0+D#1)!%,(PTMP0+D#1)!% ; Get P(1) for test point
             11100000 01000000 10100100 00001010 11000001 10000100
 233   00024          STMS LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010

 235   00025  ;  Start of sample interval 10, set P(4)
 236   00025          CALL  ,EVERY!%
             00010000 00000101 01100110 11111111 11000000 00000000
 237   00026          LOAD  P,P                  ; Input P(4)
             11100000 01000001 00100100 00010010 11000001 10000100
 238   00027          STMS (PTMP0+D#4)!%,(PTMP0+D#4)!% ; Store in Ptmp4
             11100000 01000000 11010100 00001101 11000000 00101010
 239   00028          LOAD (PTMP0+D#2)!%,(PTMP0+D#2)!% ; Get P(2) for the test point
             11100000 01000000 10110100 00001011 11000001 10000100
 240   00029          STMS LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010

 242   0002A  ;  Start of sample interval 11, set P(5)
 243   0002A          CALL  ,EVERY!%
             00010000 00000101 01100110 11111111 11000000 00000000
 244   0002B          LOAD  P,P                  ; Input P(5)
             11100000 01000001 00100100 00010010 11000001 10000100
 245   0002C          STMS  (PTMP0+D#5)!%,(PTMP0+D#5)!% ; Store in Ptmp1
             11100000 01000000 11100100 00001110 11000000 00101010
 246   0002D          LOAD  (PTMP0+D#3)!%,(PTMP0+D#3)!% ; Get P(3) for test point
             11100000 01000000 11000100 00001100 11000001 10000100
 247   0002E          STMS  LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010
 249   0002F  ;  Start of sample interval 12
 250   0002F          CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
 251   00030          LOAD  (PTMP0+D#4):%,(PTMP0+D#4):% ; Get P(4) for test point
             11100000 01000000 11010100 00001101 11000001 10000100
 252   00031          STMS  LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010

 254   00032  ;  Start of sample interval 13, GET FEL
 255   00032          CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
 256   00033          CALL   ,QMFL:%               ; Put the QMF filter
             00010000 00001010 01110110 11111111 11000000 00000000
 257   00034          LOAD (PTMP0+D#5):%,(PTMP0+D#5):% ; Get P(5) for test point
             11100000 01000000 11100100 00001110 11000001 10000100
 258   00035          STMS LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010

 260   00036  ;  Start of sample interval 14, set P(6)
 261   00036          CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
 262   00037          LOAD  P,P                    ; Input P(6)
             11100000 01000001 00100100 00010010 11000001 10000100
 263   00038          STMS  (PTMP0+D#6):%,(PTMP0+D#6):%
             11100000 01000000 11110100 00001111 11000000 00101010
 264   00039          STMS  LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010

 266   0003A  ;  Start of sample interval 15, Get P(7)
 267   0003A          CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
 268   0003B          LOAD  P,P                    ; Input P(7)
             11100000 01000001 00100100 00010010 11000001 10000100
 269   0003C          STMS  (PTMP0+D#7):%,(PTMP0+D#7):% ; Store in Ptemp7
             11100000 01000001 00000100 00010000 11000000 00101010
 270   0003D          STMS  LSTP,LSTP
             11100000 01000001 01000100 00010100 11000000 00101010

 272   0003E  ;  Start of sample interval 16
 273   0003E          CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
 274   0003F          FILTER1 ,CONO,10(0):%      ; Load zero into Acc
             11100000 01010000 00000000 00000000 11100000 01000100
 275   00040          FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
 276   00041          STMS  LSTP,LSTP             ; Set test point predictor to zero
             11100000 01000001 01000100 00010100 11000000 00101010
 277   00042          CALL   ,CPYPRED:%          ; Copy predictors from temporary buff
             00010000 00010110 00010110 11111111 11000000 00000000
 278   00043                                     ; to the active area

 280   00043  ;  Start of sample interval 17
 281   00043          CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
```

```
283   00044  ;  Start of sample interval 18, set FEH
284   00044          CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000
285   00045          CALL   ,QMFH!%            ; Quadrature mirror filter
         00010000 00000111 11100110 11111111 11000000 00000000

287   00046  ;  Start of sample interval 19
288   00046          CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

290   00047  ; FOR n := 0 TO 13 DO
291   00047          PUSH  D#13!%            ; Define next instruction
         01000000 00000000 11010110 11111111 11000000 00000000
292   00048                                  ; as the start of loop
293   00048                                  ; execute loop 14 times
294   00048   ;  Start of sample interval 20 + 10n
295   00048            CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

297   00049   ;  Start of sample interval 21 + 10n
298   00049            CALL ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

300   0004A   ;  Start of sample interval 22 + 10n
301   0004A            CALL ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

303   0004B   ;  Start of sample interval 23 + 10n, input FEL
304   0004B            CALL ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000
305   0004C            CALL   ,QMFL!%          ; Quadrature mirror filter low
         00010000 00001010 01110110 11111111 11000000 00000000

307   0004D  ;  Start of sample interval 24 + 10n
308   0004D            CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

310   0004E  ;  Start of sample interval 25 + 10n
311   0004E            CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

313   0004F  ;  Start of sample interval 26 + 10n
314   0004F            CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

316   00050  ;  Start of sample interval 27 + 10n
317   00050            CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000

319   00051  ;  Start of sample interval 28 + 10n, Input FEH
320   00051            CALL   ,EVERY!%
         00010000 00000101 01100110 11111111 11000000 00000000
321   00052            CALL   ,QMFH!%    ; Input and filter FEH
         00010000 00000111 11100110 11111111 11000000 00000000
```

```
 323   00053  ;  Start of sample interval 29 + 10n
 324   00053               CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
 325   00054          REPLP                    ; ENDFOR
             10000000 01101111 11110110 11111111 11000000 00000000
 326   00055          JMP ,SOFENT:%            ; Do everything again for next frame
             00110000 00000000 10100110 11111111 11000000 00000000
 328   00056  ;************************************************************
 329   00056  ;*     PROCEDURE operations_done_EVERY_sample_interval        *
 330   00056  ;*  { This procedure performs all the operations which are   *
 331   00056  ;* performed every sample interval.  All the input ports are *
 332   00056  ;* read immediatly after SCLCK goes low.  The required input *
 333   00056  ;* data is transfered to working memory in the main procedure*
 334   00056  ;* }                                                         *
 335   00056  ;*    Wait for rising edge of S enable complement.           *
 336   00056  ;*    Output S;                                              *
 337   00056  ;*    Input FE,P;                                            *
 338   00056  ;*    Read test point switch and output appropriate data;    *
 339   00056  ;*    Move data to the delay buffer;                         *
 340   00056  ;*    Low pass the reconstructed residual;                   *
 341   00056  ;*    Perturb and high pass the reconstructed residual;      *
 342   00056  ;*    Sum the high and low pass residuals;                   *
 343   00056  ;*    Pass data through the predictor                        *
 344   00056  ;*    Preemphasiz the data;                                  *
 345   00056  ;*    Shift the ring buffer;                                 *
 346   00056  ;*    Store zero in the ring buffer base; {required for inter*
 347   00056  ;*                                           polation.      }*
 348   00056  ;*    END;                                                   *
 349   00056  ;************************************************************
 350   00056  EVERY:   JMP  .SENBL,EVERY:% ; Wait for .S Enable to be low
              00110101 00000101 01100110 11111111 11000000 00000000
 351   00057  WAITLP:  JMP  .SENBL,OUTLP:% ; Wait for .S Enable to be high
              00110101 00000101 10010110 11111111 11000000 00000000
 352   00058           JMP  ,WAITLP:%
              00110000 00000101 01110110 11111111 11000000 00000000
 353   00059  ;  Now output S
 354   00059  OUTLP:   LDMR S              ; Load S into the Acc
              11100000 01010000 00000001 00111110 11000001 10000100
 355   0005A           OUTPUT SPRT,SPRT  ; Output S
              11100000 01100100 00000110 01000000 11000000 00100011
 356   0005B  ;  Now input from both input ports in case we need to read them
 357   0005B           INPUT  GNINPUT,FEPRT,.MULT1 ; Get FEL or FEH {depending on samp
              11100000 01010111 00000110 01100000 11100000 01000100
 358   0005C           FILTERL              ; And double it
              11100000 01101111 11110110 11111111 11100000 11000000
 359   0005D           STLS   ,FE,FE      ; Store it away for possible later use
              11100000 01000001 00110100 00010011 11000000 00111001
 360   0005E           INPUT  ,PPRT,      ; Input a predictor
              11100000 01010000 00000110 01000000 11000001 10000100
 361   0005F           STMS   P,P         ; Store it away for possible later use
              11100000 01000001 00100100 00010010 11000000 00101010
 362   00060  ;  Test the 'TEST POINT SELECT' switch and output the data to test port
 363   00060           JMP  TSTFEL,NOTFEL:% ; Test if FEL to be output
              00111000 00000110 00100110 11111111 11000000 00000000
 364   00061           LOAD LSTFEL,LSTFEL   ;   Yes - Load FEL into Acc
              11100000 01000001 01010100 00010101 11000001 10000100
 365   00062  NOTFEL:  JMP  TSTFEH,NOTFEH:% ; Test if FEH to be output
              00111001 00000110 01000110 11111111 11000000 00000000
 366   00063           LOAD LSTFEH,LSTFEH   ;   Yes - Load FEH into Acc
              11100000 01000001 01100100 00010110 11000001 10000100
```

```
367   00064  NOTFEH:  JMP  TSTFE,NOTFE:%   ; Test if FE to be output
             00111010 00000110 01100110 11111111 11000000 00000000
368   00065           LOAD  LSTFE,LSTFE    ;   Yes - Load FE into Acc
             11100000 01000001 01110100 00010111 11000001 10000100
369   00066  NOTFE:   JMP  TSTE,NOTE:%      ; Test if AEDD to be output
             00111011 00000110 10000110 11111111 11000000 00000000
370   00067           LDMR  (LPFBS+LPFSZ+D#1):% ;  Yes - Load AEDD into Acc
             11100000 01010000 00000000 11100001 11000001 10000100
371   00068  NOTE:    JMP  TSTHPF,NOTHPF:% ; Test if HPF to be output
             00111100 00000110 10100110 11111111 11000000 00000000
372   00069           LDMR  (HPFBS+HPFSZ+D#1):% ;   Yes - output HPF
             11100000 01010000 00000001 01100110 11000001 10000100
373   0006A  NOTHPF:  JMP  TSTSUM:NOTSUM:% ; Test if regenerated residual to be outpu
             00111101 00000110 11000110 11111111 11000000 00000000
374   0006B           LDMR (SUMBS+D#1):%    ;   Yes - output sum
             11100000 01010000 00000000 11100010 11000001 10000100
375   0006C  NOTSUM:  JMP  TSTSPE,NOTSPE:% ; Test if SPE to be output
             00111110 00000110 11100110 11111111 11000000 00000000
376   0006D           LDMR  (PREDBS+D#1):%  ;   Yes - Load SPE into Acc
             11100000 01010000 00000001 00110100 11000001 10000100
377   0006E  NOTSPE:  JMP  TSTS,NOTS:%      ; Test of S to be output
             00111111 00000111 00000110 11111111 11000000 00000000
378   0006F           LDMR  (DEMBS+D#2):% ;   Yes - Load S into Acc
             11100000 01010000 00000001 00111110 11000001 10000100
379   00070  NOTS:    OUTPUT TEST,TEST      ; Send the value to the analog test port
             11100000 01100010 00000110 00000000 11000000 00100011
380   00071           CALL  ,LPF:%          ; Low pass the unperturbed samples
             00010000 00001111 01000110 11111111 11000000 00000000
381   00072           CALL     ,HPF:%       ; High pass the perturbed samples
             00010000 00010011 11010110 11111111 11000000 00000000
382   00073           FILTER1 ,CON1,(LPFBS+LPFSZ):% ; add the HPF and LPF signals
             11100000 01010000 00010000 11100000 11100000 01000100
383   00074           FILTERN ,CON1,(HPFBS+LPFSZ):%
             11100000 01010000 00010001 01100011 11100010 11000100
384   00075           FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
385   00076           STLR     ,SUMS;
             11100000 00111111 11110000 11100001 11000000 00111001
386   00077  ; Now pass the data through the predictor
387   00077           CALL     ,PRED:%
             00010000 00010100 10110110 11111111 11000000 00000000
388   00078  ; Now deemphasize the data and shift ring buffer data
389   00078           CALL     ,DEEM:%
             00010000 00010111 00010110 11111111 11000000 00000000
390   00079  ; Now load zero into base of ring buffer
391   00079           SPL      ,,,,.SHIFT   ; Shift ring buffer data
             11100000 01101111 11110110 11111111 11001000 00000000
392   0007A           FILTER1  ,CONO,10(0):% ;
             11100000 01010000 00000000 00000000 11100000 01000100
393   0007B           FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
394   0007C           STMR     ,(PERTBS-1):%
             11100000 00111111 11110001 01000000 11000000 00101010
395   0007D           STMR     CRTN,DMFBS ; Store zero and return
             10100000 00111111 11110000 00000000 11000000 00101010
```

```
397   0007E  ;****************************************************************
398   0007E  ;*    PROCEDURE Quadrature Mirror Filter High                   *
399   0007E  ;*(  This  procedure performs a quadrature mirror filter operation*
400   0007E  ;* assuming that FEH is in the FE input buffer.  It place FEH at  *
401   0007E  ;* the start of the QMF portion of the ring buffer.               *
402   0007E  ;*                                                              *
403   0007E  ;* sum := FEH[n] * CQMF[0]                                       *
404   0007E  ;* FOR i := 1 TO 17 DO                                           *
405   0007E  ;*    sum := sum + FEH[n + 2*i] * CQMF[2*i];                     *
406   0007E  ;* sum := FEL[n] * QMF[0] - sum;
407   0007E  ;* FOR i := 0 TO 17 DO                                           *
408   0007E  ;*    sum := sum + FEL[n + 2*i] * CQMF[2*i];                     *
409   0007E  ;* LPFBS[0] := sum;                                              *
410   0007E  ;* END;                                                          *
411   0007E  ;****************************************************************
412   0007E  QMFH:      LOAD      FE,FE                    ; Fetch FEL from the FE input b
             11100000 01000001 00110100 00010011 11000001 10000100
413   0007F             STAR      ,QMFBS                   ; Store at base of QMF ring buf
             11100000 00111111 11110000 00000000 11000000 00101010
414   00080             STORE     ,LSTFEH,LSTFEH           ; Store for the test point
             11100000 01000001 01100100 00010110 11000000 00100011
415   00081             FILTER1  ,(CQMF+D#0):%,(QMFBS+D#0):%  ; Do the FEH terms first
             11100000 01010010 00110000 00000000 11100000 01000100
416   00082             FILTERN  ,(CQMF+D#2):%,(QMFBS+D#10):%
             11100000 01010010 01010000 00001010 11100010 11000100
417   00083             FILTERN  ,(CQMF+D#4):%,(QMFBS+D#20):%
             11100000 01010010 01110000 00010100 11100010 11000100
418   00084             FILTERN  ,(CQMF+D#6):%,(QMFBS+D#30):%
             11100000 01010010 10010000 00011110 11100010 11000100
419   00085             FILTERN  ,(CQMF+D#8):%,(QMFBS+D#40):%
             11100000 01010010 10110000 00101000 11100010 11000100
420   00086             FILTERN  ,(CQMF+D#10):%,(QMFBS+D#50):%
             11100000 01010010 11010000 00110010 11100010 11000100
421   00087             FILTERN  ,(CQMF+D#12):%,(QMFBS+D#60):%
             11100000 01010010 11110000 00111100 11100010 11000100
422   00088             FILTERN  ,(CQMF+D#14):%,(QMFBS+D#70):%
             11100000 01010011 00010000 01000110 11100010 11000100
423   00089             FILTERN  ,(CQMF+D#16):%,(QMFBS+D#80):%
             11100000 01010011 00110000 01010000 11100010 11000100
424   0008A             FILTERN  ,(CQMF+D#18):%,(QMFBS+D#90):%
             11100000 01010011 01010000 01011010 11100010 11000100
425   0008B             FILTERN  ,(CQMF+D#20):%,(QMFBS+D#100):%
             11100000 01010011 01110000 01100100 11100010 11000100
426   0008C             FILTERN  ,(CQMF+D#22):%,(QMFBS+D#110):%
             11100000 01010011 10010000 01101110 11100010 11000100
427   0008D             FILTERN  ,(CQMF+D#24):%,(QMFBS+D#120):%
             11100000 01010011 10110000 01111000 11100010 11000100
428   0008E             FILTERN  ,(CQMF+D#26):%,(QMFBS+D#130):%
             11100000 01010011 11010000 10000010 11100010 11000100
429   0008F             FILTERN  ,(CQMF+D#28):%,(QMFBS+D#140):%
             11100000 01010011 11110000 10001100 11100010 11000100
430   00090             FILTERN  ,(CQMF+D#30):%,(QMFBS+D#150):%
             11100000 01010100 00010000 10010110 11100010 11000100
```

```
431   00091          FILTERN  ,(CQMF+D#32):%,(QMFBS+D#160):%
               11100000 01010100 00110000 10100000 11100010 11000100
432   00092          FILTERN  ,(CQMF+D#34):%,(QMFBS+D#170):%
               11100000 01010100 01010000 10101010 11100010 11000100
433   00093          FILTRNH  ,(CQMF+D#00):%,(QMFBS+D#5):% ;sum:=FEL[0]+QMF[0]-sum
               11100000 01010010 00110000 00000101 11100110 11000100
434   00094          FILTERN  ,(CQMF+D#02):%,(QMFBS+D#15):%
               11100000 01010010 01010000 00001111 11100010 11000100
435   00095          FILTERN  ,(CQMF+D#04):%,(QMFBS+D#25):%
               11100000 01010010 01110000 00011001 11100010 11000100
436   00096          FILTERN  ,(CQMF+D#06):%,(QMFBS+D#35):%
               11100000 01010010 10010000 00100011 11100010 11000100
437   00097          FILTERN  ,(CQMF+D#08):%,(QMFBS+D#45):%
               11100000 01010010 10110000 00101101 11100010 11000100
438   00098          FILTERN  ,(CQMF+D#10):%,(QMFBS+D#55):%
               11100000 01010010 11010000 00110111 11100010 11000100
439   00099          FILTERN  ,(CQMF+D#12):%,(QMFBS+D#65):%
               11100000 01010010 11110000 01000001 11100010 11000100
440   0009A          FILTERN  ,(CQMF+D#14):%,(QMFBS+D#75):%
               11100000 01010011 00010000 01001011 11100010 11000100
441   0009B          FILTERN  ,(CQMF+D#16):%,(QMFBS+D#85):%
               11100000 01010011 00110000 01010101 11100010 11000100
442   0009C          FILTERN  ,(CQMF+D#18):%,(QMFBS+D#95):%
               11100000 01010011 01010000 01011111 11100010 11000100
443   0009D          FILTERN  ,(CQMF+D#20):%,(QMFBS+D#105):%
               11100000 01010011 01110000 01101001 11100010 11000100
444   0009E          FILTERN  ,(CQMF+D#22):%,(QMFBS+D#115):%
               11100000 01010011 10010000 01110011 11100010 11000100
445   0009F          FILTERN  ,(CQMF+D#24):%,(QMFBS+D#125):%
               11100000 01010011 10110000 01111101 11100010 11000100
446   000A0          FILTERN  ,(CQMF+D#26):%,(QMFBS+D#135):%
               11100000 01010011 11010000 10000111 11100010 11000100
447   000A1          FILTERN  ,(CQMF+D#28):%,(QMFBS+D#145):%
               11100000 01010011 11110000 10010001 11100010 11000100
448   000A2          FILTERN  ,(CQMF+D#30):%,(QMFBS+D#155):%
               11100000 01010100 00010000 10011011 11100010 11000100
449   000A3          FILTERN  ,(CQMF+D#32):%,(QMFBS+D#165):%
               11100000 01010100 00110000 10100101 11100010 11000100
450   000A4          FILTERN  ,(CQMF+D#34):%,(QMFBS+D#175):%
               11100000 01010100 01010000 10101111 11100010 11000100
451   000A5          FILTERL
               11100000 01101111 11110110 11111111 11100000 11000000
452   000A6          JMP       ,PERT:%        ; Perturb sample location
               00110000 00001101 00000110 11111111 11000000 00000000
453   000A7                                   ; and return
```

```
LINE    ADDR   SYNTHESIS PROCESSOR ASSEMBLER SOURCE FILE, SEPT 10    PAGE   18

 455    000A7  ;*****************************************************
 456    000A7  ;*   PROCEDURE Quadrature_Mirror_Filter_Low           *
 457    000A7  ;* { This procedure does the quadrature mirror filtering*
 458    000A7  ;* operation assuming FEL is in the FE input buffer.   }*
 459    000A7  ;*                                                     *
 460    000A7  ;* sum := FEH[n+0] * CQMF[1]                           *
 461    000A7  ;* FOR i := 1 TO 17 DO                                 *
 462    000A7  ;*   sum := sum + FEH[n+2*i] * CQMF[2*i+1];            *
 463    000A7  ;* FOR i := 0 TO 17 DO                                 *
 464    000A7  ;*   sum := sum + FEH[n+2*i] * CQMF[2*i+1];            *
 465    000A7  ;* LPFBS := sum;                                       *
 466    000A7  ;*                                                     *
 467    000A7  ;*****************************************************
 468    000A7  QMFL:     LOAD     FE,FE
               11100000 01000001 00110100 00010011 11000001 10000100
 469    000A8            STMR     ,QMFBS                  ; Load FEL and store in Ring b
               11100000 00111111 11110000 00000000 11000000 00101010
 470    000A9            STORE    ,LSTFEL,LSTFEL          ; Store for the test point
               11100000 01000001 01010100 00010101 11000000 00100011
 471    000AA            FILTER1  ,(CQMF+D#1):%,(QMFBS+D#5):%  ; Do the FEH terms first
               11100000 01010010 01000000 00000101 11100000 01000100
 472    000AB            FILTERN  ,(CQMF+D#3):%,(QMFBS+D#15):%
               11100000 01010010 01100000 00001111 11100010 11000100
 473    000AC            FILTERN  ,(CQMF+D#5):%,(QMFBS+D#25):%
               11100000 01010010 10000000 00011001 11100010 11000100
 474    000AD            FILTERN  ,(CQMF+D#7):%,(QMFBS+D#35):%
               11100000 01010010 10100000 00100011 11100010 11000100
 475    000AE            FILTERN  ,(CQMF+D#9):%,(QMFBS+D#45):%
               11100000 01010010 11000000 00101101 11100010 11000100
 476    000AF            FILTERN  ,(CQMF+D#11):%,(QMFBS+D#55):%
               11100000 01010010 11100000 00110111 11100010 11000100
 477    000B0            FILTERN  ,(CQMF+D#13):%,(QMFBS+D#65):%
               11100000 01010011 00000000 01000001 11100010 11000100
 478    000B1            FILTERN  ,(CQMF+D#15):%,(QMFBS+D#75):%
               11100000 01010011 00100000 01001011 11100010 11000100
 479    000B2            FILTERN  ,(CQMF+D#17):%,(QMFBS+D#85):%
               11100000 01010011 01000000 01010101 11100010 11000100
 480    000B3            FILTERN  ,(CQMF+D#19):%,(QMFBS+D#95):%
               11100000 01010011 01100000 01011111 11100010 11000100
 481    000B4            FILTERN  ,(CQMF+D#21):%,(QMFBS+D#105):%
               11100000 01010011 10000000 01101001 11100010 11000100
 482    000B5            FILTERN  ,(CQMF+D#23):%,(QMFBS+D#115):%
               11100000 01010011 10100000 01110011 11100010 11000100
 483    000B6            FILTERN  ,(CQMF+D#25):%,(QMFBS+D#125):%
               11100000 01010011 11000000 01111101 11100010 11000100
 484    000B7            FILTERN  ,(CQMF+D#27):%,(QMFBS+D#135):%
               11100000 01010011 11100000 10000111 11100010 11000100
 485    000B8            FILTERN  ,(CQMF+D#29):%,(QMFBS+D#145):%
               11100000 01010100 00000000 10010001 11100010 11000100
 486    000B9            FILTERN  ,(CQMF+D#31):%,(QMFBS+D#155):%
               11100000 01010100 00100000 10011011 11100010 11000100
 487    000BA            FILTERN  ,(CQMF+D#33):%,(QMFBS+D#165):%
               11100000 01010100 01000000 10100101 11100010 11000100
 488   000BB             FILTERN  ,(CQMF+D#35)!%,(QMFBS+D#175)!%
             11100000 01010100 01100000 10101111 11100010 11000100
 489   000BC             FILTERN  ,(CQMF+D#1)!%,(QMFBS+D#10)!% ; Now do FEL terms
             11100000 01010010 01000000 00001010 11100010 11000100
 490   000BD             FILTERN  ,(CQMF+D#3)!%,(QMFBS+D#20)!%
             11100000 01010010 01100000 00010100 11100010 11000100
 491   000BE             FILTERN  ,(CQMF+D#5)!%,(QMFBS+D#30)!%
             11100000 01010010 10000000 00011110 11100010 11000100
 492   000BF             FILTERN  ,(CQMF+D#7)!%,(QMFBS+D#40)!%
             11100000 01010010 10100000 00101000 11100010 11000100
 493   000C0             FILTERN  ,(CQMF+D#9)!%,(QMFBS+D#50)!%
             11100000 01010010 11000000 00110010 11100010 11000100
 494   000C1             FILTERN  ,(CQMF+D#11)!%,(QMFBS+D#60)!%
             11100000 01010010 11100000 00111100 11100010 11000100
 495   000C2             FILTERN  ,(CQMF+D#13)!%,(QMFBS+D#70)!%
             11100000 01010011 00000000 01000110 11100010 11000100
 496   000C3             FILTERN  ,(CQMF+D#15)!%,(QMFBS+D#80)!%
             11100000 01010011 00100000 01010000 11100010 11000100
 497   000C4             FILTERN  ,(CQMF+D#17)!%,(QMFBS+D#90)!%
             11100000 01010011 01000000 01011010 11100010 11000100
 498   000C5             FILTERN  ,(CQMF+D#19)!%,(QMFBS+D#100)!%
             11100000 01010011 01100000 01100100 11100010 11000100
 499   000C6             FILTERN  ,(CQMF+D#21)!%,(QMFBS+D#110)!%
             11100000 01010011 10000000 01101110 11100010 11000100
 500   000C7             FILTERN  ,(CQMF+D#23)!%,(QMFBS+D#120)!%
             11100000 01010011 10100000 01111000 11100010 11000100
 501   000C8             FILTERN  ,(CQMF+D#25)!%,(QMFBS+D#130)!%
             11100000 01010011 11000000 10000010 11100010 11000100
 502   000C9             FILTERN  ,(CQMF+D#27)!%,(QMFBS+D#140)!%
             11100000 01010011 11100000 10001100 11100010 11000100
 503   000CA             FILTERN  ,(CQMF+D#29)!%,(QMFBS+D#150)!%
             11100000 01010100 00000000 10010110 11100010 11000100
 504   000CB             FILTERN  ,(CQMF+D#31)!%,(QMFBS+D#160)!%
             11100000 01010100 00100000 10100000 11100010 11000100
 505   000CC             FILTERN  ,(CQMF+D#33)!%,(QMFBS+D#170)!%
             11100000 01010100 01000000 10101010 11100010 11000100
 506   000CD             FILTERN  ,(CQMF+D#35)!%,(QMFBS+D#180)!%
             11100000 01010100 01100000 10110100 11100010 11000100
 507   000CE             FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
 508   000CF             JMP       ,PERT!%        ; Perturb sample location
             00110000 00001101 00000110 11111111 11000000 00000000
 509   000D0                                      ; and return
 510   000D0                                    ; return
```

```
000D0  ;***********************************************************
000D0  ;*    PROCEDURE PERTurbate;                                 *
000D0  ;*                                                          *
000D0  ;*{  This procedure perterbates the location of the FE     }*
000D0  ;*  BEGIN                                                   *
000D0  ;*  Multiply accumulator by CON4;                           *
000D0  ;*  Generatate seed uniformly distributed in [-thresh,thresh]*
000D0  ;*  IF magnitude(FE) > seed THEN store at PERTBS            *
000D0  ;*      ELSE IF MSB of least is one THEN store at PERTBS-1  *
000D0  ;*         ELSE store at PERTBS+1                           *
000D0  ;*  END;                                                    *
000D0  ;*                                                          *
000D0  ;* { The Fe sample is perturbed with the following probability*
000D0  ;* density function.                                        *
000D0  ;*                        |                                *
000D0  ;* Probability of being   |                                *
000D0  ;* Perturbed              |                                *
000D0  ;*                        +0.5                             *
000D0  ;*                     +  |  +                             *
000D0  ;*                  +     |     +                          *
000D0  ;*               +        |        +                       *
000D0  ;*            +           |           +                    *
000D0  ;*         +              |              +                 *
000D0  ;*      +                 |                 +              *
000D0  ;* --+-----------------------------------------+---------- *
000D0  ;*  -THRESH               0                 THRESH         *
000D0  ;*                 Input sample value                      *
000D0  ;*  END}                                                    *
000D0  ;***********************************************************
000D0  PERT:
000D0  ; Now do a full precision multiply by CON2
000D0           STLR     ,LEAST        ; Store least sig prod
       11100000 00111111 11110011 11111111 11000000 00111001
000D1           STMR     ,MOST         ; store most sig product
       11100000 00111111 11110011 11111110 11000000 00101010
000D2           FILTERL  ,CON2,MOST    ; Multiply most sig by CON2
       11100000 01010110 11000011 11111110 11100000 01000100
000D3           FILTERL
       11100000 01101111 11110110 11111111 11100000 11000000
000D4           STLR     ,MOST
       11100000 00111111 11110011 11111110 11000000 00111001
000D5           LDMR     MOST          ; Shift left 16 places
       11100000 01010000 00000011 11111110 11000001 10000100
000D6           FILTRAU  ,CON2,LEAST   ; Add CON4 times least sig product
       11100000 01010110 11000011 11111111 11000010 01000100
000D7           FILTERL
       11100000 01101111 11110110 11111111 11100000 11000000
000D8           STLR     ,LEAST
       11100000 00111111 11110011 11111111 11000000 00111001
000D9           STMS     LSTFE,LSTFE   ; Save for the test point
       11100000 01000001 01110100 00010111 11000000 00101010
000DA           STMR     ,MOST         ; Store at base of LPF
       11100000 00111111 11110011 11111110 11000000 00101010
```

```
 554   000DB            STMR      ,DLYB5
               11100000 00111111 11110000 10111110 11000000 00101010
 555   000DC   ; Take the absolute value
 556   000DC            JMP    NEG,NEGATE:%          ; Test last bus value was negative
               00110010 00001101 11100110 11111111 11000000 00000000
 557   000DD            JMP    ,POSTV:%
               00110000 00001110 00010110 11111111 11000000 00000000
 558   000DE   NEGATE:  FILTER1 ,CONM1,MOST:%        ; IF negative then negate it
               11100000 01010000 00100011 11111110 11100000 01000100
 559   000DF            FILTERL
               11100000 01101111 11110110 11111111 11100000 11000000
 560   000E0            STLR    ,MOST
               11100000 00111111 11110011 11111110 11000000 00111001
 561   000E1   POSTV:
 562   000E1            FILTER1 ,THRESH,LEAST        ; Scale [-1,1) remainder
               11100000 01010000 01010011 11111111 11100000 01000100
 563   000E2                                          ; to [-THRESH,THRESH)
 564   000E2            FILTERL                      ; To get random seed
               11100000 01101111 11110110 11111111 11100000 11000000
 565   000E3            STMR    TEMP,TEMP            ; leave result in LSW
               11100000 01000001 10000100 00011000 11000000 00101010
 566   000E4            SPL     ,,(D#256*COEFB+CON1):%,(D#256*SCRAT+TEMP):%,,MULT1
               11100000 01010000 00010100 00011000 11100000 01000100
 567   000E5                                          ; load seed into accumulator
 568   000E5            FILTERL ,CONM1,MOST          ; Subtract absolute value of signal
               11100000 01010000 00100011 11111110 11100010 11000100
 569   000E6            FILTERL
               11100000 01101111 11110110 11111111 11100000 11000000
 570   000E7            STLR    ,MOST                ; Put on the bus
               11100000 00111111 11110011 11111110 11000000 00111001
 571   000E8            JMP     NEG,NOPERT:%         ; If magnitude of signal > 128 don'T
               00110010 00001111 00100110 11111111 11000000 00000000
 572   000E9            FILTERL ,(CLFF+D#15):%,LEAST ;scramble bits in LEAST
               11100000 01010001 00110011 11111111 11100000 01000100
 573   000EA            FILTERL
               11100000 01101111 11110110 11111111 11100000 11000000
 574   000EB            STLR    ,LEAST               ; Save least sig word of scrambled 1
               11100000 00111111 11110011 11111111 11000000 00111001
 575   000EC            LDMR    LEAST                ; Put MSB of LEAST into sign bit
               11100000 01010000 00000011 11111111 11000001 10000100
 576   000ED            JMP     NEG,PERTP:%
               00110010 00001111 00000110 11111111 11000000 00000000
 577   000EE            LOAD    LSTFE,LSTFE
               11100000 01000001 01110100 00010111 11000001 10000100
 578   000EF            STMR    CRTN,(PERTBS-D#1):%
               10100000 00111111 11110001 01000000 11000000 00101010
 579   000F0   PERTP:   LOAD    LSTFE,LSTFE
               11100000 01000001 01110100 00010111 11000001 10000100
 580   000F1            STMR    CRTN,(PERTBS+D#1):%
               10100000 00111111 11110001 01000010 11000000 00101010
 581   000F2   NOPERT:  LOAD    LSTFE,LSTFE
               11100000 01000001 01110100 00010111 11000001 10000100
 582   000F3            STMR    CRTN,(PERTBS+0 :%
               10100000 00111111 11110001 01000001 11000000 00101010
```

```
000F4 ;**************************************************************
000F4 ;*    PROCEDURE Low_Pass_Filter;                                *
000F4 ;*{  This Procedure low pass filters the data in the low pass*
000F4 ;* Filter buffer.                                              }*
000F4 ;*  sum := CLPF[0] * LPFBS[0];                                  *
000F4 ;*  FOR i := 1 TO 30 DO                                         *
000F4 ;*    sum := sum + CLPF[i] * LPFBS[i];                          *
000F4 ;*  LPFBS[31] := sum;                                           *
000F4 ;*  DLYBS[0] := DDFBS[0];                                       *
000F4 ;*  END;                                                        *
000F4 ;*                                                              *
000F4 ;**************************************************************
000F4 LPF:      FILTER1  ,(CLPF+D#0):%,(LPFBS+D#0):%
      11100000 01010000 01000000 11000001 11100000 01000100
000F5           FILTERN  ,(CLPF+D#1):%,(LPFBS+D#1):%
      11100000 01010000 01010000 11000010 11100010 11000100
000F6           FILTERN  ,(CLPF+D#2):%,(LPFBS+D#2):%
      11100000 01010000 01100000 11000011 11100010 11000100
000F7           FILTERN  ,(CLPF+D#3):%,(LPFBS+D#3):%
      11100000 01010000 01110000 11000100 11100010 11000100
000F8           FILTERN  ,(CLPF+D#4):%,(LPFBS+D#4):%
      11100000 01010000 10000000 11000101 11100010 11000100
000F9           FILTERN  ,(CLPF+D#5):%,(LPFBS+D#5):%
      11100000 01010000 10010000 11000110 11100010 11000100
000FA           FILTERN  ,(CLPF+D#6):%,(LPFBS+D#6):%
      11100000 01010000 10100000 11000111 11100010 11000100
000FB           FILTERN  ,(CLPF+D#7):%,(LPFBS+D#7):%
      11100000 01010000 10110000 11001000 11100010 11000100
000FC           FILTERN  ,(CLPF+D#8):%,(LPFBS+D#8):%
      11100000 01010000 11000000 11001001 11100010 11000100
000FD           FILTERN  ,(CLPF+D#9):%,(LPFBS+D#9):%
      11100000 01010000 11010000 11001010 11100010 11000100
000FE           FILTERN  ,(CLPF+D#10):%,(LPFBS+D#10):%
      11100000 01010000 11100000 11001011 11100010 11000100
000FF           FILTERN  ,(CLPF+D#11):%,(LPFBS+D#11):%
      11100000 01010000 11110000 11001100 11100010 11000100
00100           FILTERN  ,(CLPF+D#12):%,(LPFBS+D#12):%
      11100000 01010001 00000000 11001101 11100010 11000100
00101           FILTERN  ,(CLPF+D#13):%,(LPFBS+D#13):%
      11100000 01010001 00010000 11001110 11100010 11000100
00102           FILTERN  ,(CLPF+D#14):%,(LPFBS+D#14):%
      11100000 01010001 00100000 11001111 11100010 11000100
00103           FILTERN  ,(CLPF+D#15):%,(LPFBS+D#15):%
      11100000 01010001 00110000 11010000 11100010 11000100
00104           FILTERN  ,(CLPF+D#16):%,(LPFBS+D#16):%
      11100000 01010001 01000000 11010001 11100010 11000100
00105           FILTERN  ,(CLPF+D#17):%,(LPFBS+D#17):%
      11100000 01010001 01010000 11010010 11100010 11000100
00106           FILTERN  ,(CLPF+D#18):%,(LPFBS+D#18):%
      11100000 01010001 01100000 11010011 11100010 11000100
00107           FILTERN  ,(CLPF+D#19):%,(LPFBS+D#19):%
      11100000 01010001 01110000 11010100 11100010 11000100
00108           FILTERN  ,(CLPF+D#20):%,(LPFBS+D#20):%
      11100000 01010001 10000000 11010101 11100010 11000100
```

```
617   00109          FILTERN  ,(CLPF+D#21):%,(LPFBS+D#21):%
             11100000 01010001 10010000 11010110 11100010 11000100
618   0010A          FILTERN  ,(CLPF+D#22):%,(LPFBS+D#22):%
             11100000 01010001 10100000 11010111 11100010 11000100
619   0010B          FILTERN  ,(CLPF+D#23):%,(LPFBS+D#23):%
             11100000 01010001 10110000 11011000 11100010 11000100
620   0010C          FILTERN  ,(CLPF+D#24):%,(LPFBS+D#24):%
             11100000 01010001 11000000 11011001 11100010 11000100
621   0010D          FILTERN  ,(CLPF+D#25):%,(LPFBS+D#25):%
             11100000 01010001 11010000 11011010 11100010 11000100
622   0010E          FILTERN  ,(CLPF+D#26):%,(LPFBS+D#26):%
             11100000 01010001 11100000 11011011 11100010 11000100
623   0010F          FILTERN  ,(CLPF+D#27):%,(LPFBS+D#27):%
             11100000 01010001 11110000 11011100 11100010 11000100
624   00110          FILTERN  ,(CLPF+D#28):%,(LPFBS+D#28):%
             11100000 01010010 00000000 11011101 11100010 11000100
625   00111          FILTERN  ,(CLPF+D#29):%,(LPFBS+D#29):%
             11100000 01010010 00010000 11011110 11100010 11000100
626   00112          FILTERN  ,(CLPF+D#30):%,(LPFBS+D#30):%
             11100000 01010010 00100000 11011111 11100010 11000100
627   00113          FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
628   00114          STLR     ,LEAST         ; Store least sig prod
             11100000 00111111 11110011 11111111 11000000 00111001
629   00115          STMR     ,MOST          ; store most sig product
             11100000 00111111 11110011 11111110 11000000 00101010
630   00116          FILTERI  ,GNLPF,MOST   ; Multiply most sig by GNLPF
             11100000 01010111 00100011 11111110 11100000 01000100
631   00117          FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
632   00118          STLR     ,MOST
             11100000 00111111 11110011 11111110 11000000 00111001
633   0011F          LDMR     MOST           ; Shift left 16 places
             11100000 01010000 00000011 11111110 11000001 10000100
634   0011A          FILTRAC  ,GNLPF,LEAST  ; Add GNLPF times least sig product
             11100000 01010111 00100011 11111111 11000010 01000100
635   0011B          FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
636   0011C          STMR     CRTN,(LPFBS+LPFSZ):%  ; Store it away
             10100000 00111111 11110000 11100000 11000000 00101010
```

```
638   0011D  ;**********************************************************
639   0011D  ;*    PROCEDURE High_Pass_Filter;                            *
640   0011D  ;*{  This Procedure high pass filters the data in the high   *
641   0011D  ;*  pass Filter buffer.                                     }*
642   0011D  ;*  sum := CLPF[0] * HPFBS[0];                               *
643   0011D  ;*  FOR i := 1 TO 30 DO                                      *
644   0011D  ;*    sum := sum + CLPF[i] * HPFBS[i];                       *
645   0011D  ;*  HPFBS[31] := 5*HPFBS[15]-sum;                            *
646   0011D  ;*  END;                                                     *
647   0011D  ;*                                                           *
648   0011D  ;**********************************************************
649   0011D  HPF:       FILTER1  ,(CLPF+D#0):%,(HPFBS+D#0):%
             11100000 01010000 01000001 01000100 11100000 01000100
650   0011E             FILTERN  ,(CLPF+D#1):%,(HPFBS+D#1):%
             11100000 01010000 01010001 01000101 11100010 11000100
651   0011F             FILTERN  ,(CLPF+D#2):%,(HPFBS+D#2):%
             11100000 01010000 01100001 01000110 11100010 11000100
652   00120             FILTERN  ,(CLPF+D#3):%,(HPFBS+D#3):%
             11100000 01010000 01110001 01000111 11100010 11000100
653   00121             FILTERN  ,(CLPF+D#4):%,(HPFBS+D#4):%
             11100000 01010000 10000001 01001000 11100010 11000100
654   00122             FILTERN  ,(CLPF+D#5):%,(HPFBS+D#5):%
             11100000 01010000 10010001 01001001 11100010 11000100
655   00123             FILTERN  ,(CLPF+D#6):%,(HPFBS+D#6):%
             11100000 01010000 10100001 01001010 11100010 11000100
656   00124             FILTERN  ,(CLPF+D#7):%,(HPFBS+D#7):%
             11100000 01010000 10110001 01001011 11100010 11000100
657   00125             FILTERN  ,(CLPF+D#8):%,(HPFBS+D#8):%
             11100000 01010000 11000001 01001100 11100010 11000100
658   00126             FILTERN  ,(CLPF+D#9):%,(HPFBS+D#9):%
             11100000 01010000 11010001 01001101 11100010 11000100
659   00127             FILTERN  ,(CLPF+D#10):%,(HPFBS+D#10):%
             11100000 01010000 11100001 01001110 11100010 11000100
660   00128             FILTERN  ,(CLPF+D#11):%,(HPFBS+D#11):%
             11100000 01010000 11110001 01001111 11100010 11000100
661   00129             FILTERN  ,(CLPF+D#12):%,(HPFBS+D#12):%
             11100000 01010001 00000001 01010000 11100010 11000100
662   0012A             FILTERN  ,(CLPF+D#13):%,(HPFBS+D#13):%
             11100000 01010001 00010001 01010001 11100010 11000100
663   0012B             FILTERN  ,(CLPF+D#14):%,(HPFBS+D#14):%
             11100000 01010001 00100001 01010010 11100010 11000100
664   0012C             FILTERN  ,(CLPF+D#15):%,(HPFBS+D#15):%
             11100000 01010001 00110001 01010011 11100010 11000100
665   0012D             FILTERN  ,(CLPF+D#16):%,(HPFBS+D#16):%
             11100000 01010001 01000001 01010100 11100010 11000100
666   0012E             FILTERN  ,(CLPF+D#17):%,(HPFBS+D#17):%
             11100000 01010001 01010001 01010101 11100010 11000100
667   0012F             FILTERN  ,(CLPF+D#18):%,(HPFBS+D#18):%
             11100000 01010001 01100001 01010110 11100010 11000100
668   00130             FILTERN  ,(CLPF+D#19):%,(HPFBS+D#19):%
             11100000 01010001 01110001 01010111 11100010 11000100
669   00131             FILTERN  ,(CLPF+D#20):%,(HPFBS+D#20):%
             11100000 01010001 10000001 01011000 11100010 11000100
```

```
 670   00132             FILTERN  ,(CLPF+D#21):%,(HPFBS+D#21):%
              11100000 01010001 10010001 01011001 11100010 11000100
 671   00133             FILTERN  ,(CLPF+D#22):%,(HPFBS+D#22):%
              11100000 01010001 10100001 01011010 11100010 11000100
 672   00134             FILTERN  ,(CLPF+D#23):%,(HPFBS+D#23):%
              11100000 01010001 10110001 01011011 11100010 11000100
 673   00135             FILTERN  ,(CLPF+D#24):%,(HPFBS+D#24):%
              11100000 01010001 11000001 01011100 11100010 11000100
 674   00136             FILTERN  ,(CLPF+D#25):%,(HPFBS+D#25):%
              11100000 01010001 11010001 01011101 11100010 11000100
 675   00137             FILTERN  ,(CLPF+D#26):%,(HPFBS+D(26):%
              11100000 01010001 11100001 01011110 11100010 11000100
 676   00138             FILTERN  ,(CLPF+D#27):%,(HPFBS+D#27):%
              11100000 01010001 11110001 01011111 11100010 11000100
 677 - 00139             FILTERN  ,(CLPF+D#28):%,(HPFBS+D#28):%
              11100000 01010010 00000001 01100000 11100010 11000100
 678   0013A             FILTERN  ,(CLPF+D#29):%,(HPFBS+D#29):%
              11100000 01010010 00010001 01100001 11100010 11000100
 679   0013B             FILTERN  ,(CLPF+D#30):%,(HPFBS+D#30):%
              11100000 01010010 00100001 01100010 11100010 11000100
 680   0013C             FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 681   0013D             STLR     ,LEAST          ; Store least sig prod
              11100000 00111111 11110011 11111111 11000000 00111001
 682   0013E             STMR     ,MOST           ; store most sig product
              11100000 00111111 11110011 11111110 11000000 00101010
 683   0013F             FILTER1  ,GNLPF,MOST   ; Multiply most sig by GNLPF
              11100000 01010111 00100011 11111110 11100000 01000100
 684   00140             FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 685   00141             STLR     ,MOST
              11100000 00111111 11110011 11111110 11000000 00111001
 686   00142             LDMR     MOST            ; Shift left 16 places
              11100000 01010000 00000011 11111110 11000001 10000100
 687   00143             FILTRAU  ,GNLPF,LEAST  ; Add GNLPF times least sig product
              11100000 01010111 00100011 11111111 11000010 01000100
 688   00144             FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 689   00145             STMR     ,(HPFBS+LPFSZ):%  ; Store it away
              11100000 00111111 11110001 01100011 11000000 00101010
 690   00146             FILTER1  ,CON1,(HPFBS+LPFSZ):% ; subtract 5*middle element
              11100000 01010000 00010001 01100011 11100000 01000100
 691   00147                                             ; to get complement HPF from LP
 692   00147             FILTRNM  ,CON4,(HPFBS+D#15):%
              11100000 01010110 11010001 01010011 11100110 11000100
 693   00148             FILTERN  ,CON1,(HPFBS+D#15):%
              11100000 01010000 00010001 01010011 11100010 11000100
 694   00149             FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 695   0014A             STLR     CRTN,(HPFBS+LPFSZ):%  ; Store that away
              10100000 00111111 11110001 01100011 11000000 00111001
```

```
697    0014B  ;**********************************************************
698    0014B  ;*          PROCEDURE PREDict;                             *
699    0014B  ;*   { This procedure predictor filters data in the        *
700    0014B  ;* filter buffer of the ring buffer.                      }*
701    0014B  ;*                                                         *
702    0014B  ;**********************************************************
703    0014B  PRED:    FILTER1 SCRAT,(P0+D#0):%,(PREDBS+D#1):%
              11100000 01000000 00000001 00110100 11100000 01000100
704    0014C           FILTERN SCRAT,(P0+D#1):%,(PREDBS+D#2):%
              11100000 01000000 00010001 00110101 11100010 11000100
705    0014D           FILTERN SCRAT,(P0+D#2):%,(PREDBS+D#3):%
              11100000 01000000 00100001 00110110 11100010 11000100
706    0014E           FILTERN SCRAT,(P0+D#3):%,(PREDBS+D#4):%
              11100000 01000000 00110001 00110111 11100010 11000100
707    0014F           FILTERN SCRAT,(P0+D#4):%,(PREDBS+D#5):%
              11100000 01000000 01000001 00111000 11100010 11000100
708    00150           FILTERN SCRAT,(P0+D#5):%,(PREDBS+D#6):%
              11100000 01000000 01010001 00111001 11100010 11000100
709    00151           FILTERN SCRAT,(P0+D#6):%,(PREDBS+D#7):%
              11100000 01000000 01100001 00111010 11100010 11000100
710    00152           FILTERN SCRAT,(P0+D#7):%,(PREDBS+D#8):%
              11100000 01000000 01110001 00111011 11100010 11000100
711    00153           FILTRNH COEFB,PREDSCL,PREDBS
              11100000 01010110 10110001 00110011 11100110 11000100
712    00154           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
713    00155  ; Do a double precion multiply by 8
714    00155           STLR     ,LEAST          ; Store least sig prod
              11100000 00111111 11110011 11111111 11000000 00111001
715    00156           STMR     ,MOST           ; store most sig product
              11100000 00111111 11110011 11111110 11000000 00101010
716    00157           FILTER1  ,CONB,MOST    ; Multiply most sig by 8
              11100000 01010110 11100011 11111110 11100000 01000100
717    00158           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
718    00159           STLR     ,MOST
              11100000 00111111 11110011 11111110 11000000 00111001
719    0015A           LDMR      MOST           ; Shift left 16 places
              11100000 01010000 00000011 11111110 11000001 10000100
720    0015B           FILTRAU  ,CONB,LEAST   ; Add 8 times least sig product
              11100000 01010110 11100011 11111111 11000010 01000100
721    0015C           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
722    0015D           STMR    ,PREDBS
              11100000 00111111 11110001 00110011 11000000 00101010
723    0015E           FILTER1  ,GNPRED,PREDBS ; Scale so no overflow in deemphasis
              11100000 01010111 01010001 00110011 11100000 01000100
724    0015F           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
725    00160           STMR   CRTN,DEMBS
              10100000 00111111 11110001 00111100 11000000 00101010
```

```
727   00161  ;*************************************************************
728   00161  ;*       PROCEDURE CoPY_PREDictor_coefficients                *
729   00161  ;*( This procedure copies the predictor coefficients from    *
730   00161  ;* the temporary predictor buffer to the active predictor    *
731   00161  ;* buffer.                                                  }*
732   00161  ;*                                                           *
733   00161  ;*************************************************************
734   00161  CPYPRED: LOAD (PTMP0):%,(PTMP0):%
             11100000 01000000 10010100 00001001 11000001 10000100
735   00162           STORE ,(P0):%,(P0):%
             11100000 01000000 00000100 00000000 11000000 00100011
736   00163           LOAD (PTMP0+D#1):%,(PTMP0+D#1):%
             11100000 01000000 10100100 00001010 11000001 10000100
737   00164           STORE ,B(P0+D#1):%,(P0+D#1):%
             11100000 01000000 00010100 00000001 11000000 00100011
738   00165           LOAD B(PTMP0+D#2):%,(PTMP0+D#2):%
             11100000 01000000 10110100 00001011 11000001 10000100
739   00166           STORE ,B(P0+D#2):%,(P0+D#2):%
             11100000 01000000 00100100 00000010 11000000 00100011
740   00167           LOAD B(PTMP0+D#3):%,(P0+D#3):%
             11100000 01000000 11000100 00000011 11000001 10000100
741   0016B           STORE ,B(P0+D#3):%,(P0+D#3):%
             11100000 01000000 00110100 00000011 11000000 00100011
742   00169           LOAD B(PTMP0+D#4):%,(PTMP0+D#4):%
             11100000 01000000 11010100 00001101 11000001 10000100
743   0016A           STORE ,B(P0+D#4):%,(P0+D#4):%
             11100000 01000000 01000100 00000100 11000000 00100011
744   0016B           LOAD B(PTMP0+D#5):%,(PTMP0+D#5):%
             11100000 01000000 11100100 00001110 11000001 10000100
745   0016C           STORE ,B(P0+D#5):%,(P0+D#5):%
             11100000 01000000 01010100 00000101 11000000 00100011
746   0016D           LOAD B(PTMP0+D#6):%,(PTMP0+D#6):%
             11100000 01000000 11110100 00001111 11000001 10000100
747   0016E           STORE ,B(P0+D#6):%,(P0+D#6):%
             11100000 01000000 01100100 00000110 11000000 00100011
748   0016F           LOAD B(PTMP0+D#7):%,(PTMP0+D#7):%
             11100000 01000001 00000100 00010000 11000001 10000100
749   00170           STORE CRTN,B(P0+D#7):%,(P0+D#7):%
             10100000 01000000 01110100 00000111 11000000 00100011
```

```
 751   00171  ;**********************************************************
 752   00171  ;*     PROCEDURE DEEMPhasize;                              *
 753   00171  ;*  { This procedure deemphasizes data in the deemphasis  *
 754   00171  ;* filter memory.                                        }*
 755   00171  ;*                                                         *
 756   00171  ;**********************************************************
 757   00171  DEEM:     LDMR     DEMBS
              11100000 01010000 00000001 00111100 11000001 10000100
 758   00172            FILTERA ,A,(DEMBS+D#1):%
              11100000 01010000 00110001 00111101 11100010 01000100
 759   00173            FILTERN ,A,(DEMBS+D#1):%
              11100000 01010000 00110001 00111101 11100010 11000100
 760   00174            FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 761   00175            STMR     ,DEMBS
              11100000 00111111 11110001 00111100 11000000 00101010
 762   00176            FILTER1 ,GNOUT,DEMBS  ; Scale output to prevent overflow
              11100000 01010111 01100001 00111100 11100000 01000100
 763   00177            FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 764   00178            STMR     CRTN,(DEMBS+D#1):%
              10100000 00111111 11110001 00111101 11000000 00101010
 765   00179            END


  TOTAL ASSEMBLY ERRORS =     0
```

```
                          CROSS REFERENCE TABLE

LABEL     TYPE   VALUE         REFERENCES
A          A     00003     -106   107   758   759
ABSZ       A     00001      -65
ADC        A     00200        0
AUTO1      D                  0
AUTOL      D                  0
AUTON      D                  0
CALL       D                190   193   196   199   200   203   208   213
                            218   223   224   229   236   243   250   255
                            256   261   267   273   277   281   284   285
                            288   295   298   301   304   305   308   311
                            314   317   320   321   324   380   381   387
                            389
CBBF       A     00047     -109   110
CHPF       A     0004A     -110   111
CJP        A     00003        0
CJPP       A     0000B        0
CJB        A     00001        0
CLKP       A     00080        0
CLKXY      A     00040        0
CLPF       A     00004     -107   108   124   572   596   597   598   599
                            600   601   602   603   604   605   606   607
                            608   609   610   611   612   613   614   615
                            616   617   618   619   620   621   622   623
                            624   625   626   649   650   651   652   653
                            654   655   656   657   658   659   660   661
                            662   663   664   665   666   667   668   669
                            670   671   672   673   674   675   676   677
                            678   679
COEFB      A     00005      167   566   711
CON0       A     00000     -103   167   274   392
CON1       A     00001     -104   382   383   566   690   693
CON128     A     0006F     -116   117
CON1QRTR   A     0006B     -112
CON2       A     0006C     -113   114   545   549
CON4       A     0006D     -114   115   692
CON8       A     0006E     -115   116   716   720
CONM1      A     00002     -105   558   568
CONT       A     0000E        0
CPYPRED    A     00161      277  -734
CQMF       A     00023     -108   109   415   416   417   418   419   420
                            421   422   423   424   425   426   427   428
                            429   430   431   432   433   434   435   436
                            437   438   439   440   441   442   443   444
                            445   446   447   448   449   450   471   472
                            473   474   475   476   477   478   479   480
                            481   482   483   484   485   486   487   488
                            489   490   491   492   493   494   495   496
                            497   498   499   500   501   502   503   504
                            505   506
```

```
CROSEEN   A    0006B        0
CRTN      A    0000A      395   578   580   582   636   695   725   749
                          764
DDFSZ     A    00003      -64   110
DEEM      A    00171      389  -757
DEMBS     A    00130      -78    79    82   378   725   757   758   759
                          761   762   764
DEMSZ     A    00002      -58
DIR       A    00010        0
DLY       A    00052      -59    76
DLYBS     A    000BE      -73    74   554
EVERY     A    00056      190   193   196   199   203   208   213   218
                          223   229   236   243   250   255   261   267
                          273   281   284   286   295   298   301   304
                          308   311   314   317   320   324  -350   350
FALSE     A    00001        0
FE        A    00013      -93    94   359   359   412   412   468   468
FEPRT     A    00003     -132   357
FILTER1   D               167   274   382   392   415   471   545   558
                          562   572   596   630   649   683   690   703
                          716   723   762
FILTERA   D               758
FILTERL   D               168   275   358   384   393   451   507   546
                          550   559   564   569   573   627   631   635
                          680   684   688   694   712   717   721   724
                          760   763
FILTERN   D               383   416   417   418   419   420   421   422
                          423   424   425   426   427   428   429   430
                          431   432   434   435   436   437   438   439
                          440   441   442   443   444   445   446   447
                          448   449   450   472   473   474   475   476
                          477   478   479   480   481   482   483   484
                          485   486   487   488   489   490   491   492
                          493   494   495   496   497   498   499   500
                          501   502   503   504   505   506   568   597
                          598   599   600   601   602   603   604   605
                          606   607   608   609   610   611   612   613
                          614   615   616   617   618   619   620   621
                          622   623   624   625   626   650   651   652
                          653   654   655   656   657   658   659   660
                          661   662   663   664   665   666   667   668
                          669   670   671   672   673   674   675   676
                          677   678   679   693   704   705   706   707
                          708   709   710   759
FILTRAM   D                 0
FILTRAU   D               549   634   687   720
FILTRNK   D               433   692   711
GNDDF     A    00073     -120   121
GNHPF     A    00074     -121   122
GNINPUT   A    00070     -117   118   357
GNLPF     A    00072     -119   120   630   634   683   687
GNOUT     A    00076     -123   762
GNPRED    A    00075     -122   123   723
```

```
GNQMF     A     00071       -118   119
HFROUT    A     00163        -81
HPF       A     0011D        381  -649
HPFBS     A     00144        -80    81   372   383   649   650   651   652
                             653   654   655   656   657   658   659   660
                             661   662   663   664   665   666   667   668
                             669   670   671   672   673   674   675   676
                             677   678   679   689   690   692   693   695
HPFSZ     A     00021        -66   111   372
INPUT     D                  357   360
IO        A     0000C          0
IOPD      A     00000          0
J         A     00009        -54    91    92
JMP       D                  176   189   326   350   351   352   363   365
                             367   369   371   373   375   377   452   508
                             556   557   571   576
JRP       A     00007          0
JSRP      A     00005          0
JZ        A     00000          0
LDCT      A     0000C          0
LDMR      D                  354   370   372   374   376   378   548   575
                             633   686   719   757
LEAST     A     003FF        -83   543   549   551   562   572   574   575
                             626   634   681   687   714   720
LOAD      D                  204   209   214   219   225   230   232   237
                             239   244   246   251   257   262   268   364
                             366   368   412   468   577   579   581   734
                             736   738   740   742   744   746   748
LOOP      D                    0
LP        A     0000D          0
LPF       A     000F4        380  -596
LPFBS     A     000D1        -74    75   370   382   596   597   598   599
                             600   601   602   603   604   605   606   607
                             608   609   610   611   612   613   614   615
                             616   617   618   619   620   621   622   623
                             624   625   626   636
LPFSZ     A     0001F        -56    75    81   106   370   382   383   636
                             689   690   695
LSTFE     A     00017        -97    98   368   368   552   552   577   577
                             579   579   581   581
LSTFEH    A     00016        -96    97   366   366   414   414
LSTFEL    A     00015        -95    96   364   364   470   470
LSTF      A     00014        -94    95   220   220   226   226   233   233
                             240   240   247   247   252   252   258   258
                             264   264   270   270   276   276
MEMEN     A     00004          0
MOST      A     003FE        -85   544   545   547   548   553   558   560
                             568   570   629   630   632   633   682   683
                             685   686   715   716   718   719
NEG       A     00002        556   571   576
NEGATE    A     000DE        556  -558
NEWAUTO   A     01000          0
NOMEM     A     006FF          0
```

```
NOFERT   A   000F2      571  -581
NORMAL   A   00000        0
NOTE     A   0006B      369  -371
NOTFE    A   00066      367  -369
NOTFEH   A   00064      365  -367
NOTFEL   A   00062      363  -365
NOTHPF   A   0006A      371  -373
NOTS     A   00070      377  -379
NOTSPE   A   0006E      375  -377
NOTSUM   A   0006C      373  -375
OUTLP    A   00059      351  -354
OUTPUT   D              169   170   355   379
P        A   00012      -92    93   204   204   209   209   214   214
                        230   230   237   237   244   244   262   262
                        268   269   361   361
P0       A   00000      -90    91   703   704   705   706   707   708
                        709   710   735   735   737   737   739   739
                        740   741   741   743   743   745   745   747
                        747   749   749
PAI      A   00003        0
PERT     A   000D0      452   508  -541
PERTBS   A   00141      -79    80   394   578   580   582
PERTP    A   000F0      576  -579
POSTV    A   000E1      557  -561
PPRT     A   00002     -131   360
PRED     A   0014B      380  -703
PREDBS   A   00133      -76    77    78   376   703   704   705   706
                        707   708   709   710   711   722   723
PREDOUT  A   0013C      -77
PREDBCL  A   0005F     -111   112   113   711
PREDSZ   A   00009      -57    77    78
PREL     A   00100        0
PSH      A   00004        0
PTMPO    A   00009      -91    92   205   205   210   210   215   215
                        219   219   225   225   231   231   232   232
                        238   238   239   239   245   245   246   246
                        251   251   257   257   263   263   269   269
                        734   734   736   736   738   738   740   742
                        742   744   744   746   746   748   748
PUSH     D              171   291
QMFBS    A   00000      -72    73   395   413   415   416   417   418
                        419   420   421   422   423   424   425   426
                        427   428   429   430   431   432   433   434
                        435   436   437   438   439   440   441   442
                        443   444   445   446   447   448   449   450
                        469   471   472   473   474   475   476   477
                        478   479   480   481   482   483   484   485
                        486   487   488   489   490   491   492   493
                        494   495   496   497   498   499   500   501
                        502   503   504   505   506
QMFH     A   0007E      224   285   321  -412
QMFL     A   000A7      200   256   305  -468
QMFSZ    A   00024      -55    73   109
```

```
REPLP     D              175   325
RET       D                0
RFCT      A    00006       0
RINGB     A    00000       0
RPCT      A    00009       0
S         A    0013E     -82   354
SCRAT     A    00004     566   703   704   705   706   707   708   709
                         710
SHFTCLK   A    00800       0
SHFTLFT1  D                0
SHFTLFT2  D                0
SOFENT    A    0000A     176  -189   189   326
SPL       D              172   173   391   566
SPRT      A    00002    -133   169   169   355   355
STLR      D              385   543   547   551   560   570   574   628
                         632   681   685   695   714   718
STLS      D              359
STMR      D              174   394   395   413   469   544   553   554
                         578   580   582   629   636   682   689   715
                         722   725   761   764
STMS      D              205   210   215   220   226   231   233   238
                         240   245   247   252   258   263   264   269
                         270   276   361   552   565
STORE     D              414   470   735   737   739   741   743   745
                         747   749
SUP       A    00400       0
SUMBS     A    000E1     -75    76   374   385
TC        A    02000       0
TEMP      A    0001B     -98   565   565   566
TEST      A    00001    -130   170   170   379   379
THRESH    A    00005    -124   562
TRUE      A    00000       0
TSL       A    00001       0
TSK       A    00002       0
TSTE      A    0000B    -140   369
TSTFE     A    0000A    -139   367
TSTFEH    A    00009    -138   365
TSTFEL    A    00008    -137   363
TSTHFF    A    0000C    -141   371
TSTS      A    0000F    -144   377
TSTSPE    A    0000E    -143   375
TSTSUM    A    0000D    -142   373
TWB       A    0000F       0
WAITLP    A    00057    -351   352
WRITE     A    00020       0
.LD       A    00184       0
.MARK     A    01000       0
.MULT1    A    02044     357   566
.MULTA    A    02244       0
.MULTAM   A    02644       0
.MULTAU   A    00244       0
.MULTL    A    020C0       0
.MULTN    A    022C4       0
```

```
.MULTRM   A     02604          0
.NOP      A     00000          0
.SENBL    A     00005       -134   350   351
.SHIFT    A     00800        172   391
.SOF      A     00007        189
.STL      A     00039          0
.STM      A     0002A          0
.STR      A     00023          0
.TEST0    A     00008        137
.TEST1    A     00009        138
.TEST2    A     0000A        139
.TEST3    A     0000B        140
.TEST4    A     0000C        141
.TEST5    A     0000D        142
.TEST6    A     0000E        143
.TEST7    A     0000F        144
```

A2. RESIDUAL REGENERATION THROUGH INTERPOLATION OF LOW PASS COMPONENT BY ZERO AMPLITUDE INSERTION, AND SAMPLE POSITION PERTURBATION

```
00000 TITLE SYNTHESIS PROCESSOR ASSEMBLER SOURCE FILE. SEPT 13,1982
00000        LIST X
00000 ;**************************************************************
00000 ;*          PROJECT:     472A, RELP CODEC                      *
00000 ;*          BOARD:       SYNTHESIS PROCESSOR                   *
00000 ;*          DEVICE:      6 MICROCODE PROMS, 3636 or 3628       *
00000 ;*          LOCATION:    G34,P1,P15,P29,P43,P57                *
00000 ;*          PROG.DEV#:   60-0690  -  60-0695                   *
00000 ;*          ASSEMBLER:   META, April 82                        *
00000 ;*          FILE:        SYNTHESIS.ASM                         *
00000 ;*          REVISION:    SEPT 13,1982                          *
00000 ;*          UPDATE TABLE:                                      *
00000 ;*               OCT  4:   HPF and LPF removed.                *
00000 ;*               SEPT 9:   Initial folding exp. hopeful        *
00000 ;*                         Higgins method implemented.         *
00000 ;*               AUG 3:    Aliasing high freq regeneration     *
00000 ;*               JULY 7:   Each filter assigned gain in PROM   *
00000 ;*               JULY 2:   More precision maintained.          *
00000 ;*               JUNE 28:  Filter Gains Adjusted.              *
00000 ;*               JUNE 14:  Mods made to use subtract inst corr *
00000 ;*               APRIL 23 - MAY 23: Initial progam entry JM    *
00000 ;*                                                             *
00000 ;**************************************************************
```

```
00000  ;***************************************************************
00000  ;*          TABLE OF CONTENTS                                  *
00000  ;*                                                             *
00000  ;*    PAGE      DESCRIPTION                                    *
00000  ;*   -------------------------------------------------        *
00000  ;*      1       File Header and Update Table                   *
00000  ;*      2       Table of Contents                              *
00000  ;*      3       Filter size specifications                     *
00000  ;*      4       Ring Buffer memory map                         *
00000  ;*      5       Scratch Pad memory map                         *
00000  ;*      6       Coefficient Prom memory map                    *
00000  ;*      7       Input/Output definitions                       *
00000  ;*      8       PROCEDURE start_up;                            *
00000  ;*      9       PROCEDURE main;                                *
00000  ;*     14       PROCEDURE operations_done_EVERY_sample         *
00000  ;*     17       PROCEDURE Quadrature_Mirror_Filter_High        *
00000  ;*     20       PROCEDURE Quadrature_Mirror_Filter_Low         *
00000  ;*     23       PROCEDURE Low_Pass_Filter                      *
00000  ;*     25       PROCEDURE Double_Difference_Filter             *
00000  ;*     26       PROCEDURE High_Pass_Filter                     *
00000  ;*     28       PROCEDURE Predict                              *
00000  ;*     29       PROCEDURE CoPY_PREDictor_Coefficients          *
00000  ;*     30       PROCEDURE DE-EMphasize                         *
00000  ;*     31       Cross Reference Table                          *
00000  ;***************************************************************
```

```
 51  00000 ;*************************************************************
 52  00000 ;*          FILTER SIZE SPECIFICATIONS                        *
 53  00000 ;*                                                           *
 54  00000 ;*************************************************************
 55  00000 J:      EQU    9     ; Predictor filter order.
 56  00000 QMFSZ:  EQU   36     ; Quadrature Mirror Filter Size
 57  00000 LPFSZ:  EQU   31     ; Low pass filter size
 58  00000 PREDSZ: EQU    9     ; Predictor Filter Size
 59  00000 DEMSZ:  EQU    2     ; Deemphasis Filter Size
 60  00000 DLY:    EQU  100     ; Delay so predictors line up with frame
 61  00000 ; The following constants define filter sizes in the rectification
 62  00000 ; RELP implementation.  They are used here only to specify the location
 63  00000 ; the filters in the coefficient PROM, which is the same for both
 64  00000 ; the perturbation and rectication implementations.
 65  00000 DDFSZ:  EQU    3
 66  00000 ABSZ:   EQU    1
 67  00000 HPFSZ:  EQU   33
```

```
00000  ;****************************************************************
00000  ;*           RING BUFFER MEMORY MAP                             *
00000  ;*                                                              *
00000  ;****************************************************************
00000  QMFBS:  EQU   10(0):% ; Base of quadrature mirror filter
00000  DLYBS:  EQU   10(QMFBS+5*QMFSZ+10):% ; Compensating delay for PERT
00000  LPFBS:  EQU   10(DLYBS+D#3):%          ; Low pass filter base
00000  SUMBS:  EQU   10(LPFBS+LPFSZ+D#1):%  ; summing node for HPF and LPF path
00000  PREDBS: EQU   10(LPFBS+DLY):%          ; Predictor base(DLY IS MATCHING DE
00000  PREDOUT:EQU   10(PREDBS+PREDSZ)    ; End of predictor
00000  DEMBS:  EQU   10(PREDBS+PREDSZ):% ; Deemphasis base
00000  PERTBS: EQU   10(DEMBS+D#5):%      ; Base of compensating delay buffer
00000  HPFBS:  EQU   10(PERTBS+D#3):%     ; End of compensating delay
00000  HFROUT: EQU   10(HPFBS+LPFSZ):%    ; Output of High Freq Regeneration
00000  S:      EQU   10(DEMBS+2):%        ; Synthesized speech location
00000  LEAST:  EQU   10(D#1023):%         ; Temp location used in double prec
00000                                     ; arithmatic
00000  MOST:   EQU   10(D#1022):%         ; As above
```

```
00000  ;***********************************************************
00000  ;*    SCRATCH PAD MEMORY MAP                                *
00000  ;***********************************************************
00000  P0:      EQU    8(0);%                    ; base of predictor storage
00000  PTMP0:   EQU    8(P0+J);%                 ; Base of temporary predictor storage
00000  P:       EQU    8(PTMP0+J);%              ; Last predictor coef. input
00000  FE:      EQU    8(P+1);%                  ; FE(H,L) storage
00000  LSTP:    EQU    8(FE+1);%                 ; Last P used
00000  LSTFEL:  EQU    8(LSTP+1);%               ; Last FEL input
00000  LSTFEH:  EQU    8(LSTFEL+1);%             ; Last FEH input
00000  LSTFE:   EQU    8(LSTFEH+1);%             ; Last FE calculated
00000  TEMP:    EQU    8(LSTFE+1);%              ; Temporary storage location
```

```
00000 ;************************************************************
00000 ;*     COEFFICIENT PROM MEMORY MAP                          *
00000 ;************************************************************
00000 CON0:   EQU   B(0):%              ; constant zero
00000 CON1:   EQU   B(1):%              ; constant one
00000 CONM1:  EQU   B(2):%              ; constant minus one
00000 A:      EQU   B(3):%              ; Pre-emphasis constant/2
00000 CLPF:   EQU   B(A+1):%            ; Base of low pass filter coef
00000 CQMF:   EQU   B(CLPF+LPFSZ):%     ; Base of quadrature mirror filter
00000 CDDF:   EQU   B(CQMF+QMFSZ):%     ; Base oF double difference filter
00000 CHPF:   EQU   B(CDDF+DDFSZ):%     ; Base of high pass filter
00000 PREDSCL:EQU   B(CHPF+HPFSZ):%     ; Scale factor for predictor
00000 CON1QRTR: EQU B(PREDSCL):%        ; Scale facter is 1/4
00000 CON2:   EQU   B(PREDSCL+1):%
00000 CON4:   EQU   B(CON2+1):%
00000 CON8:   EQU   B(CON4+1):%
00000 CON128: EQU   B(CON8+1):%
00000 GNINPUT:EQU   B(CON128+1):%
00000 GNQMF:  EQU   B(GNINPUT+1):%      ; Filter gains follow
00000 GNLPF:  EQU   B(GNQMF+1):%
00000 GNDDF:  EQU   B(GNLPF+1):%
00000 GNHPF:  EQU   B(GNDDF+1):%
00000 GNPRED: EQU   B(GNHPF+1):%
00000 GNOUT:  EQU   B(GNPRED+1):%
00000 THRESH: EQU   B(GNOUT):%         ; = 526, USED TO AVOID BURNING
00000                                   ; PROm BEFORE FINDING OPTIMUM VALUE
```

```
 128    00000  ;*************************************************************
 129    00000  ;*   INPUT/OUTPUT DEFINITIONS                                   *
 130    00000  ;*************************************************************
 131    00000  TEST:   EQU B#01        ; Analog test port address
 132    00000  PPRT:   EQU B#10        ; Predictor input port
 133    00000  FEPRT:  EQU B#11        ; FE input port
 134    00000  SPRT:   EQU B#10        ; Synthesized speech output port
 135    00000  .SENBL: EQU 4H#5        ; S enable test input address

 137    00000  ; Test switch input definitions
 138    00000  TSTFEL: EQU .TEST0                      ; Test point select 0 = FEL
 139    00000  TSTFEH: EQU .TEST1                      ; Test point select 1 = FEH
 140    00000  TSTFE:  EQU .TEST2                      ; Test point select 2 = FE
 141    00000  TSTE:   EQU .TEST3                      ; Test point select 3 = E
 142    00000  TSTHPF: EQU .TEST4                      ; Test point select 4 = HPF
 143    00000  TSTSUM: EQU .TEST5                      ; Test point select 5 = SUM HPF & E
 144    00000  TSTSPE: EQU .TEST6                      ; Test point select 6 = SPE
 145    00000  TSTS:   EQU .TEST7                      ; Test point select 7 = S
```

```
147   00000  ;**************************************************************
148   00000  ;*   PROCEDURE start_up;                                       *
149   00000  ;*  This procedure is executed on power up.  It initializes*
150   00000  ;* the signal processor and waits a sufficient time to       *
151   00000  ;* allow the rest of the system to start functioning before*
152   00000  ;* processing.  This prevents noise from being output       *
153   00000  ;* during the power up sequence.                             *
154   00000  ;*                                                           *
155   00000  ;*   BEGIN                                                   *
156   00000  ;*     Load the Accumulator with zero                        *
157   00000  ;*     Zero sample output port so no signal output during    *
158   00000  ;*            power up sequence.                             *
159   00000  ;*     FOR i := 0 TO 4095 DO                                 *
160   00000  ;*       shift data in ring buffer                           *
161   00000  ;*       Store Accumulator in ring buffer location zero      *
162   00000  ;*     ENDFOR                                                *
163   00000  ;*     Start normal processing loop at start_of_frame entry*
164   00000  ;*   END;                                                    *
165   00000  ;*                                                          *
166   00000  ;**************************************************************
167   00000          ORG     0
168   00000          FILTER1 COEFB,CON0,0:%  ; Zero the Accumulator
             11100000 01010000 00000000 00000000 11100000 01000100
169   00001          FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
170   00002          OUTPUT  SPRT,SPRT        ; Zero the sample output port
             11100000 01100100 00000110 01000000 11000000 00100011
171   00003          OUTPUT  TEST,TEST        ; Zero the analog test point
             11100000 01100010 00000110 00100000 11000000 00100011
172   00004          PUSH    (D#4095):%        ; FOR i := 0 TO 4095 DO
             01000000 11111111 11110110 11111111 11000000 00000000
173   00005          SPL     ,,,,.SHIFT       ;   shift data in ring buffer
             11100000 01101111 11110110 11111111 11001000 00000000
174   00006          SPL                      ;   NOP, wait for shift to complete
             11100000 01101111 11110110 11111111 11000000 00000000
175   00007          STHF    ,0:%             ;   store zero in ring buf
             11100000 00111111 11110000 00000000 11000000 00101010
176   00008          REPLF                    ; ENDFOR
             10000000 01101111 11110110 11111111 11000000 00000000
177   00009          JMP     ,(SOFENT):%      ; Jump into main prodedure at
             00110000 00000000 10100110 11111111 11000000 00000000
178   0000A                                   ; Start_of_Frame entry point
```

```
180   0000A  ;*********************************************************
181   0000A  ;*      PROCEDURE main;                                   *
182   0000A  ;* { This procedure does the calls for all filtering     *
183   0000A  ;* operations in the synthesis processor, as well as all the*
184   0000A  ;* input.  All input and output operations are done after   *
185   0000A  ;* S enable goes low but before S clock goes high. Sample  *
186   0000A  ;* output takes place after S enable goes high and all     *
187   0000A  ;* calculations for the sample interval are complete
188   0000A  ;*********************************************************
189   0000A  ;  Start of sample interval 0
190   0000A  SOFENT:  JMP  .SOF,SOFENT:%       ; Wait for start of frame
             00110111 00000000 10100110 11111111 11000000 00000000
191   0000B          CALL ,EVERY:%            ; Do the stuff done every sample
             00010000 00000101 01100110 11111111 11000000 00000000

193   0000C  ;  Start of sample interval 1
194   0000C          CALL ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

196   0000D  ;  Start of sample interval 2
197   0000D          CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

199   0000E  ;  Start of sample interval 3, set FEL
200   0000E          CALL  ,EVERY:%         ; Do stuff done every frame
             00010000 00000101 01100110 11111111 11000000 00000000
201   0000F          CALL  ,QMFL:%          ; Do Low QMF filter stuff
             00010000 00001010 00010110 11111111 11000000 00000000

203   00010  ;  Start of sample interval 4, set P(0)
204   00010          CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
205   00011          LOAD  P,P                ; Input P(0)
             11100000 01000001 00100100 00010010 11000001 10000100
206   00012          STMS  PTMP0,PTMP0        ; Store in PTMP0
             11100000 01000000 10010100 00001001 11000000 00101010

208   00013  ;  Start of sample interval 5, set P(1)
209   00013          CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
210   00014          LOAD  P,P                ; Input P(1)
             11100000 01000001 00100100 00010010 11000001 10000100
211   00015          STMS  (PTMP0+D#1):%,(PTMP0+D#1):% ; Store in Ptmp1
             11100000 01000000 10100100 00001010 11000000 00101010

213   00016  ;  Start of sample interval 6, set P(2)
214   00016          CALL  ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
215   00017          LOAD  P,P                ; Input P(2)
             11100000 01000001 00100100 00010010 11000001 10000100
216   00018          STMS (PTMP0+D#2):%,(PTMP0+D#2):% ; Store in Ptmp2
             11100000 01000000 10110100 00001011 11000000 00101010
```

```
 218   00019  ;  Start of sample interval 7
 219   00019           CALL ,EVERY:%
              00010000 00000101 01100110 11111111 11000000 00000000
 220   0001A           LOAD PTMP0,PTMP0        ; Get P(0) for the test port
              11100000 01000000 10010100 00001001 11000001 10000100
 221   0001B           STMS LSTP,LSTP          ; Save it
              11100000 01000001 01000100 00010100 11000000 00101010

 223   0001C  ;  Start of sample interval 8, Get FEH
 224   0001C           CALL  ,EVERY:%,
              00010000 00000101 01100110 11111111 11000000 00000000
 225   0001D           CALL  ,QMFH:%             ; Do QMF for FEH input
              00010000 00000111 10000110 11111111 11000000 00000000
 226   0001E           LOAD (PTMP0+D#0):%,(PTMP0+D#0):% ; Get P(0) for test point
              11100000 01000000 10010100 00001001 11000001 10000100
 227   0001F           STMS LSTP,LSTP
              11100000 01000001 01000100 00010100 11000000 00101010

 229   00020  ;  Start of sample interval 9, set P(3)
 230   00020           CALL  ,EVERY:%
              00010000 00000101 01100110 11111111 11000000 00000000
 231   00021           LOAD  P,P                 ; Input P(3)
              11100000 01000001 00100100 00010010 11000001 10000100
 232   00022           STMS (PTMP0+D#3):%,(PTMP0+D#3):% ; Store in Ptmp3
              11100000 01000000 11000100 00001100 11000000 00101010
 233   00023           LOAD (PTMP0+D#1):%,(PTMP0+D#1):% ; Get P(1) for test point
              11100000 01000000 10100100 00001010 11000001 10000100
 234   00024           STMS LSTP,LSTP
              11100000 01000001 01000100 00010100 11000000 00101010

 236   00025  ;  Start of sample interval 10, set P(4)
 237   00025           CALL  ,EVERY:%
              00010000 00000101 01100110 11111111 11000000 00000000
 238   00026           LOAD  P,P                 ; Input P(4)
              11100000 01000001 00100100 00010010 11000001 10000100
 239   00027           STMS (PTMP0+D#4):%,(PTMP0+D#4):% ; Store in Ptmp4
              11100000 01000000 11010100 00001101 11000000 00101010
 240   00028           LOAD (PTMP0+D#2):%,(PTMP0+D#2):% ; Get P(2) for the test point
              11100000 01000000 10110100 00001011 11000001 10000100
 241   00029           STMS LSTP,LSTP
              11100000 01000001 01000100 00010100 11000000 00101010

 243   0002A  ;  Start of sample interval 11, set P(5)
 244   0002A           CALL  ,EVERY:%
              00010000 00000101 01100110 11111111 11000000 00000000
 245   0002B           LOAD  P,P                 ; Input P(5)
              11100000 01000001 00100100 00010010 11000001 10000100
 246   0002C           STMS  (PTMP0+D#5):%,(PTMP0+D#5):% ; Store in Ptmp1
              11100000 01000000 11100100 00001110 11000000 00101010
 247   0002D           LOAD  (PTMP0+D#3):%,(PTMP0+D#3):% ; Get P(3) for test point
              11100000 01000000 11000100 00001100 11000001 10000100
 248   0002E           STMS  LSTP,LSTP
              11100000 01000001 01000100 00010100 11000000 00101010
 250    0002F  ;  Start of sample interval 12
 251    0002F          CALL   ,EVERY:%
               00010000 00000101 01100110 11111111 11000000 00000000
 252    00030          LOAD  (PTMP0+D#4):%,(PTMP0+D#4):% ; Get P(4) for test point
               11100000 01000000 11010100 00001101 11000001 10000100
 253    00031          STMS  LSTP,LSTP
               11100000 01000001 01000100 00010100 11000000 00101010

 255    00032  ;  Start of sample interval 13, GET FEL
 256    00032          CALL   ,EVERY:%
               00010000 00000101 01100110 11111111 11000000 00000000
 257    00033          CALL   ,QMFL:%                ; Put the QMF filter
               00010000 00001010 00010110 11111111 11000000 00000000
 258    00034          LOAD (PTMP0+D#5):%,(PTMP0+D#5):% ; Get P(5) for test point
               11100000 01000000 11100100 00001110 11000001 10000100
 259    00035          STMS LSTP,LSTP
               11100000 01000001 01000100 00010100 11000000 00101010

 261    00036  ;  Start of sample interval 14, set P(6)
 262    00036          CALL   ,EVERY:%
               00010000 00000101 01100110 11111111 11000000 00000000
 263    00037          LOAD  P,P                     ; Input P(6)
               11100000 01000001 00100100 00010010 11000001 10000100
 264    00038          STMS  (PTMP0+D#6):%,(PTMP0+D#6):%
               11100000 01000000 11110100 00001111 11000000 00101010
 265    00039          STMS  LSTP,LSTP
               11100000 01000001 01000100 00010100 11000000 00101010

 267    0003A  ;  Start of sample interval 15, Get P(7)
 268    0003A          CALL   ,EVERY:%
               00010000 00000101 01100110 11111111 11000000 00000000
 269    0003B          LOAD  P,P                     ; Input P(7)
               11100000 01000001 00100100 00010010 11000001 10000100
 270    0003C          STMS  (PTMP0+D#7):%,(PTMP0+D#7):% ; Store in Ptemp7
               11100000 01000001 00000100 00010000 11000000 00101010
 271    0003D          STMS  LSTP,LSTP
               11100000 01000001 01000100 00010100 11000000 00101010

 273    0003E  ;  Start of sample interval 16
 274    0003E          CALL   ,EVERY:%
               00010000 00000101 01100110 11111111 11000000 00000000
 275    0003F          FILTER1 ,CONO,10(0):%       ; Load zero into Acc
               11100000 01010000 00000000 00000000 11100000 01000100
 276    00040          FILTERL
               11100000 01101111 11110110 11111111 11100000 11000000
 277    00041          STMS  LSTP,LSTP               ; Set test point predictor to zero
               11100000 01000001 01000100 00010100 11000000 00101010
 278    00042          CALL   ,CPYPRED:%             ; Copy predictors from temporary buff
               00010000 00010101 10100110 11111111 11000000 00000000
 279    00043                                        ; to the active area

 281    00043  ;  Start of sample interval 17
 282    00043          CALL   ,EVERY:%
               00010000 00000101 01100110 11111111 11000000 00000000
```

```
284   00044  ; Start of sample interval 18, set FEH
285   00044            CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
286   00045            CALL   ,QMFH:%             ; Quadrature mirror filter
             00010000 00000111 10000110 11111111 11000000 00000000

288   00046  ; Start of sample interval 19
289   00046            CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

291   00047  ; FOR n := 0 TO 13 DO
292   00047            PUSH  D#13:%               ; Define next instruction
             01000000 00000000 11010110 11111111 11000000 00000000
293   00048                                       ; as the start of loop
294   00048                                       ; execute loop 14 times
295   00048   ;  Start of sample interval 20 + 10n
296   00048              CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

298   00049   ;  Start of sample interval 21 + 10n
299   00049              CALL ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

301   0004A   ;  Start of sample interval 22 + 10n
302   0004A              CALL ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

304   0004B   ;  Start of sample interval 23 + 10n, input FEL
305   0004B              CALL ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
306   0004C              CALL   ,QMFL:%         ; Quadrature mirror filter low
             00010000 00001010 00010110 11111111 11000000 00000000

308   0004D  ;  Start of sample interval 24 + 10n
309   0004D              CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

311   0004E  ;  Start of sample interval 25 + 10n
312   0004E              CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

314   0004F  ;  Start of sample interval 26 + 10n
315   0004F              CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

317   00050  ;  Start of sample interval 27 + 10n
318   00050              CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000

320   00051  ;  Start of sample interval 28 + 10n, Input FEH
321   00051              CALL   ,EVERY:%
             00010000 00000101 01100110 11111111 11000000 00000000
322   00052              CALL   ,QMFH:%    ; Input and filter FEH
             00010000 00000111 10000110 11111111 11000000 00000000
```

```
 324   00053  ;  Start of sample interval 29 + 10n
 325   00053               CALL  ,EVERY:%
              00010000 00000101 01100110 11111111 11000000 00000000
 326   00054          REPLP                   ; ENDFOR
              10000000 01101111 11110110 11111111 11000000 00000000
 327   00055          JMP ,SOFENT:%           ; Do everything again for next frame
              00110000 00000000 10100110 11111111 11000000 00000000
```

```
329   00056  ;*******************************************************************
330   00056  ;*        PROCEDURE operations_done_EVERY_sample_interval          *
331   00056  ;*  { This procedure performs all the operations which are        *
332   00056  ;* performed every sample interval.  All the input ports are *
333   00056  ;* read immediatly after SCLCK goes low.  The required input *
334   00056  ;* data is transfered to working memory in the main procedure*
335   00056  ;* }                                                               *
336   00056  ;*     Wait for rising edge of S enable complement.               *
337   00056  ;*     Output S;                                                   *
338   00056  ;*     Input FE,P;                                                 *
339   00056  ;*     Read test point switch and output appropriate data;        *
340   00056  ;*     Move data to the delay buffer;                              *
341   00056  ;*     Low pass the reconstructed residual;                        *
342   00056  ;*     Perturb and high pass the reconstructed residual;           *
343   00056  ;*     Sum the high and low pass residuals;                        *
344   00056  ;*     Pass data through the predictor                             *
345   00056  ;*     Preemphasiz the data;                                       *
346   00056  ;*     Shift the ring buffer;                                      *
347   00056  ;*     Store zero in the ring buffer base; {required for inter*
348   00056  ;*                                                 polation.     }*
349   00056  ;*     END;                                                        *
350   00056  ;*******************************************************************
351   00056  EVERY:    JMP   .SENBL,EVERY!% ; Wait for .S Enable to be low
             00110101 00000101 01100110 11111111 11000000 00000000
352   00057  WAITLP:   JMP   .SENBL,OUTLP!% ; Wait for .S Enable to be high
             00110101 00000101 10010110 11111111 11000000 00000000
353   00058            JMP   ,WAITLP!%
             00110000 00000101 01110110 11111111 11000000 00000000
354   00059  ;  Now output S
355   00059  OUTLP:    LDMR S                  ; Load S into the Acc
             11100000 01010000 00000001 00110000 11000001 10000100
356   0005A            OUTPUT SPRT,BPRT  ; Output S
             11100000 01100100 00000110 01000000 11000000 00100011
357   0005B  ;  Now input from both input ports in case we need to read them
358   0005B            INPUT  GNINPUT,FEPRT,.MULT1 ; Get FEL or FEH {depending on samp
             11100000 01010111 00000110 01100000 11100000 01000100
359   0005C            FILTERL                 ; And double it
             11100000 01101111 11110110 11111111 11100000 11000000
360   0005D            STLS   ,FE,FE      ; Store it away for possible later use
             11100000 01000001 00110100 00010011 11000000 00111001
361   0005E            INPUT  ,PPRT,      ; Input a predictor
             11100000 01010000 00000110 01000000 11000001 10000100
362   0005F            STMS   P,P         ; Store it away for possible later use
             11100000 01000001 00100100 00010010 11000000 00101010
363   00060  ;  Test the 'TEST POINT SELECT' switch and output the data to test port
364   00060            JMP  TSTFEL,NOTFEL!% ; Test if FEL to be output
             00111000 00000110 00100110 11111111 11000000 00000000
365   00061            LOAD LSTFEL,LSTFEL   ;   Yes - Load FEL into Acc
             11100000 01000001 01010100 00010101 11000001 10000100
366   00062  NOTFEL:  JMP  TSTFEH,NOTFEH!% ; Test if FEH to be output
             00111001 00000110 01000110 11111111 11000000 00000000
367   00063            LOAD LSTFEH,LSTFEH   ;   Yes - Load FEH into Acc
             11100000 01000001 01100100 00010110 11000001 10000100
```

```
 368   00064  NOTFEH:  JMP  TSTFE,NOTFE:%    ; Test if FE to be output
              00111010 00000110 01100110 1111111 11000000 00000000
 369   00065           LOAD  LSTFE,LSTFE    ;  Yes - Load FE into Acc
              11100000 01000001 01110100 00010111 11000001 10000100
 370   00066  NOTFE:   JMP  TSTE,NOTE:%       ; Test if AEDD to be output
              00111011 00000110 10000110 11111111 11000000 00000000
 371   00067           LDMR  (LPFBS+LPFSZ+D#1):% ;  Yes - Load AEDD into Acc
              11100000 01010000 00000000 11100001 11000001 10000100
 372   00068  NOTE:    JMP  TSTHPF,NOTHPF:% ; Test if HPF to be output
              00111100 00000110 10100110 11111111 11000000 00000000
 373   00069           LDMR  (HPFBS+HPFSZ+D#1):% ;   Yes - output HPF
              11100000 01010000 00000001 01011000 11000001 10000100
 374   0006A  NOTHPF:  JMP  TSTSUM,NOTSUM:% ; Test if regenerated residual to be outpu
              00111101 00000110 11000110 11111111 11000000 00000000
 375   0006B           LDMR (SUMBS+D#1):%   ;   Yes - output sum
              11100000 01010000 00000000 11100010 11000001 10000100
 376   0006C  NOTSUM:  JMP  TSTSPE,NOTSPE:% ; Test if SPE to be output
              00111110 00000110 11100110 11111111 11000000 00000000
 377   0006D           LDMR  (PREDBS+D#1):%  ;   Yes - Load SPE into Acc
              11100000 01010000 00000001 00100110 11000001 10000100
 378   0006E  NOTSPE:  JMP  TSTS,NOTS:%       ; Test of S to be output
              00111111 00000111 00000110 11111111 11000000 00000000
 379   0006F           LDMR  (DEMBS+D#2):%  ;   Yes - Load S into Acc
              11100000 01010000 00000001 00110000 11000001 10000100
 380   00070  NOTS:    OUTPUT TEST,TEST     ; Send the value to the analog test port
              11100000 01100010 00000110 00100000 11000000 00100011
 381   00071  ;        CALL  ,LPF:%            ; Low pass the unperturbed samples
 382   00071  ;        CALL    ,HPF:%          ; High pass the perturbed samples
 383   00071  ;        FILTER1 ,CON1,(LPFBS+LPFSZ):% ; add the HPF and LPF signals
 384   00071  ;        FILTERN ,CON1,(HPFBS+LPFSZ):%
 385   00071  ;        FILTERL
 386   00071  ;        STLR    ,SUMBS
 387   00071  ; Now pass the data through the predictor
 388   00071           CALL    ,PRED:%
              00010000 00010100 01000110 11111111 11000000 00000000
 389   00072  ; Now deemphasize the data and shift ring buffer data
 390   00072           CALL    ,DEEM:%
              00010000 00010110 10100110 11111111 11000000 00000000
 391   00073  ; Now load zero into base of ring buffer
 392   00073           SPL     ,,,,.SHIFT   ; Shift ring buffer data
              11100000 01101111 11110110 11111111 11001000 00000000
 393   00074           FILTER1  ,CON0,10(0):% ;
              11100000 01010000 00000000 00000000 11100000 01000100
 394   00075           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 395   00076           STMR    ,(PERTBS-1):%
              11100000 00111111 11110001 00110010 11000000 00101010
 396   00077           STMR    CRTN,QMFBS ; Store zero and return
              10100000 00111111 11110000 00000000 11000000 00101010
```

```
398   00078  ;*****************************************************************
399   00078  ;*    PROCEDURE Quadrature Mirror Filter High                    *
400   00078  ;*  This  procedure performs a quadrature mirror filter operation*
401   00078  ;* assuming that FEH is in the FE input buffer.  It placc FEH at  *
402   00078  ;* the start of the QMF portion of the ring buffer.               *
403   00078  ;*                                                               *
404   00078  ;* sum := FEH[n] * CQMF[0]                                       *
405   00078  ;* FOR i := 1 TO 17 DO                                           *
406   00078  ;*   sum := sum + FEH[n + 2*i] * CQMF[2*i];                       *
407   00078  ;* sum := FEL[n] * QMF[0] - sum;
408   00078  ;* FOR i := 0 TO 17 DO                                           *
409   00078  ;*   sum := sum + FEL[n + 2*i] * CQMF[2*i];                       *
410   00078  ;* LPFBS[0] := sum;                                              *
411   00078  ;* END;                                                          *
412   00078  ;*****************************************************************
413   00078  QMFH:     LOAD     FE,FE                    ; Fetch FEL from the FE input b
             11100000 01000001 00110100 00010011 11000001 10000100
414   00079            STMF     ,QMFBS                  ; Store at base of QMF ring buf
             11100000 00111111 11110000 00000000 11000000 00101010
415   0007A            STORE    ,LSTFEH,LSTFEH          ; Store for the test point
             11100000 01000001 01100100 00010110 11000000 00100011
416   0007B            FILTERN  ,(CQMF+D#0):%,(QMFBS+D#0):%  ; Do the FEH terms first
             11100000 01010010 00110000 00000000 11100000 01000100
417   0007C            FILTERN  ,(CQMF+D#2):%,(QMFBS+D#10):%
             11100000 01010010 01010000 00001010 11100010 11000100
418   0007D            FILTERN  ,(CQMF+D#4):%,(QMFBS+D#20):%
             11100000 01010010 01110000 00010100 11100010 11000100
419   0007E            FILTERN  ,(CQMF+D#6):%,(QMFBS+D#30):%
             11100000 01010010 10010000 00011110 11100010 11000100
420   0007F            FILTERN  ,(CQMF+D#8):%,(QMFBS+D#40):%
             11100000 01010010 10110000 00101000 11100010 11000100
421   00080            FILTERN  ,(CQMF+D#10):%,(QMFBS+D#50):%
             11100000 01010010 11010000 00110010 11100010 11000100
422   00081            FILTERN  ,(CQMF+D#12):%,(QMFBS+D#60):%
             11100000 01010010 11110000 00111100 11100010 11000100
423   00082            FILTERN  ,(CQMF+D#14):%,(QMFBS+D#70):%
             11100000 01010011 00010000 01000110 11100010 11000100
424   00083            FILTERN  ,(CQMF+D#16):%,(QMFBS+D#80):%
             11100000 01010011 00110000 01010000 11100010 11000100
425   00084            FILTERN  ,(CQMF+D#18):%,(QMFBS+D#90):%
             11100000 01010011 01010000 01011010 11100010 11000100
426   00085            FILTERN  ,(CQMF-D#20):%,(GMFBS+D#100):%
             11100000 01010011 01110000 01100100 11100010 11000100
427   00086            FILTERN  ,(CQMF+D#22):%,(QMFBS+D#110):%
             11100000 01010011 10010000 01101110 11100010 11000100
428   00087            FILTERN  ,(CQMF+D#24):%,(QMFBS+D#120):%
             11100000 01010011 10110000 01111000 11100010 11000100
429   00088            FILTERN  ,(CQMF+D#26):%,(QMFBS+D#130):%
             11100000 01010011 11010000 10000010 11100010 11000100
430   00089            FILTERN  ,(CQMF+D#28):%,(QMFBS+D#140):%
             11100000 01010011 11110000 10001100 11100010 11000100
431   0008A            FILTERN  ,(CQMF+D#30):%,(QMFBS+D#150):%
             11100000 01010100 00010000 10010110 11100010 11000100
```

```
 432   0008B              FILTERN  ,(CQMF+D#32):%,(QMFBS+D#160):%
                11100000 01010100 00110000 10100000 11100010 11000100
 433   0008C              FILTERN  ,(CQMF+D#34):%,(QMFBS+D#170):%
                11100000 01010100 01010000 10101010 11100010 11000100
 434   0008D              FILTRNM  ,(CQMF+D#00):%,(QMFBS+D#5):% ;sum:=FEL[0]+QMF[0]-sum
                11100000 01010010 00110000 00000101 11100110 11000100
 435   0008E              FILTERN  ,(CQMF+D#02):%,(QMFBS+D#15):%
                11100000 01010010 01010000 00001111 11100010 11000100
 436   0008F              FILTERN  ,(CQMF+D#04):%,(QMFBS+D#25):%
                11100000 01010010 01110000 00011001 11100010 11000100
 437   00090              FILTERN  ,(CQMF+D#06):%,(QMFBS+D#35):%
                11100000 01010010 10010000 00100011 11100010 11000100
 438   00091              FILTERN  ,(CQMF+D#08):%,(QMFBS+D#45):%
                11100000 01010010 10110000 00101101 11100010 11000100
 439   00092              FILTERN  ,(CQMF+D#10):%,(QMFBS+D#55):%
                11100000 01010010 11010000 00110111 11100010 11000100
 440   00093              FILTERN  ,(CQMF+D#12):%,(QMFBS+D#65):%
                11100000 01010010 11110000 01000001 11100010 11000100
 441   00094              FILTERN  ,(CQMF+D#14):%,(QMFBS+D#75):%
                11100000 01010011 00010000 01001011 11100010 11000100
 442   00095              FILTERN  ,(CQMF+D#16):%,(QMFBS+D#85):%
                11100000 01010011 00110000 01010101 11100010 11000100
 443   00096              FILTERN  ,(CQMF+D#18):%,(QMFBS+D#95):%
                11100000 01010011 01010000 01011111 11100010 11000100
 444   00097              FILTERN  ,(CQMF+D#20):%,(QMFBS+D#105):%
                11100000 01010011 01110000 01101001 11100010 11000100
 445   00098              FILTERN  ,(CQMF+D#22):%,(QMFBS+D#115):%
                11100000 01010011 10010000 01110011 11100010 11000100
 446   00099              FILTERN  ,(CQMF+D#24):%,(QMFBS+D#125):%
                11100000 01010011 10110000 01111101 11100010 11000100
 447   0009A              FILTERN  ,(CQMF+D#26):%,(QMFBS+D#135):%
                11100000 01010011 11010000 10000111 11100010 11000100
 448   0009B              FILTERN  ,(CQMF+D#28):%,(QMFBS+D#145):%
                11100000 01010011 11110000 10010001 11100010 11000100
 449   0009C              FILTERN  ,(CQMF+D#30):%,(QMFBS+D#155):%
                11100000 01010100 00010000 10011011 11100010 11000100
 450   0009D              FILTERN  ,(CQMF+D#32):%,(QMFBS+D#165):%
                11100000 01010100 00110000 10100101 11100010 11000100
 451   0009E              FILTERN  ,(CQMF+D#34):%,(QMFBS+D#175):%
                11100000 01010100 01010000 10101111 11100010 11000100
 452   0009F              FILTERL
                11100000 01101111 11110110 11111111 11100000 11000000
 453   000A0              JMP      ,PERT:%          ; Perturb sample location
                00110000 00001100 10100110 11111111 11000000 00000000
 454   000A1                                        ; and return
```

```
456    000A1   ;*************************************************************
457    000A1   ;*   PROCEDURE Quadrature_Mirror_Filter_Low                  *
458    000A1   ;* { This procedure does the quadrature mirror filtering*
459    000A1   ;* operation assuming FEL is in the FE input buffer.    }*
460    000A1   ;*                                                       *
461    000A1   ;* sum := FEH[n+0] * CQMF[1]                             *
462    000A1   ;* FOR i := 1 TO 17 DO                                   *
463    000A1   ;*   sum := sum + FEH[n+2*i] * CQMF[2*i+1];              *
464    000A1   ;* FOR i := 0 TO 17 DO                                   *
465    000A1   ;*   sum := sum + FEH[n+2*i] * CQMF[2*i+1];              *
466    000A1   ;* LPFBS := sum;                                         *
467    000A1   ;*                                                       *
468    000A1   ;*************************************************************
469    000A1   QMFL:     LOAD     FE,FE
               11100000 01000001 00110100 00010011 11000001 10000100
470    000A2             STMR      ,QMFBS                    ; Load FEL and store in Ring b
               11100000 00111111 11110000 00000000 11000000 00101010
471    000A3             STORE     ,LSTFEL,LSTFEL            ; Store for the test point
               11100000 01000001 01010100 00010101 11000000 00100011
472    000A4             FILTER1   ,(CQMF+D#1):%,(QMFBS+D#5):%  ; Do the FEH terms first
               11100000 01010010 01000000 00000101 11100000 01000100
473    000A5             FILTERN   ,(CQMF+D#3):%,(QMFBS+D#15):%
               11100000 01010010 01100000 00001111 11100010 11000100
474    000A6             FILTERN   ,(CQMF+D#5):%,(QMFBS+D#25):%
               11100000 01010010 10000000 00011001 11100010 11000100
475    000A7             FILTERN   ,(CQMF+D#7):%,(QMFBS+D#35):%
               11100000 01010010 10100000 00100011 11100010 11000100
476    000A8             FILTERN   ,(CQMF+D#9):%,(QMFBS+D#45):%
               11100000 01010010 11000000 00101101 11100010 11000100
477    000A9             FILTERN   ,(CQMF+D#11):%,(QMFBS+D#55):%
               11100000 01010010 11100000 00110111 11100010 11000100
478    000AA             FILTERN   ,(CQMF+D#13):%,(QMFBS+D#65):%
               11100000 01010011 00000000 01000001 11100010 11000100
479    000AB             FILTERN   ,(CQMF+D#15):%,(QMFBS+D#75):%
               11100000 01010011 00100000 01001011 11100010 11000100
480    000AC             FILTERN   ,(CQMF+D#17):%,(QMFBS+D#85):%
               11100000 01010011 01000000 01010101 11100010 11000100
481    000AD             FILTERN   ,(CQMF+D#19):%,(QMFBS+D#95):%
               11100000 01010011 01100000 01011111 11100010 11000100
482    000AE             FILTERN   ,(CQMF+D#21):%,(QMFBS+D#105):%
               11100000 01010011 10000000 01101001 11100010 11000100
483    000AF             FILTERN   ,(CQMF+D#23):%,(QMFBS+D#115):%
               11100000 01010011 10100000 01110011 11100010 11000100
484    000B0             FILTERN   ,(CQMF+D#25):%,(QMFBS+D#125):%
               11100000 01010011 11000000 01111101 11100010 11000100
485    000B1             FILTERN   ,(CQMF+D#27):%,(QMFBS+D#135):%
               11100000 01010011 11100000 10000111 11100010 11000100
486    000B2             FILTERN   ,(CQMF+D#29):%,(QMFBS+D#145):%
               11100000 01010100 00000000 10010001 11100010 11000100
487    000B3             FILTERN   ,(CQMF+D#31):%,(QMFBS+D#155):%
               11100000 01010100 00100000 10011011 11100010 11000100
488    000B4             FILTERN   ,(CQMF+D#33):%,(QMFBS+D#165):%
               11100000 01010100 01000000 10100101 11100010 11000100
```

```
489  000B5          FILTERN  ,(CQMF+D#35):%,(QMFBS+D#175):%
           11100000 01010100 01100000 10101111 11100010 11000100
490  000B6          FILTERN  ,(CQMF+D#1):%,(QMFBS+D#10):% ; Now do FEL terms
           11100000 01010010 01000000 00001010 11100010 11000100
491  000B7          FILTERN  ,(CQMF+D#3):%,(QMFBS+D#20):%
           11100000 01010010 01100000 00010100 11100010 11000100
492  000B8          FILTERN  ,(CQMF+D#5):%,(QMFBS+D#30):%
           11100000 01010010 10000000 00011110 11100010 11000100
493  000B9          FILTERN  ,(CQMF+D#7):%,(QMFBS+D#40):%
           11100000 01010010 10100000 00101000 11100010 11000100
494  000BA          FILTERN  ,(CQMF+D#9):%,(QMFBS+D#50):%
           11100000 01010010 11000000 00110010 11100010 11000100
495  000BB          FILTERN  ,(CQMF+D#11):%,(QMFBS+D#60):%
           11100000 01010010 11100000 00111100 11100010 11000100
496  000BC          FILTERN  ,(CQMF+D#13):%,(QMFBS+D#70):%
           11100000 01010011 00000000 01000110 11100010 11000100
497  000BD          FILTERN  ,(CQMF+D#15):%,(QMFBS+D#80):%
           11100000 01010011 00100000 01010000 11100010 11000100
498  000BE          FILTERN  ,(CQMF+D#17):%,(QMFBS+D#90):%
           11100000 01010011 01000000 01011010 11100010 11000100
499  000BF          FILTERN  ,(CQMF+D#19):%,(QMFBS+D#100):%
           11100000 01010011 01100000 01100100 11100010 11000100
500  000C0          FILTERN  ,(CQMF+D#21):%,(QMFBS+D#110):%
           11100000 01010011 10000000 01101110 11100010 11000100
501  000C1          FILTERN  ,(CQMF+D#23):%,(QMFBS+D#120):%
           11100000 01010011 10100000 01111000 11100010 11000100
502  000C2          FILTERN  ,(CQMF+D#25):%,(QMFBS+D#130):%
           11100000 01010011 11000000 10000010 11100010 11000100
503  000C3          FILTERN  ,(CQMF+D#27):%,(QMFBS+D#140):%
           11100000 01010011 11100000 10001100 11100010 11000100
504  000C4          FILTERN  ,(CQMF+D#29):%,(QMFBS+D#150):%
           11100000 01010100 00000000 10010110 11100010 11000100
505  000C5          FILTERN  ,(CQMF+D#31):%,(QMFBS+D#160):%
           11100000 01010100 00100000 10100000 11100010 11000100
506  000C6          FILTERN  ,(CQMF+D#33):%,(QMFBS+D#170):%
           11100000 01010100 01000000 10101010 11100010 11000100
507  000C7          FILTERN  ,(CQMF+D#35):%,(QMFBS+D#180):%
           11100000 01010100 01100000 10110100 11100010 11000100
508  000C8          FILTERL
           11100000 01101111 11110110 11111111 11100000 11000000
509  000C9          JMF       ,PERT:%       ; Perturb sample location
           00110000 00001100 10100110 11111111 11000000 00000000
510  000CA                                  ; and return
511  000CA                                 ; return
```

```
513   000CA  ;*******************************************************
514   000CA  ;*    PROCEDURE PERTurbate;                             *
515   000CA  ;*                                                      *
516   000CA  ;*{  This procedure perterbates the location of the FE  }*
517   000CA  ;*  BEGIN                                               *
518   000CA  ;*  Multiply accumulator by CON4;                       *
519   000CA  ;*  Generatate seed uniformly distributed in [-thresh,thresh]*
520   000CA  ;*  IF magnitude(FE) > seed THEN store at PERTBS        *
521   000CA  ;*      ELSE IF MSB of least is one THEN store at PERTBS-1  *
522   000CA  ;*        ELSE store at PERTBS+1                        *
523   000CA  ;*  END;                                                *
524   000CA  ;*                                                      *
525   000CA  ;* { The Fe sample is perturbed with the following probability*
526   000CA  ;* density function.                                    *
527   000CA  ;*                      |                               *
528   000CA  ;* Probablilty of being  |                              *
529   000CA  ;* Perturbed             |                              *
530   000CA  ;*                      +0.5                            *
531   000CA  ;*                    +  |  +                           *
532   000CA  ;*                 +     |    +                         *
533   000CA  ;*              +        |      +                       *
534   000CA  ;*           +           |         +                    *
535   000CA  ;*        +              |            +                 *
536   000CA  ;*      +                |               +              *
537   000CA  ;* ----+------------------------------------+----------- *
538   000CA  ;*  -THRESH               0               THRESH        *
539   000CA  ;*                Input sample value                    *
540   000CA  ;*  END;                                                *
541   000CA  ;*******************************************************
542   000CA  PERT:
543   000CA  ; Now do a full precision multiply by CON6
544   000CA          STLR      ,LEAST         ; Store least sig prod
             11100000 00111111 11110011 11111111 11000000 00111001
545   000CB          STMR      ,MOST          ; store most sig product
             11100000 00111111 11110011 11111110 11000000 00101010
546   000CC          FILTERI   ,CON6,MOST     ; Multiply most sig by CON6
             11100000 01010110 11100011 11111110 11100000 01000100
547   000CD          FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
548   000CE          STLR      ,MOST
             11100000 00111111 11110011 11111110 11000000 00111001
549   000CF          LDMR      MOST           ; Shift left 16 places
             11100000 01010000 00000011 11111110 11000001 10000100
550   000D0          FILTRAU   ,CON6,LEAST    ; Add CON6 times least sig product
             11100000 01010110 11100011 11111111 11000010 01000100
551   000D1          FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
552   000D2          STLR      ,LEAST
             11100000 00111111 11110011 11111111 11000000 00111001
553   000D3          STMS      LSTFE,LSTFE    ; Save for the test point
             11100000 01000001 01110100 00010111 11000000 00101010
554   000D4          STMR      ,MOST          ; Store at base of LPF
             11100000 00111111 11110011 11111110 11000000 00101010
```

```
 555   000D5  ;        STMR      ,DLYBS
 556   000D5  ; Take the absolute value
 557   000D5           JMP    NEG,NEGATE:%         ; Test last bus value was negative
              00110010 00001101 01110110 11111111 11000000 00000000
 558   000D6           JMP    ,POSTV:%
              00110000 00001101 10100110 11111111 11000000 00000000
 559   000D7  NEGATE:  FILTER1 ,CONM1,MOST:%       ; IF negative then negate it
              11100000 01010000 00100011 11111110 11100000 01000100
 560   000D8           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 561   000D9           STLR    ,MOST
              11100000 00111111 11110011 11111110 11000000 00111001
 562   000DA  POSTV:
 563   000DA           FILTER1 ,THRESH,LEAST       ; Scale [-1,1) remainder
              11100000 01010111 01100011 11111111 11100000 01000100
 564   000DB                                       ; to [-THRESH,THRESH)
 565   000DB           FILTERL                     ; To set random seed
              11100000 01101111 11110110 11111111 11100000 11000000
 566   000DC           STMS    TEMP,TEMP           ; leave result in LSW
              11100000 01000001 10000100 00011000 11000000 00101010
 567   000DD           SPL     ,,(D#256*COEFB+CON1):%,(D#256*SCRAT+TEMP):%,.MULT1
              11100000 01010000 00010100 00011000 11100000 01000100
 568   000DE                                       ; load seed into accumulator
 569   000DE           FILTERN ,CONM1,MOST         ; Subtract absolute value of signal
              11100000 01010000 00100011 11111110 11100010 11000100
 570   000DF           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 571   000E0           STLR    ,MOST               ; Put on the bus
              11100000 00111111 11110011 11111110 11000000 00111001
 572   000E1           JMP    NEG,NOPERT:%         ; If magnitude of signal > 128 don'T
              00110010 00001110 10110110 11111111 11000000 00000000
 573   000E2           FILTER1 ,(CLPF+D#15):%,LEAST ;scramble bits in LEAST
              11100000 01010001 00110011 11111111 11100000 01000100
 574   000E3           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
 575   000E4           STLR    ,LEAST              ; Save least sig word of scrambled 1
              11100000 00111111 11110011 11111111 11000000 00111001
 576   000E5           LDMR    LEAST               ; Put MSB of LEAST into sign bit
              11100000 01010000 00000011 11111111 11000001 10000100
 577   000E6           JMP    NEG,PERTP:%
              00110010 00001110 10010110 11111111 11000000 00000000
 578   000E7           LOAD   LSTFE,LSTFE
              11100000 01000001 01110100 00010111 11000001 10000100
 579   000E8           STMR   CRTN,(LPFBS-D#1):%
              10100000 00111111 11110000 11000000 11000000 00101010
 580   000E9  PERTP:   LOAD   LSTFE,LSTFE
              11100000 01000001 01110100 00010111 11000001 10000100
 581   000EA           STMR   CRTN,(LPFBS+D#1):%
              10100000 00111111 11110000 11000010 11000000 00101010
 582   000EB  NOPERT:  LOAD   LSTFE,LSTFE
              11100000 01000001 01110100 00010111 11000001 10000100
 583   000EC           STMR   CRTN,(LPFBS+0):%
              10100000 00111111 11110000 11000001 11000000 00101010
 585   000ED  ;***************************************************************
 586   000ED  ;*    PROCEDURE Low_Pass_Filter;                                *
 587   000ED  ;*{ This Procedure low pass filters the data in the low pass*
 588   000ED  ;* Filter buffer.                                            }*
 589   000ED  ;*  sum := CLPF[0] * LPFBS[0];                                 *
 590   000ED  ;*  FOR i := 1 TO 30 DO                                         *
 591   000ED  ;*    sum := sum + CLPF[i] * LPFBS[i];                          *
 592   000ED  ;*  LPFBS[31] := sum;                                           *
 593   000ED  ;*  DLYBS[0] := DDFBS[0];                                       *
 594   000ED  ;*  END;                                                        *
 595   000ED  ;*                                                              *
 596   000ED  ;***************************************************************
 597   000ED  LPF:        FILTER1  ,(CLPF+D#0):%,(LPFBS+D#0):%
              11100000 01010000 01000000 11000001 11100000 01000100
 598   000EE              FILTERN  ,(CLPF+D#1):%,(LPFBS+D#1):%
              11100000 01010000 01010000 11000010 11100010 11000100
 599   000EF              FILTERN  ,(CLPF+D#2):%,(LPFBS+D#2):%
              11100000 01010000 01100000 11000011 11100010 11000100
 600   000F0              FILTERN  ,(CLPF+D#3):%,(LPFBS+D#3):%
              11100000 01010000 01110000 11000100 11100010 11000100
 601   000F1              FILTERN  ,(CLPF+D#4):%,(LPFBS+D#4):%
              11100000 01010000 10000000 11000101 11100010 11000100
 602   000F2              FILTERN  ,(CLPF+D#5):%,(LPFBS+D#5):%
              11100000 01010000 10010000 11000110 11100010 11000100
 603   000F3              FILTERN  ,(CLPF+D#6):%,(LPFBS+D#6):%
              11100000 01010000 10100000 11000111 11100010 11000100
 604   000F4              FILTERN  ,(CLPF+D#7):%,(LPFBS+D#7):%
              11100000 01010000 10110000 11001000 11100010 11000100
 605   000F5              FILTERN  ,(CLPF+D#8):%,(LPFBS+D#8):%
              11100000 01010000 11000000 11001001 11100010 11000100
 606   000F6              FILTERN  ,(CLPF+D#9):%,(LPFBS+D#9):%
              11100000 01010000 11010000 11001010 11100010 11000100
 607   000F7              FILTERN  ,(CLPF+D#10):%,(LPFBS+D#10):%
              11100000 01010000 11100000 11001011 11100010 11000100
 608   000F8              FILTERN  ,(CLPF+D#11):%,(LPFBS+D#11):%
              11100000 01010000 11110000 11001100 11100010 11000100
 609   000F9              FILTERN  ,(CLPF+D#12):%,(LPFBS+D#12):%
              11100000 01010001 00000000 11001101 11100010 11000100
 610   000FA              FILTERN  ,(CLPF+D#13):%,(LPFBS+D#13):%
              11100000 01010001 00010000 11001110 11100010 11000100
 611   000FB              FILTERN  ,(CLPF+D#14):%,(LPFBS+D#14):%
              11100000 01010001 00100000 11001111 11100010 11000100
 612   000FC              FILTERN  ,(CLPF+D#15):%,(LPFBS+D#15):%
              11100000 01010001 00110000 11010000 11100010 11000100
 613   000FD              FILTERN  ,(CLPF+D#16):%,(LPFBS+D#16):%
              11100000 01010001 01000000 11010001 11100010 11000100
 614   000FE              FILTERN  ,(CLPF+D#17):%,(LPFBS+D#17):%
              11100000 01010001 01010000 11010010 11100010 11000100
 615   000FF              FILTERN  ,(CLPF+D#18):%,(LPFBS+D#18):%
              11100000 01010001 01100000 11010011 11100010 11000100
 616   00100              FILTERN  ,(CLPF+D#19):%,(LPFBS+D#19):%
              11100000 01010001 01110000 11010100 11100010 11000100
 617   00101              FILTERN  ,(CLPF+D#20):%,(LPFBS+D#20):%
              11100000 01010001 10000000 11010101 11100010 11000100
```

```
618   00102            FILTERN  ,(CLPF+D#21):%,(LPFBS+D#21):%
             11100000 01010001 10010000 11010110 11100010 11000100
619   00103            FILTERN  ,(CLPF+D#22):%,(LPFBS+D#22):%
             11100000 01010001 10100000 11010111 11100010 11000100
620   00104            FILTERN  ,(CLPF+D#23):%,(LPFBS+D#23):%
             11100000 01010001 10110000 11011000 11100010 11000100
621   00105            FILTERN  ,(CLPF+D#24):%,(LPFBS+D#24):%
             11100000 01010001 11000000 11011001 11100010 11000100
622   00106            FILTERN  ,(CLPF+D#25):%,(LPFBS+D#25):%
             11100000 01010001 11010000 11011010 11100010 11000100
623   00107            FILTERN  ,(CLPF+D#26):%,(LPFBS+D#26):%
             11100000 01010001 11100000 11011011 11100010 11000100
624   00108            FILTERN  ,(CLPF+D#27):%,(LPFBS+D#27):%
             11100000 01010001 11110000 11011100 11100010 11000100
625   00109            FILTERN  ,(CLPF+D#28):%,(LPFBS+D#28):%
             11100000 01010010 00000000 11011101 11100010 11000100
626   0010A            FILTERN  ,(CLPF+D#29):%,(LPFBS+D#29):%
             11100000 01010010 00010000 11011110 11100010 11000100
627   0010B            FILTERN  ,(CLPF+D#30):%,(LPFBS+D#30):%
             11100000 01010010 00100000 11011111 11100010 11000100
628   0010C            FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
629   0010D            STLR     ,LEAST         ; Store least sig prod
             11100000 00111111 11110011 11111111 11000000 00111001
630   0010E            STMR     ,MOST          ; store most sig product
             11100000 00111111 11110011 11111110 11000000 00101010
631   0010F            FILTERI  ,GNLPF,MOST    ; Multiply most sig by GNLPF
             11100000 01010111 00100011 11111110 11100000 01000100
632   00110            FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
633   00111            STLR     ,MOST
             11100000 00111111 11110011 11111110 11000000 00111001
634   00112            LDMR     MOST           ; Shift left 16 places
             11100000 01010000 00000011 11111110 11000001 10000100
635   00113            FILTRAU  ,GNLPF,LEAST   ; Add GNLPF times least sig product
             11100000 01010111 00100011 11111111 11000010 01000100
636   00114            FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
637   00115            STMR     CRTN,(LPFBS+LPFSZ):%   ; Store it away
             10100000 00111111 11110000 11100000 11000000 00101010
```

```
 639   00116  ;***********************************************************
 640   00116  ;*    PROCEDURE High_Pass_Filter;                            *
 641   00116  ;*{  This Procedure high pass filters the data in the high   *
 642   00116  ;*  pass Filter buffer.                                     }*
 643   00116  ;*  sum := CLPF[0] * HPFBS[0];                               *
 644   00116  ;*  FOR i := 1 TO 30 DO                                      *
 645   00116  ;*     sum := sum + CLPF[i] * HPFBS[i];                      *
 646   00116  ;*  HPFBS[31] := 5*HPFBS[15]-sum;                            *
 647   00116  ;*  END;                                                     *
 648   00116  ;*                                                           *
 649   00116  ;***********************************************************
 650   00116  HPF:       FILTER1  ,(CLPF+D#0):%,(HPFBS+D#0):%
              11100000 01010000 01000001 00110110 11100000 01000100
 651   00117             FILTERN  ,(CLPF+D#1):%,(HPFBS+D#1):%
              11100000 01010000 01010001 00110111 11100010 11000100
 652   00118             FILTERN  ,(CLPF+D#2):%,(HPFBS+D#2):%
              11100000 01010000 01100001 00111000 11100010 11000100
 653   00119             FILTERN  ,(CLPF+D#3):%,(HPFBS+D#3):%
              11100000 01010000 01110001 00111001 11100010 11000100
 654   0011A             FILTERN  ,(CLPF+D#4):%,(HPFBS+D#4):%
              11100000 01010000 10000001 00111010 11100010 11000100
 655   0011B             FILTERN  ,(CLPF+D#5):%,(HPFBS+D#5):%
              11100000 01010000 10010001 00111011 11100010 11000100
 656   0011C             FILTERN  ,(CLPF+D#6):%,(HPFBS+D#6):%
              11100000 01010000 10100001 00111100 11100010 11000100
 657   0011D             FILTERN  ,(CLPF+D#7):%,(HPFBS+D#7):%
              11100000 01010000 10110001 00111101 11100010 11000100
 658   0011E             FILTERN  ,(CLPF+D#8):%,(HPFBS+D#8):%
              11100000 01010000 11000001 00111110 11100010 11000100
 659   0011F             FILTERN  ,(CLPF+D#9):%,(HPFBS+D#9):%
              11100000 01010000 11010001 00111111 11100010 11000100
 660   00120             FILTERN  ,(CLPF+D#10):%,(HPFBS+D#10):%
              11100000 01010000 11100001 01000000 11100010 11000100
 661   00121             FILTERN  ,(CLPF+D#11):%,(HPFBS+D#11):%
              11100000 01010000 11110001 01000001 11100010 11000100
 662   00122             FILTERN  ,(CLPF+D#12):%,(HPFBS+D#12):%
              11100000 01010001 00000001 01000010 11100010 11000100
 663   00123             FILTERN  ,(CLPF+D#13):%,(HPFBS+D#13):%
              11100000 01010001 00010001 01000011 11100010 11000100
 664   00124             FILTERN  ,(CLPF+D#14):%,(HPFBS+D#14):%
              11100000 01010001 00100001 01000100 11100010 11000100
 665   00125             FILTERN  ,(CLPF+D#15):%,(HPFBS+D#15):%
              11100000 01010001 00110001 01000101 11100010 11000100
 666   00126             FILTERN  ,(CLPF+D#16):%,(HPFBS+D#16):%
              11100000 01010001 01000001 01000110 11100010 11000100
 667   00127             FILTERN  ,(CLPF+D#17):%,(HPFBS+D#17):%
              11100000 01010001 01010001 01000111 11100010 11000100
 668   00128             FILTERN  ,(CLPF+D#18):%,(HPFBS+D#18):%
              11100000 01010001 01100001 01001000 11100010 11000100
 669   00129             FILTERN  ,(CLPF+D#19):%,(HPFBS+D#19):%
              11100000 01010001 01110001 01001001 11100010 11000100
 670   0012A             FILTERN  ,(CLPF+D#20):%,(HPFBS+D#20):%
              11100000 01010001 10000001 01001010 11100010 11000100
```

```
671    0012B           FILTERN  ,(CLPF+D#21):%,(HPFBS+D#21):%
              11100000 01010001 10010001 01001011 11100010 11000100
672    0012C           FILTERN  ,(CLPF+D#22):%,(HPFBS+D#22):%
              11100000 01010001 10100001 01001100 11100010 11000100
673    0012D           FILTERN  ,(CLPF+D#23):%,(HPFBS+D#23):%
              11100000 01010001 10110001 01001101 11100010 11000100
674    0012E           FILTERN  ,(CLPF+D#24):%,(HPFBS+D#24):%
              11100000 01010001 11000001 01001110 11100010 11000100
675    0012F           FILTERN  ,(CLPF+D#25):%,(HPFBS+D#25):%
              11100000 01010001 11010001 01001111 11100010 11000100
676    00130           FILTERN  ,(CLPF+D#26):%,(HPFBS+D#26):%
              11100000 01010001 11100001 01010000 11100010 11000100
677    00131           FILTERN  ,(CLPF+D#27):%,(HPFBS+D#27):%
              11100000 01010001 11110001 01010001 11100010 11000100
678    00132           FILTERN  ,(CLPF+D#28):%,(HPFBS+D#28):%
              11100000 01010010 00000001 01010010 11100010 11000100
679    00133           FILTERN  ,(CLPF+D#29):%,(HPFBS+D#29):%
              11100000 01010010 00010001 01010011 11100010 11000100
680    00134           FILTERN  ,(CLPF+D#30):%,(HPFBS+D#30):%
              11100000 01010010 00100001 01010100 11100010 11000100
681    00135           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
682    00136           STLR     ,LEAST         ; Store least sig prod
              11100000 00111111 11110011 11111111 11000000 00111001
683    00137           STMR     ,MOST          ; store most sig product
              11100000 00111111 11110011 11111110 11000000 00101010
684    00138           FILTER1  ,GNLPF,MOST    ; Multiply most sig by GNLPF
              11100000 01010111 00100011 11111110 11100000 01000100
685    00139           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
686    0013A           STLR     ,MOST
              11100000 00111111 11110011 11111110 11000000 00111001
687    0013B           LDMP      MOST          ; Shift left 16 places
              11100000 01010000 00000011 11111110 11000001 10000100
688    0013C           FILTRAD  ,GNLPF,LEAST  ; Add GNLPF times least sig product
              11100000 01010111 00100011 11111111 11000010 01000100
689    0013D           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
690    0013E           STMR     ,(HPFBS+LPFSZ):%  ; Store it away
              11100000 00111111 11110001 01010101 11000000 00101010
691    0013F           FILTER1  ,CON1,(HPFBS+LPFSZ):% ; subtract 5*middle element
              11100000 01010000 00010001 01010101 11100000 01000100
692    00140                                    ; to get complement HPF from LP
693    00140           FILTRNM  ,CON4,(HPFBS+D#15):%
              11100000 01010110 11010001 01000101 11100110 11000100
694    00141           FILTERN  ,CON1,(HPFBS+D#15):%
              11100000 01010000 00010001 01000101 11100010 11000100
695    00142           FILTERL
              11100000 01101111 11110110 11111111 11100000 11000000
696    00143           STLR      CRTN,(HPFBS+LPFSZ):%  ; Store that away
              10100000 00111111 11110001 01010101 11000000 00111001
```

```
698   00144  ;***********************************************************
699   00144  ;*          PROCEDURE PREDict;                              *
700   00144  ;*    { This procedure predictor filters data in the       *
701   00144  ;* filter buffer of the ring buffer.                      }*
702   00144  ;*                                                         *
703   00144  ;***********************************************************
704   00144  PRED:     FILTER1 SCRAT,(P0+D#0):%,(PREDBS+D#1):%
             11100000 01000000 00000001 00100110 11100000 01000100
705   00145            FILTERN SCRAT,(P0+D#1):%,(PREDBS+D#2):%
             11100000 01000000 00010001 00100111 11100010 11000100
706   00146            FILTERN SCRAT,(P0+D#2):%,(PREDBS+D#3):%
             11100000 01000000 00100001 00101000 11100010 11000100
707   00147            FILTERN SCRAT,(P0+D#3):%,(PREDBS+D#4):%
             11100000 01000000 00110001 00101001 11100010 11000100
708   00148            FILTERN SCRAT,(P0+D#4):%,(PREDBS+D#5):%
             11100000 01000000 01000001 00101010 11100010 11000100
709   00149            FILTERN SCRAT,(P0+D#5):%,(PREDBS+D#6):%
             11100000 01000000 01010001 00101011 11100010 11000100
710   0014A            FILTERN SCRAT,(P0+D#6):%,(PREDBS+D#7):%
             11100000 01000000 01100001 00101100 11100010 11000100
711   0014B            FILTERN SCRAT,(P0+D#7):%,(PREDBS+D#8):%
             11100000 01000000 01110001 00101101 11100010 11000100
712   0014C            FILTRNM COEFB,PREDSCL,PREDBS
             11100000 01010110 10110001 00100101 11100110 11000100
713   0014D            FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
714   0014E  ; Do a double precion multiply by 8
715   0014E            STLR      ,LEAST         ; Store least sig prod
             11100000 00111111 11110011 11111111 11000000 00111001
716   0014F            STMR      ,MOST          ; store most sig product
             11100000 00111111 11110011 11111110 11000000 00101010
717   00150            FILTER1   ,CON8,MOST     ; Multiply most sig by 8
             11100000 01010110 11100011 11111110 11100000 01000100
718   00151            FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
719   00152            STLR      ,MOST
             11100000 00111111 11110011 11111110 11000000 00111001
720   00153            LDMR      MOST           ; Shift left 16 places
             11100000 01010000 00000011 11111110 11000001 10000100
721   00154            FILTRAU   ,CON8,LEAST    ; Add 8 times least sig product
             11100000 01010110 11100011 11111111 11000010 01000100
722   00155            FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
723   00156            STMR    ,PREDBS
             11100000 00111111 11110001 00100101 11000000 00101010
724   00157            FILTER1   ,GNPRED,PREDBS ; Scale so no overflow in deemphasis
             11100000 01010111 01010001 00100101 11100000 01000100
725   00158            FILTERL
             11100000 01101111 11110110 11111111 11100000 11000000
726   00159            STMR    CRTN,DEMBS
             10100000 00111111 11110001 00101110 11000000 00101010
```

```
 728   0015A  ;************************************************************
 729   0015A  ;*      PROCEDURE CoPY_PREDictor_coefficients                  *
 730   0015A  ;*{ This procedure copies the predictor coefficients from    *
 731   0015A  ;* the temporary predictor buffer to the active predictor    *
 732   0015A  ;* buffer.                                                  }*
 733   0015A  ;*                                                           *
 734   0015A  ;************************************************************
 735   0015A  CPYPRED: LOAD (PTMP0)!%,(PTMP0)!%
              11100000 01000000 10010100 00001001 11000001 10000100
 736   0015B           STORE ,(P0)!%,(P0)!%
              11100000 01000000 00000100 00000000 11000000 00100011
 737   0015C           LOAD (PTMP0+D#1)!%,(PTMP0+D#1)!%
              11100000 01000000 10100100 00001010 11000001 10000100
 738   0015D           STORE ,B(P0+D#1)!%,(P0+D#1)!%
              11100000 01000000 00010100 00000001 11000000 00100011
 739   0015E           LOAD B(PTMP0+D#2)!%,(PTMP0+D#2)!%
              11100000 01000000 10110100 00001011 11000001 10000100
 740   0015F           STORE ,B(P0+D#2)!%,(P0+D#2)!%
              11100000 01000000 00100100 00000010 11000000 00100011
 741   00160           LOAD B(PTMP0+D#3)!%,(P0+D#3)!%
              11100000 01000000 11000100 00000011 11000001 10000100
 742   00161           STORE ,B(P0+D#3)!%,(P0+D#3)!%
              11100000 01000000 00110100 00000011 11000000 00100011
 743   00162           LOAD B(PTMP0+D#4)!%,(PTMP0+D#4)!%
              11100000 01000000 11010100 00001101 11000001 10000100
 744   00163           STORE ,B(P0+D#4)!%,(P0+D#4)!%
              11100000 01000000 01000100 00000100 11000000 00100011
 745   00164           LOAD B(PTMP0+D#5)!%,(PTMP0+D#5)!%
              11100000 01000000 11100100 00001110 11000001 10000100
 746   00165           STORE ,B(P0+D#5)!%,(P0+D#5)!%
              11100000 01000000 01010100 00000101 11000000 00100011
 747   00166           LOAD B(PTMP0+D#6)!%,(PTMP0+D#6)!%
              11100000 01000000 11110100 00001111 11000001 10000100
 748   00167           STORE ,B(P0+D#6)!%,(P0+D#6)!%
              11100000 01000000 01100100 00000110 11000000 00100011
 749   00168           LOAD B(PTMP0+D#7)!%,(PTMP0+D#7)!%
              11100000 01000001 00000100 00010000 11000001 10000100
 750   00169           STORE CRTN,B(P0+D#7)!%,(P0+D#7)!%
              10100000 01000000 01110100 00000111 11000000 00100011
```

```
0016A ;*******************************************************
0016A ;*     PROCEDURE DEEMPhasize;                          *
0016A ;*  { This procedure deemphasizes data in the deemphasis *
0016A ;* filter memory.                                      }*
0016A ;*                                                     *
0016A ;*******************************************************
0016A DEEM:    LDMR    DEMBS
      11100000 01010000 00000001 00101110 11000001 10000100
0016B          FILTERA ,A,(DEMBS+D#1):%
      11100000 01010000 00110001 00101111 11100010 01000100
0016C          FILTERN ,A,(DEMBS+D#1):%
      11100000 01010000 00110001 00101111 11100010 11000100
0016D          FILTERL
      11100000 01101111 11110110 11111111 11100000 11000000
0016E          STMR    ,DEMBS
      11100000 00111111 11110001 00101110 11000000 00101010
0016F          FILTER1 ,GNOUT,DEMBS  ; Scale output to prevent overflow
      11100000 01010111 01100001 00101110 11100000 01000100
00170          FILTERL
      11100000 01101111 11110110 11111111 11100000 11000000
00171          STMR    CRTN,(DEMBS+D#1):%
      10100000 00111111 11110001 00101111 11000000 00101010
00172          END


 TOTAL ASSEMBLY ERRORS =    0
```

```
                    CROSS REFERENCE TABLE

LABEL     TYPE   VALUE        REFERENCES
A          A     00003      -107   108   759   760
ABSZ       A     00001       -66
ACC        A     00200         0
AUTOI      D                   0
AUTOL      D                   0
AUTON      D                   0
CALL       D                 191   194   197   200   201   204   209   214
                             219   224   225   230   237   244   251   256
                             257   262   268   274   278   282   285   286
                             289   296   299   302   305   306   309   312
                             315   318   321   322   325   388   390
CBDF       A     00047      -110   111
CHPF       A     0004A      -111   112
CJP        A     00003         0
CJPP       A     0000B         0
CJS        A     00001         0
CLKP       A     00080         0
CLKXY      A     00040         0
CLPF       A     00004      -108   109   573   597   598   599   600   601
                             602   603   604   605   606   607   608   609
                             610   611   612   613   614   615   616   617
                             618   619   620   621   622   623   624   625
                             626   627   650   651   652   653   654   655
                             656   657   658   659   660   661   662   663
                             664   665   666   667   668   669   670   671
                             672   673   674   675   676   677   678   679
                             680
COEFB      A     00005       168   567   712
CON0       A     00000      -104   168   275   393
CON1       A     00001      -105   567   691   694
CON128     A     0006F      -117   118
CON1QRTR   A     0006B      -113
CON2       A     0006C      -114   115
CON4       A     0006D      -115   116   693
CON8       A     0006E      -116   117   546   550   717   721
CONM1      A     00002      -106   559   569
CONT       A     0000E         0
CPYPRED    A     0015A       278  -735
CQMF       A     00023      -109   110   416   417   418   419   420   421
                             422   423   424   425   426   427   428   429
                             430   431   432   433   434   435   436   437
                             438   439   440   441   442   443   444   445
                             446   447   448   449   450   451   472   473
                             474   475   476   477   478   479   480   481
                             482   483   484   485   486   487   488   489
                             490   491   492   493   494   495   496   497
                             498   499   500   501   502   503   504   505
                             506   507
CROSSEN    A     00008         0
```

```
CRTN      A    0000A     396  579  581  583  637  696  726  750
                         765
DDFSZ     A    00003     -65  111
DEEM      A    0016A     390 -758
DEMBS     A    0012E     -79   80   83  379  726  758  759  760
                         762  763  765
DEMSZ     A    00002     -59
DIR       A    00010       0
DLY       A    00064     -60   77
DLYBS     A    000BE     -74   75
EVERY     A    00056     191  194  197  200  204  209  214  219
                         224  230  237  244  251  256  262  268
                         274  282  285  289  296  299  302  305
                         309  312  315  318  321  325 -351  351
FALSE     A    00001       0
FE        A    00013     -94   95  360  360  413  413  469  469
FEPRT     A    00003    -133  355
FILTER1   D              168  275  393  416  472  546  559  563
                         573  597  631  650  684  691  704  717
                         724  763
FILTERA   D              759
FILTERL   D              169  276  359  394  452  508  547  551
                         560  565  570  574  628  632  636  681
                         685  689  695  713  718  722  725  761
                         764
FILTERN   D              417  418  419  420  421  422  423  424
                         425  426  427  428  429  430  431  432
                         433  435  436  437  438  439  440  441
                         442  443  444  445  446  447  448  449
                         450  451  473  474  475  476  477  478
                         479  480  481  482  483  484  485  486
                         487  488  489  490  491  492  493  494
                         495  496  497  498  499  500  501  502
                         503  504  505  506  507  569  598  599
                         600  601  602  603  604  605  606  607
                         608  609  610  611  612  613  614  615
                         616  617  618  619  620  621  622  623
                         624  625  626  627  651  652  653  654
                         655  656  657  658  659  660  661  662
                         663  664  665  666  667  668  669  670
                         671  672  673  674  675  676  677  678
                         679  680  694  705  706  707  708  709
                         710  711  760
FILTRAM   D                0
FILTRAU   D              550  635  688  721
FILTRNM   D              434  693  712
GNDDF     A    00073    -121  122
GNHPF     A    00074    -122  123
GNINPUT   A    00070    -118  119  358
GNLPF     A    00072    -120  121  631  635  684  688
GNOUT     A    00076    -124  125  763
GNPRED    A    00075    -123  124  724
GNQMF     A    00071    -119  120
```

```
HFROUT    A     00155         -82
HPF       A     00116        -650
HPFBS     A     00136         -81    82   373   650   651   652   653   654
                              655   656   657   658   659   660   661   662
                              663   664   665   666   667   668   669   670
                              671   672   673   674   675   676   677   678
                              679   680   690   691   693   694   696
HPFSZ     A     00021         -67   112   373
INPUT     D                   358   361
ID        A     0000C           0
IDFD      A     00000           0
J         A     00009         -55    92    93
JMP       D                   177   190   327   351   352   353   364   366
                              368   370   372   374   376   378   453   509
                              557   558   572   577
JRF       A     00007           0
JSRP      A     00005           0
JZ        A     00000           0
LDCT      A     0000C           0
LDMR      D                   355   371   373   375   377   379   549   576
                              634   687   720   758
LEAST     A     003FF         -84   544   550   552   563   573   575   576
                              629   635   682   688   715   721
LOAD      D                   205   210   215   220   226   231   233   238
                              240   245   247   252   258   263   269   365
                              367   369   413   469   578   580   582   735
                              737   739   741   743   745   747   749
LOOP      D                     0
LP        A     0000B           0
LPF       A     000ED        -597
LPFBS     A     000C1         -75    76    77   371   579   581   583   597
                              598   599   600   601   602   603   604   605
                              606   607   608   609   610   611   612   613
                              614   615   616   617   618   619   620   621
                              622   623   624   625   626   627   637
LPFSZ     A     0001F         -57    76    82   109   371   637   690   691
                              696
LSTFE     A     00017         -98    99   369   369   553   553   578   578
                              580   580   582   582
LSTFEH    A     00016         -97    98   367   367   415   415
LSTFEL    A     00015         -96    97   365   365   471   471
LSTF      A     00014         -95    96   221   221   227   227   234   234
                              241   241   248   248   253   253   259   259
                              265   265   271   271   277   277
MEMEN     A     00004           0
MOST      A     003FE         -86   545   546   548   549   554   559   561
                              569   571   630   631   633   634   683   684
                              686   687   716   717   719   720
NEG       A     00002         557   572   577
NEGATE    A     000D7         557  -559
NEWAUTO   A     01000           0
NOMEM     A     006FF           0
NOPERT    A     000EB         572  -582
```

```
NORMAL     A      00000          0
NOTE       A      00068        370  -372
NOTFE      A      00066        368  -370
NOTFEH     A      00064        366  -368
NOTFEL     A      00062        364  -366
NOTHPF     A      0006A        372  -374
NOTS       A      00070        378  -380
NOTSPE     A      0006E        376  -378
NOTSUM     A      0006C        374  -376
OUTLP      A      00059        352  -355
OUTPUT     D                   170   171   356   380
P          A      00012        -93    94   205   205   210   210   215   215
                               231   231   238   238   245   245   263   263
                               269   269   362   362
P0         A      00000        -91    92   704   705   706   707   708   709
                               710   711   736   736   738   738   740   740
                               741   742   742   744   744   746   746   748
                               748   750   750
PAD        A      00003          0
PERT       A      000CA        453   509  -542
PERTBS     A      00133        -80    81   395
PERTP      A      000E9        577  -580
POSTV      A      000DA        558  -562
PPRT       A      00002       -132   361
PRED       A      00144        388  -704
PREDBS     A      00125        -77    78    79   377   704   705   706   707
                               708   709   710   711   712   723   724
PREDOUT    A      0012E        -78
PREDSCL    A      0006B       -112   113   114   712
PREDSZ     A      00009        -58    78    79
PREL       A      00100          0
PSH        A      00004          0
PTMP0      A      00009        -92    93   206   206   211   211   216   216
                               220   220   226   226   232   232   233   233
                               239   239   240   240   246   246   247   247
                               252   252   258   258   264   264   270   270
                               735   735   737   737   739   739   741   743
                               743   745   745   747   747   749   749
PUSH       D                   172   292
QMFBS      A      00000        -73    74   396   414   416   417   418   419
                               420   421   422   423   424   425   426   427
                               428   429   430   431   432   433   434   435
                               436   437   438   439   440   441   442   443
                               444   445   446   447   448   449   450   451
                               470   472   473   474   475   476   477   478
                               479   480   481   482   483   484   485   486
                               487   488   489   490   491   492   493   494
                               495   496   497   498   499   500   501   502
                               503   504   505   506   507
QMFH       A      00078        225   286   322  -413
QMFL       A      000A1        201   257   306  -469
QMFSZ      A      00024        -56    74   110
REPLP      D                   176   326
```

```
RET       D              0
RFCT      A   00008      0
RINGB     A   00000      0
RPCT      A   00009      0
S         A   00130    -83   355
SCRAT     A   00004    567   704   705   706   707   708   709   710
                       711
SHFTCLK   A   00800      0
SHFTLFT1  D              0
SHFTLFT2  D              0
SOFENT    A   0000A    177  -190   190   327
SPL       D            173   174   392   567
SPRT      A   00002   -134   170   170   356   356
STLR      D            544   548   552   561   571   575   629   633
                       682   686   696   715   719
STLS      D            360
STMR      D            175   395   396   414   470   545   554   579
                       581   583   630   637   683   690   716   723
                       726   762   765
STMS      D            206   211   216   221   227   232   234   239
                       241   246   248   253   259   264   265   270
                       271   277   362   553   566
STORE     D            415   471   736   738   740   742   744   746
                       745   750
SUB       A   00400      0
SUMBS     A   000E1    -76   375
TC        A   02000      0
TEMP      A   00018    -99   566   566   567
TEST      A   00001   -131   171   171   380   380
THRESH    A   00076   -125   563
TRUE      A   00000      0
TSL       A   00001      0
TSM       A   00002      0
TSTE      A   0000B   -141   370
TSTFE     A   0000A   -140   368
TSTFEH    A   00009   -139   366
TSTFEL    A   00008   -138   364
TSTHPF    A   0000C   -142   372
TSTS      A   0000F   -145   378
TSTSPE    A   0000E   -144   376
TSTSUM    A   0000D   -143   374
TWB       A   0000F      0
WAITLP    A   00057   -352   353
WRITE     A   00020      0
.LD       A   00184      0
.MARK     A   01000      0
.MULT1    A   02044    358   567
.MULTA    A   02244      0
.MULTAM   A   02644      0
.MULTAU   A   00244      0
.MULTL    A   020C0      0
.MULTN    A   022C4      0
.MULTNM   A   026C4      0
```

```
.NOP      A     00000        0
.SENBL    A     00005     -135   351   352
.SHIFT    A     00800      173   392
.SOF      A     00007      190
.STL      A     00039        0
.STM      A     0002A        0
.STR      A     00023        0
.TEST0    A     00008      138
.TEST1    A     00009      139
.TEST2    A     0000A      140
.TEST3    A     0000B      141
.TEST4    A     0000C      142
.TEST5    A     0000D      143
.TEST6    A     0000E      144
.TEST7    A     0000F      145
```

APPENDIX B

APPENDIX B

MULDEM SIMULATOR FIRMWARE LISTINGS

APPENDIX B

MULDEM SIMULATOR FIRMWARE LISTINGS

B1. 2-BIT ADAPTIVE RESIDUAL QUANTIZER

```
VAX/VMS 8086/8087/8088 MACRO ASSEMBLER V1.0VX ASSEMBLY OF MODULE M2LDEM
OBJECT MODULE PLACED IN M2LDEM.OBJ
NO INVOCATION LINE CONTROLS


LOC  OBJ               LINE    SOURCE

                        1
                        2
                        3                           NAME   M2LDEM
                        4
                        5
                        6
                        7           ;*******************************************************
                        8           ;*                                                     *
                        9           ;* PROJECT:     472A RELP CODEC                        *
                       10           ;* BOARD:       MULDEM SIMULATOR                       *
                       11           ;* DEVICE:      2 FIRMWARE EPROMS (2732A-2)            *
                       12           ;* BOARD LOCATION: E15, E29                            *
                       13           ;* PROGRAMMABLE                                        *
                       14           ;* DEVICE NUMBER: 60-0670, 60-0671                     *
                       15           ;*                                                     *
                       16           ;*******************************************************
                       17
                       18 +1   $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          19
                          20
                          21
                          22
                          23              ;*****************************************************
                          24              ;*                                                   *
                          25              ;* THIS FIRMWARE SIMULATES THE MULtiplexor/DEMulti-  *
                          26              ;* plexor IN THE RELP CODEC SYSTEM. REFER TO MDA     *
                          27              ;* DOCUMENT 00 - 3035 - R01 FOR DETAIL.              *
                          28              ;*                                                   *
                          29              ;*****************************************************
                          30
                          31
                          32
                          33
                          34      ;************************
                          35      ;*  SYSTEM CONSTANTS    *
                          36      ;************************
                          37
                          38
                          39
0020                      40      LEN_FE_BUFFER           EQU    32      ;ACCOMODATE 16 PAIRS OF FEX
                          41
0008                      42      LEN_P_BUFFER_FRAME      EQU    8       ;EACH FRAME ACCOMODATES 8 PREDICTOR COEFFICIE
                                  NTS
                          43
0020                      44      LAST_P_BUFFER_FRAME     EQU    32      ;THE BASE OFFSET ADDRESS OF LAST P FRAME
                          45                                             ;FROM THE BEGINNING OF P BUFFER. THIS VALUE
                          46                                             ;MUST BE IN MULTIPLE OF LEN_P_BUFFER_FRAME *
                                  TYPE P_BUFFER
                          47                                             ;THE FRAME DELAY IS EQUAL TO 1 +
                          48                                             ;(LAST_P_BUFFER_FRAME / 16)
                          49
                          50
0001                      51      FE_HIGH                 EQU    1       ;THIS REPRESENTS THE FEH'S TYPE
0000                      52      FE_LOW                  EQU    0       ;THIS REPRESENTS THE FEL'S TYPE
                          53
0002                      54      NLEV                    EQU    2       ;NUMBER OF QUANTIZER REGIONS
                          55
000A                      56      FSVMN                   EQU    10      ;FSVMN  - MINIMUM SCALING FACTOR
                          57
03E8                      58      FSVMX                   EQU    1000    ;FSVMX  - MAXIMUM SCALING FACTOR
                          59
000F                      60      FSVTH                   EQU    15      ;FSVTH  - THRESHOLD VALUE. IF THE
                          61                                             ;SCALE FACTOR IS LESS THAN FSVTH, A
                          62                                             ;MID-TREAD QUANTIZER IS USED INSTEAD
                          63                                             ;OF A MID-RISE QUANTIZER. THE MID-TREAD
                          64                                             ;QUANTIZER USES ZERO AS AN OUTPUT LEVEL
                          65                                             ;INSTEAD OF YQ(1).
                          66
0000                      67      POSITIVE                EQU    0       ;POSITIVE SIGN
                          68
0001                      69      NEGATIVE                EQU    1       ;NEGATIVE SIGN
                          70
                          71
```

```
LOC  OBJ               LINE    SOURCE

                        72     ;     HARDWARE DEPENDENT CONSTANTS
                        73
0042                    74     C_DATA_BUS               EQU      42H      ;THIS IS THE PORT(B) ADDRESS ON THE 8255'S
                        75                                                ;FOR THE 16 BIT CODER DATA BUS
                        76
0044                    77     C_CONTROL_BUS            EQU      44H      ;THIS IS THE PORT(C) ADDRESS ON THE 8255'S
                        78                                                ;FOR THE 16 BIT CODER CONTROL BUS
                        79
0046                    80     CODER_BUS_SPVR           EQU      46H      ;THIS IS THE SUPERVISOR PORT(CONTROL PORT)
                        81                                                ;FOR THE CODER BUS(THE CONTROL BUS AND
                        82                                                ;THE DATA BUS)
                        83
                        84
                        85
004A                    86     D_DATA_BUS               EQU      4AH      ;THIS IS THE PORT(B) ADDRESS ON THE 8255'S
                        87                                                ;FOR THE 16 BIT DECODER DATA BUS
                        88
                        89
004C                    90     D_CONTROL_BUS            EQU      4CH      ;THIS IS THE PORT(C) ADDRESS ON THE 8255'S
                        91                                                ;FOR THE 16 BIT DECODER CONTROL BUS
                        92
                        93
004E                    94     DECODER_BUS_SPVR         EQU      4EH      ;THIS IS THE SUPERVISOR PORT(CONTROL PORT)
                        95                                                ;FOR THE DECODER BUS(THE CONTROL BUS AND
                        96                                                ;THE DATA BUS)
                        97
                        98
                        99
                       100
                       101
                       102
0064                   103     ERROR_LED                EQU      64H      ;
                       104
0066                   105     ERROR_BUS_SPVR           EQU      66H      ;
                       106
                       107
                       108          ;8259A PIC DEPENDENT CONSTANTS
                       109
0078                   110     PIC_PORT_0               EQU      78H      ;8259A PROGRAMMABLE INTERRUPT CONTROLLER
007A                   111     PIC_PORT_1               EQU      7AH      ;CONTROL PORTS.
                       112
0013                   113     ICW1                     EQU      13H      ;SINGLE PIC, ICW4 NEEDED
0008                   114     ICW2                     EQU      08H      ;STARTING INTERRUPT VECTOR = 8
0003                   115     ICW4                     EQU      03H      ;SFNM = 0, AEOI = 1, NON-BUFFERED MODE
                       116                                                ;SP/EN WILL HAVE INPUT = 1 =>MASTER
                       117                                                ;UPM = 1 =>8086/88 SYSTEM.
00FE                   118     ENABLE_SOF_MASK          EQU      0FEH     ;MASK OUT ALL EXCEPT
                       119                                                ;SOF INTERRUPT(R0)
                       120
00E0                   121     ENABLE_NORMAL_MASK       EQU      0E0H     ;MASK OUT UNUSED
                       122                                                ;INTERRUPT(R7,R6,R5)
                       123
                       124
                       125
                       126          ;8255A PPI DEPENTENT CONTSTANTS
```

```
LOC  OBJ                LINE     SOURCE

                         127
  9B9B                   128     C_DATA_BUS_IN           EQU     9B9BH   ;PROGRAM THE CODER DATA BUS TO BE INPUT, ALL
                         129                                             ;OTHER PORTS ON CODER BUS ARE INPUT.
                         130                                             ;MSB AND LSB OF THIS WORD EACH ADDRESSES TO O
                                 NE PPI
                         131
  9999                   132     C_DATA_BUS_OUT          EQU     9999H   ;PROGRAM THE CODER DATA BUS TO BE OUTPUT(IE.
                         133                                             ;TAKE CONTROL OF THE DATA BUS), ALL OTHER
                         134                                             ;PORTS ON THE CODER BUS ARE INPUT. MSB
                         135                                             ;AND LSB OF THIS WORD EACH ADDRESSES ONE PPI.
                         136
  9B9B                   137     D_DATA_BUS_IN           EQU     9B9BH   ;PROGRAM THE DECODER DATA BUS TO BE INPUT, AL
                                 L
                         138                                             ;OTHER PORTS ON DECODER BUS ARE INPUT.
                         139                                             ;MSB AND LSB OF THIS WORD EACH ADDRESSES TO O
                                 NE PPI
                         140
  9999                   141     D_DATA_BUS_OUT          EQU     9999H   ;PROGRAM THE DECODER DATA BUS TO BE OUTPUT(IE
                                 .
                         142                                             ;TAKE CONTROL OF THE DATA BUS), ALL OTHER
                         143                                             ;PORTS ON THE DECODER BUS ARE INPUT. MSB
                         144                                             ;AND LSB OF THIS WORD EACH ADDRESSES ONE PPI.
                         145
  0092                   146     ERROR_LED_ON            EQU     0092H   ;PROGRAM THE ERROR BUS TO BE:
                         147                                             ;  PORT A - INPUT } NOT USED HERE
                         148                                             ;  PORT B - INPUT } NOT USED HERE
                         149                                             ;  PORT C - OUTPUT } TO ERROR_LED
                         150
  009B                   151     ERROR_LED_OFF           EQU     009BH   ;PROGRAM THE ERROR BUS TO BE:
                         152                                             ;  PORT A - INPUT } NOT USED HERE
                         153                                             ;  PORT B - INPUT } NOT USED HERE
                         154                                             ;  PORT C - INPUT } TO ERROR_LED
                         155                                             ;THE LED DISPLAY IN THIS STATE WILL
                         156                                             ;BE 'FF'
                         157
                         158
                         159          ;CONTROL BUS DEPENDENT CONSTANTS
                         160
                         161          ;DECODER BUS
                         162          ;-----------
                         163
  0004                   164     SPEN                    EQU     0004H   ;MASK FOR SPEN SIGNAL(LOW TRUE)
  0010                   165     SFEEN                   EQU     0010H   ;MASK FOR SFEEN SIGNAL(LOW TRUE)
                         166
                         167
                         168          ;CODER BUS
                         169          ;---------
                         170
  0800                   171     CFECLK                  EQU     0800H   ;MASK FOR CFECLK SIGNAL(HIGH TRUE)
  0020                   172     CPCLK                   EQU     0020H   ;MASK FOR CPCLK SIGNAL(HIGH TRUE)
                         173
                         174
                         175
                         176     ;MDA IN HOUSE MONITOR ENTRY POINT
                         177
```

```
LOC  OBJ                LINE     SOURCE

0010                     178     MONT_86_IP           EQU     0010H   ;OFFSET ADDRESS
FE9F                     179     MONT_86_CS           EQU     0FE9FH  ;SEGMENT ADDRESS
                         180
                         181 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           182
                           183
----                       184     INTERRUPT_VECTOR        SEGMENT          WORD AT 0H
                           185
                           186
                           187
                           188     ;                   ******************************
                           189     ;                   *  INTERRUPT VECTOR TABLE    *
                           190     ;                   ******************************
                           191
                           192     ;INTEL RESERVES INT 5 TO 32 FOR INTERNAL USES. CURRENT IMPLEMENTATION
                           193     ;                   VIOLATES THIS RESTRICTION.
                           194
                           195
                           196     ;8086 PREDIFINED INTERRUPTS: (INT 0 TO 4)
                           197
0000 (1                    198         DIVIDE_INT_IP      DW  1 DUP(?)
     ????
     )
0002 (1                    199         DIVIDE_INT_CS      DW  1 DUP(?)
     ????
     )
0004 (1                    200         SINGLE_STEP_IP     DW  1 DUP(?)
     ????
     )
0006 (1                    201         SINGLE_STEP_CS     DW  1 DUP(?)
     ????
     )
0008 (1                    202         NMI_IP             DW  1 DUP(?)
     ????
     )
000A (1                    203         NMI_CS             DW  1 DUP(?)
     ????
     )
000C (1                    204         BREAKPOINT_IP      DW  1 DUP(?)
     ????
     )
000E (1                    205         BREAKPOINT_CS      DW  1 DUP(?)
     ????
     )
0010 (1                    206         OVERFLOW_IP        DW  1 DUP(?)
     ????
     )
0012 (1                    207         OVERFLOW_CS        DW  1 DUP(?)
     ????
     )
                           208
                           209
                           210     ;MULDEM APPLICATION INTERRUPTS: (INT 8 TO 15)
                           211
0020                       212           ORG        20H
0020 (1                    213           INT_8_IP       DW  1 DUP(?)
     ????
     )
0022 (1                    214           INT_8_CS       DW  1 DUP(?)
LOC  OBJ                  LINE     SOURCE

     ????
     )
0024 (1                    215          INT_9_IP      DW  1 DUP(?)
     ????
     )
0026 (1                    216          INT_9_CS      DW  1 DUP(?)
     ????
     )
0028 (1                    217          INT_10_IP     DW  1 DUP(?)
     ????
     )
002A (1                    218          INT_10_CS     DW  1 DUP(?)
     ????
     )
002C (1                    219          INT_11_IP     DW  1 DUP(?)
     ????
     )
002E (1                    220          INT_11_CS     DW  1 DUP(?)
     ????
     )
0030 (1                    221          INT_12_IP     DW  1 DUP(?)
     ????
     )
0032 (1                    222          INT_12_CS     DW  1 DUP(?)
     ????
     )
0034 (1                    223          INT_13_IP     DW  1 DUP(?)
     ????
     )
0036 (1                    224          INT_13_CS     DW  1 DUP(?)
     ????
     )
0038 (1                    225          INT_14_IP     DW  1 DUP(?)
     ????
     )
003A (1                    226          INT_14_CS     DW  1 DUP(?)
     ????
     )
003C (1                    227          INT_15_IP     DW  1 DUP(?)
     ????
     )
003E (1                    228          INT_15_CS     DW  1 DUP(?)
     ????
     )
                           229
                           230
----                       231     INTERRUPT_VECTOR        ENDS
                           232
                           233
                           234
                           235 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          236
                          237
----                      238      DATA    SEGMENT
                          239
                          240          ;THIS SEGMENT WILL RESIDE IN RAM
                          241
0000 (24                  242          PREDICTOR_BUFFER   DW      (((LAST_P_BUFFER_FRAME/16)+1) * LEN_P_BUFFER_FRAME) D
                                   UP(?)
     ????
     )
0030 (32                  243          FE_BUFFER          DW      (LEN_FE_BUFFER) DUP(?)
     ????
     )
                          244
0070 (1                   245          LED_DISPLAY_VALUE  DW      1 DUP(?)
     ????
     )
                          246
0072 (1                   247          ABS_XIN            DW      1 DUP(?)
     ????
     )
0074 (1                   248          XOUT               DW      1 DUP(?)
     ????
     )
                          249
0076 (1                   250          FSV                DW      1 DUP(?)
     ????
     )
0078 (1                   251          FSVL               DW      1 DUP(?)
     ????
     )
007A (1                   252          FSVH               DW      1 DUP(?)
     ????
     )
                          253
007C (1                   254          I                  DW      1 DUP(?)
     ????
     )
007E (1                   255          IL                 DW      1 DUP(?)
     ????
     )
0080 (1                   256          L2                 DW      1 DUP(?)
     ????
     )
                          257
0082 (1                   258          FE_SERVICE_COUNTER DW      1 DUP(?)
     ????
     )
                          259
0084 (1                   260          FE_SERVICE_PTR     DW      1 DUP(?)
     ????
     )
                          261
0086 (1                   262          FE_TYPE            DW      1 DUP(?)
     ????
LOC  OBJ                  LINE     SOURCE

          )
                           263
0088 (1                    264         SIGN_FLAG          DW      1 DUP(?)
          ????
          )
                           265
                           266
----                       267     DATA    ENDS
                           268
                           269
                           270
                           271
                           272
                           273
                           274
                           275
                           276
----                       277     STACK                  SEGMENT
                           278
                           279             ;*********************************************
                           280             ;* THE STACK IS SOLELY USED BY THE 8086 TO    *
                           281             ;* STORE RETURN ADDRESS IN INTERRUPT ROUTINES*
                           282             ;*********************************************
0000 (60                   283                            DW      60 DUP (?)
          ????
          )
0078                       284     TOS                    LABEL   WORD
                           285
----                       286     STACK                  ENDS
                           287
                           288 +1  $EJECT
```

```
LOC  OBJ                    LINE     SOURCE

                             289
                             290
                             291
                             292     ;SYSTEM MACROS:
                             293     ;--------------
                             294
                             295
                             296     ;THE MACROS ARE NOT LISTED IN THE ASSEMBLER GENERATED LISTING, REFER
                             297     ;TO THE SOURCE FILE FOR MACRO CONTENTS.
                             298
                             299
                             300
                             301
                             302
                             303
                             304
                             305
                             306
                             307
                             308
                             309
                             310
                             311
                             312
                             313
                             314
                             315 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           316
                           317
                           318
                           319              ;*********************************************************
                           320              ;*                                                       *
                           321              ;*   REGISTER USAGE IN M2LDEM SIMULATOR SYSTEM :         *
                           322              ;*                                                       *
                           323              ;*********************************************************
                           324
                           325
                           326     ;DEDICATED USAGE FOR ALL PARTS OF THE SYSTEM AT ALL TIME:
                           327
                           328              ;SI      - FE_BUFFER INPUT POINTER
                           329              ;BX      - FE_BUFFER OUTPUT POINTER [THIS REGISTER IS ALSO USED UNDER
                           330              ;          NON-INTERRUPT DRIVEN QUANTIZATION PROCESS, HOWEVER, THE
                           331              ;          ORIGINAL REGISTER IS PRESERVED]
                           332              ;BP      - PREDICTOR_BUFFER FRAME POINTER
                           333              ;DI      - PREDICTOR_BUFFER OFFSET POINTER
                           334
                           335
                           336              ;CS      - CODE SEGMENT
                           337              ;DS      - DATA SEGMENT
                           338              ;ES      - DATA SEGMENT OR INTERRUPT_VECTOR SEGMENT
                           339              ;SS/SP   - STACK OPERATION
                           340
                           341
                           342     ;UNASSIGNED REGISTERS:
                           343
                           344              ;THESE REGISTERS DO NOT CARRY DEDICATED FUNCTIONS IN THE M2LDEM
                           345              ;SIMULATOR:
                           346
                           347              ;AX, CX, DX
                           348
                           349 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         350
                         351
                         352                PUBLIC         START_ADDR
                         353
                         354
----                     355      CODE                     SEGMENT
                         356
                         357                ASSUME         CS:CODE, DS:DATA, SS:STACK, ES:INTERRUPT_VECTOR
                         358
                         359
                         360
                         361                ;**********************************
                         362                ;*    STATIC    VARIABLES          *
                         363                ;**********************************
                         364
                         365
                         366                      ;YQ  -   ARRAY OF NLEV NORMALIZED QUANTIZER OUTPUT VALUES
                         367                      ;        (IN INCREASING ORDER)
                         368                      ;        VALUE SCALED BY ** IN ** REPRESENTATION
0000 0000                369                YQ             DW     0, 8192,  24576
0002 0020
0004 0060
                         370
                         371                      ;XQ  -   ARRAY OF NLEV-1 NORMALIZED QUANTIZER BREAK POINTS
                         372                      ;        (IN INCREASING ORDER)
                         373                      ;        VALUE SCALED BY ** IN ** REPRESENTATION
0006 0000                374                XQ             DW     0, 16384
0008 0040
                         375
                         376                      ;QMLT   - ARRAY OF NLEV, QUANTIZER MULTIPLIERS
                         377                      ;        VALUE SCALED BY ** IN ** REPRESENTATION
000A 0000                378                QMLT           DW     0, 27853, 62259
000C CD6C
000E 33F3
                         379
                         380
                         381 +1     $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           382
                           383     ;   ****************************************************************
                           384     ;   *                    M A I N     P R O G R A M                 *
                           385     ;   ****************************************************************
                           386
                           387
                           388     ;        ************************************************************
                           389     ;        *                                                          *
                           390     ;        *        REGISTER VALUES ARE NOT PRESERVED                 *
                           391     ;        *        IN ALL PROCEDURES. THEY SHOULD BE                 *
                           392     ;        *             SAVED BEFORE ENTERING.                       *
                           393     ;        *                                                          *
                           394     ;        ************************************************************
                           395
                           396
0010 FA                    397     START_ADDR:     CLI                     ;DISABLE EXTERNAL INTERRUPT
0011 B8----         R      398                     MOV     AX, DATA        ;CANNOT MOVE IMMED. VALUE TO
0014 8ED8                  399                     MOV     DS, AX          ;SEGMENT REGISTER.
                           400
0016 8EC0                  401                     MOV     ES, AX          ;ES AND DS ARE REFERING TO SAME
                           402                                             ;SEGMENT
                           403
                           404                     ;RESET THE LED_DISPLAY_VALUE
0018 C70670000000          405                     MOV     LED_DISPLAY_VALUE, 0
                           406
001E B89200                407                     MOV     AX, ERROR_LED_ON
0021 E666                  408                     OUT     ERROR_BUS_SPVR, AL
                           409
                           410                     ;OUTPUT ERROR DISPLAY VALUE
                           411
0023 B80000                412                     MOV     AX, 0
0026 E664                  413                     OUT     ERROR_LED, AL
                           414
                           415     ;        ***********************************************
                           416     ;        * INITIALIZE THE INTERRUPT VECTOR  TABLE      *
                           417     ;        *                                             *
                           418     ;        * THIS SUBSYSTEM USES FOLLOWING SIGNALS FROM  *
                           419     ;        * THE SYSTEM BUS:                             *
                           420     ;        *                                             *
                           421     ;        *      - START OF FRAME (SOF_)                *
                           422     ;        *      - FE ENABLE (SFEEN_)                   *
                           423     ;        *      - P ENABLE (SPEN_)                     *
                           424     ;        *      - FE CLOCK (CFECLK)                    *
                           425     ;        *      - P CLOCK (CPCLK)                      *
                           426     ;        ***********************************************
                           427
                           428
                           429
                           430
0028 B80000                431     ERROR_ENTRY:    MOV     AX, INTERRUPT_VECTOR
002B 8EC0                  432                     MOV     ES, AX                  ;USE EXTRA SEGMENT TO ADDRESS
                           433                                                     ;THE INTERRUPT VECTOR TABLE
                           434
002D 26C70608001000        435                     MOV     NMI_IP, MONT_86_IP
0034 26C7060A009FFE        436                     MOV     NMI_CS, MONT_86_CS
```

```
LOC  OBJ                  LINE     SOURCE

                           437
003B B80901                438                MOV     AX, OFFSET EXCEPTION_INT
003E BB----90       R      439                MOV     BX, SEG EXCEPTION_INT
0042 26A30000              440                MOV     DIVIDE_INT_IP, AX
0046 26891E0200            441                MOV     DIVIDE_INT_CS, BX
004B 26A31000              442                MOV     OVERFLOW_IP, AX
004F 26891E1200            443                MOV     OVERFLOW_CS, BX
                           444
                           445
0054 26C70620000000        446                MOV     INT_8_IP, OFFSET SOF_INTERRUPT
005B 26C7062200---- R      447                MOV     INT_8_CS, SEG SOF_INTERRUPT
                           448
0062 26C70624005D00        449                MOV     INT_9_IP, OFFSET PREDICTOR_OUTPUT_INTERRUPT
0069 26C7062600---- R      450                MOV     INT_9_CS, SEG PREDICTOR_OUTPUT_INTERRUPT
                           451
0070 26C70628008E00        452                MOV     INT_10_IP, OFFSET FE_OUTPUT_INTERRUPT
0077 26C7062A00---- R      453                MOV     INT_10_CS, SEG FE_OUTPUT_INTERRUPT
                           454
007E 26C7062C00EA00        455                MOV     INT_11_IP, OFFSET PREDICTOR_INPUT_INTERRUPT
0085 26C7062E00---- R      456                MOV     INT_11_CS, SEG PREDICTOR_INPUT_INTERRUPT
                           457
008C 26C70630008F00        458                MOV     INT_12_IP, OFFSET FE_INPUT_INTERRUPT
0093 26C7063200---- R      459                MOV     INT_12_CS, SEG FE_INPUT_INTERRUPT
                           460 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           461
                           462
                           463             ASSUME ES:DATA
                           464
                           465
                           466
                           467             ;INITIALIZE 8086 PROCESSOR ENVIRONMENT
                           468
009A B8----          R     469                     MOV     AX, DATA        ;CANNOT MOVE IMMED. VALUE TO
009D 8ED8                  470                     MOV     DS, AX          ;SEGMENT REGISTER.
                           471
009F 8EC0                  472                     MOV     ES, AX          ;ES AND DS ARE REFERING TO SAME
                           473                                             ;SEGMENT
00A1 B8----          R     474                     MOV     AX, STACK
00A4 8ED0                  475                     MOV     SS, AX
00A6 BC7500                476                     MOV     SP, OFFSET TOS
                           477
                           478
                           479             ;INITIALIZE ALL SYSTEM HARDWARE
                           480
                           481
                           482
                           483             ;       ******************
                           484             ;       *   P I C        *
                           485             ;       ******************
                           486
                           487
                           488             ;CAUTION:  AUTOMATIC EOI IS USED HERE. REFER TO INTEL APPLICATION
                           489             ;          NOTE AP-59 'USING THE 8259A PROGRAMMABLE INTERRUPT
                           490             ;          CONTROLLER', UNDER HEADING 'AUTOMATIC EOI MODE'
                           491
                           492
00A9 B013                  493                     MOV     AL, ICW1
00AB E676                  494                     OUT     PIC_PORT_0, AL
                           495
00AD BA7A00                496                     MOV     DX, PIC_PORT_1
00B0 B008                  497                     MOV     AL, ICW2
00B2 EE                    498                     OUT     DX, AL
                           499
                           500                     ;ICW 3 IS NOT NEEDED FOR CURRENT HARDWARE CONFIGURATION
                           501
00B3 B003                  502                     MOV     AL, ICW4
00B5 EE                    503                     OUT     DX, AL
                           504
00B6 B0FE                  505                     MOV     AL, ENABLE_SOF_MASK
00B8 EE                    506                     OUT     DX, AL
                           507
                           508
                           509             ;       ******************
                           510             ;       *   P P I        *
                           511             ;       ******************
                           512
                           513
                           514             ;INITIALLY ALL PORT ARE PROGRAMMED TO BE INPUT PORTS IN MODE 0
                           515             ;EXCEPT THE ERROR LED PORT, WHICH WILL BE OUTPUT ALL THE TIME
```

```
LOC  OBJ                  LINE     SOURCE

                           516
00B9 B89B9B                517                MOV     AX, C_DATA_BUS_IN
00BC E746                  518                OUT     CODER_BUS_SPVR, AX
                           519
                           520
00BE B89B9B                521                MOV     AX, D_DATA_BUS_IN
00C1 E74E                  522                OUT     DECODER_BUS_SPVR, AX
                           523 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           524
                           525           ;**********************************************
                           526           ;*  INITIALIZE APPLICATION PROGRAM ENVIRONMENT *
                           527           ;**********************************************
                           528
                           529
                           530
                           531
                           532
                           533                    ; FILL BUFFER AREAS WITH ZEROS
                           534
00C3 B80000                535                    MOV     AX, 0                         ;FILL VALUE
00C6 B91800                536                    MOV     CX, LENGTH PREDICTOR_BUFFER   ;ITERATION COUNT
00C9 BF0000                537                    MOV     DI, OFFSET PREDICTOR_BUFFER
                           538
                           539                    ;ES SETS TO THE SEGMENT BASE OF DATA SEGMENT
                           540
00CC F3                    541           REP      STOS    PREDICTOR_BUFFER
00CD AB
                           542
                           543
00CE B92000                544                    MOV     CX, LENGTH FE_BUFFER          ;ITERATION COUNT
00D1 BF3000                545                    MOV     DI, OFFSET FE_BUFFER
                           546
00D4 F3                    547           REP      STOS    FE_BUFFER
00D5 AB
                           548
                           549
                           550                    ;RESET SERVICE COUNTER
00D6 C70682000000          551                    MOV     FE_SERVICE_COUNTER, 0
                           552
                           553                    ;INITIALIZE FE TYPE TO LOW
00DC C70686000000          554                    MOV     FE_TYPE, FE_LOW
                           555
                           556
                           557                    ;INITIALIZE BUFFER POINTERS TO APPROPRIATE VALUES
                           558
                           559                    ;PREDICTOR BUFFER , BP IS THE FRAME POINTER,
                           560                    ;DI IS THE OFFSET POINTER WITHIN A FRAME.
                           561
00E2 BD2000                562                    MOV     BP, LAST_P_BUFFER_FRAME       ;ON FIRST SOF INTERRUPT AFTER
                           563                                                          ;POWER UP, SOF_INTERRUPT ROUT
                                   INE
                           564                                                          ;WILL SET BP = 0, DI = 0
                           565
00E5 BF1000                566                    MOV     DI, LEN_P_BUFFER_FRAME * (TYPE PREDICTOR_BUFFER)
                           567
                           568                    ;FE BUFFER INITIALIZATION
                           569
00E8 BE1000                570                    MOV     SI, 16                        ;POINTS TO FEL(4)
00EB BB0000                571                    MOV     BX, 0
                           572
                           573                    ;QUANTIZATION SERVICE
                           574
00EE C70684001000          575                    MOV     FE_SERVICE_PTR, 16
```

```
LOC  OBJ                LINE     SOURCE

00F4 C70678000A00        576                     MOV    FSVL, FSVMN
00FA C7067A000A00        577                     MOV    FSVH, FSVMN
                         578
                         579
                         580
0100 FB                  581                     STI                                   ;ENABLE EXTERNAL INTERRUPT
                         582
0101 F4                  583                     HLT                                   ;WAIT UNTIL THE FIRST
                         584                                                           ;START OF FRAME INTERRUPT
                         585
                         586 +1  $EJECT
```

```
LOC  OBJ            LINE   SOURCE



                                   ;**********************************************
                                   ;*                                            *
                                   ;*   START   OF  APPLICATION   PROGRAM         *
                                   ;*                                            *
                                   ;* AFTER POWER UP, THE FOLLOWING APPLICATION  *
                                   ;* PROGRAM WILL NOT START EXECUTION UNTIL THE *
                                   ;* FIRST SOF_INTERRUPT HAS BEEN SERVICED.     *
                                   ;**********************************************


0102                       LABEL_WAIT_FOR_NEW_FE:

                                         ;LOOP AROUND UNTIL THERE IS A NEW FE TO SERVICE

0102 39368400                            CMP     FE_SERVICE_PTR, SI
0106 74FA                                JE      LABEL_WAIT_FOR_NEW_FE

                                         ;TWO POINTERS ARE NOT EQUAL, POSSIBLE NEW FE'S


                                         ;CHECK SERVICE COUNTER
0108 833E820000                          CMP     FE_SERVICE_COUNTER, 0
010D 747C                                JE      LABEL_PTR_ERROR         ;THE SERVICE PTR AND
                                                                         ;INPUT POINTER(SI) ARE NOT
                                                                         ;EQUAL, SERVICE COUNTER = 0
                                                                         ;SYNC ERROR


                                         ;SERVICE COUNTER >=1 , NORMAL CONDITION

010F 833E820008                          CMP     FE_SERVICE_COUNTER, 8
0114 7375                                JAE     LABEL_PTR_ERROR         ;THERE ARE MORE THAN 4 PAIRS
                                                                         ;OF FEX TO BE SERVICED. CPU
                                                                         ;IS RUNNING TOO SLOW

                                         ;1 =< SERVICE COUNTER < 8, NORMAL CONDITION.
                                         ; BEGIN SERVICE THE FE'S



                        +1   $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          630
0116                      631     LABEL_SERVICE_FE:
                          632                             ;MOVE THE NEW FEX INTO AX AND CORRESPONDING FSV TO CX
                          633
                          634                             ;NEED TO USE THE BX REGISTER TO ACCESS THE FE_BUFFER
0116 8B0E8400             635                             MOV     CX, FE_SERVICE_PTR
                          636
011A FA                   637                             CLI
011B 87CB                 638                             XCHG    CX, BX
011D 8B4730               639                             MOV     AX, FE_BUFFER[BX]
0120 87CB                 640                             XCHG    CX, BX
0122 FB                   641                             STI
                          642
0123 833EB60000           643                             CMP     FE_TYPE, FE_LOW
0128 752C                 644                             JNE     LABEL_FE_HIGH
                          645
                          646
                          647
                          648
                          649                             ;IT IS FEL TO BE PROCESSED
012A 8B0E7800             650                             MOV     CX, FSVL
012E E86D00               651                             CALL    APCMQ
                          652
                          653                             ;RESULT FSV IN DX, XOUT IN CX
                          654
0131 89167800             655                             MOV     FSVL, DX                  ;SAVE THE NEW FSVL
                          656                             ;STORE THE NEW XOUT INTO THE FE_BUFFER
0135 A18400               657                             MOV     AX, FE_SERVICE_PTR
0138 FA                   658                             CLI
0139 93                   659                             XCHG    AX, BX
                          660
013A 894F30               661                             MOV     FE_BUFFER[BX], CX
013D FF0E8200             662                             DEC     FE_SERVICE_COUNTER
                          663
0141 93                   664                             XCHG    AX, BX
0142 FB                   665                             STI
                          666
                          667                             ;UPDATE THE FE TYPE TO BE SERVICED NEXT
0143 C70686000100         668                             MOV     FE_TYPE, FE_HIGH
                          669
                          670
                          671                             ;INCREMENT THE SERVICE POINTER, AX CONTAINS FE_SERVICE_PTR
0149 050200               672                             ADD     AX, TYPE FE_BUFFER
014C 3D4000               673                             CMP     AX, LEN_FE_BUFFER * TYPE FE_BUFFER       ;END OF BUFFER?
014F 7331                 674                             JAE     LABEL_WRAP_AROUND
0151 A38400               675                             MOV     FE_SERVICE_PTR, AX
0154 EBAC                 676                             JMP     LABEL_WAIT_FOR_NEW_FE
                          677
                          678 +1  $EJECT
```

```
LOC  OBJ             LINE     SOURCE

                      679
                      680
0156                  681     LABEL_FE_HIGH:  ;THE FEH IS TO BE PROCESSED.
0156 8B0E7A00         682                     MOV     CX, FSVH
015A E84100           683                     CALL    APCMD
                      684
                      685                     ;RESULT FSV IN DX, XOUT IN CX
                      686
015D 89167A00         687                     MOV     FSVH, DX                ;SAVE THE NEW FSVH
                      688
                      689                     ;STORE THE NEW XOUT INTO THE FE_BUFFER
0161 A16400           690                     MOV     AX, FE_SERVICE_PTR
                      691
0164 FA               692                     CLI
0165 93               693                     XCHG    AX, BX
                      694
0166 894F30           695                     MOV     FE_BUFFER[BX], CX
0169 FF0E6200         696                     DEC     FE_SERVICE_COUNTER
                      697
016D 93               698                     XCHG    AX, BX
016E FB               699                     STI
                      700
                      701                     ;UPDATE THE FE TYPE TO BE SERVICED NEXT
016F C70686000000     702                     MOV     FE_TYPE, FE_LOW
                      703
                      704
                      705                     ;INCREMENT THE SERVICE POINTER
0175 050200           706                     ADD     AX, TYPE FE_BUFFER
0178 3D4000           707                     CMP     AX, LEN_FE_BUFFER * TYPE FE_BUFFER      ;END OF BUFFER?
017B 7305             708                     JAE     LABEL_WRAP_AROUND
017D A36400           709                     MOV     FE_SERVICE_PTR, AX
                      710
0180 EB80             711                     JMP     LABEL_WAIT_FOR_NEW_FE
                      712
                      713
0182                  714     LABEL_WRAP_AROUND:
                      715                     ;SERVICE POINTER WAS POINTING TO THE LAST ELEMENT IN FE
                      716                     ;ARRAY, MOVE TO THE FIRST ELEMENT.
                      717
0182 C70684000000     718                     MOV     FE_SERVICE_PTR, 0
0188 E977FF           719                     JMP     LABEL_WAIT_FOR_NEW_FE
                      720
                      721 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         722
018B                     723     LABEL_PTR_ERROR:
                         724
                         725             ;*****************************************************************
                         726             ;* THIS PART OF CODE HANDLES THE SYNCHRONIZATION ERROR           *
                         727             ;* BETWEEN THE QUANTIZATION PROCESS AND THE VARIOUS              *
                         728             ;* INTERRUPT DRIVEN I/O PROCESSES.                               *
                         729             ;*                                                               *
                         730             ;* AN ERROR CONDITION MAY BE ONE OR MORE OF THE FOLLOWINGS:      *
                         731             ;*                                                               *
                         732             ;* (1) BOTH INPUT(REGISTER SI - DEDICATED) AND SERVICE POINTER *
                         733             ;*     ARE POINTING TO DIFFERENT ELEMENT IN THE FE_BUFFER        *
                         734             ;*     AND THE SERVICE COUNTER HAS A VALUE OF ZERO.              *
                         735             ;* (2) THE SERVICE COUNTER HAS A VALUE HIGHER THAN 8. THIS       *
                         736             ;*     MEANS THAT FOUR PAIRS OF FEX OR MORE ARE NOT QUANTIZED  *
                         737             ;*     YET.                                                      *
                         738             ;* THE MULDEM SIMULATOR PROCESS WILL CONTINUE ONCE THE           *
                         739             ;* ABOVE ERROR(S) OCCURS. THE FOLLOWING ROUTINE WILL CAUSE THE *
                         740             ;* MOST SIGNIFICANT ERROR_LED TO INCREMENT BY 1.                 *
                         741             ;*****************************************************************
                         742
                         743
018B FA                  744                     CLI                               ;DISABLE EXTERNAL INTERRUPT
018C A17000              745                     MOV     AX,LED_DISPLAY_VALUE
018F 80C410              746                     ADD     AH,10H
0192 7207                747                     JC      LABEL_PTR_EXT
0194 A37000              748                     MOV     LED_DISPLAY_VALUE,AX
0197 0AC4                749                     OR      AL,AH
0199 E664                750                     OUT     ERROR_LED,AL
019B E98AFE              751     LABEL_PTR_EXT:  JMP     ERROR_ENTRY
                         752
                         753                     ;*********************************
                         754                     ;*    END  OF  MAIN  PROGRAM     *
                         755                     ;*********************************
                         756
                         757
                         758
                         759 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE



019E                               APCMQ          PROC           NEAR


                                   ;*********************************************************************
                                   ;* PURPOSE:                                                          *
                                   ;*       THIS ROUTINE QUANTIZES A SAMPLE USING AN ADAPIVE QUANTIZER.  *
                                   ;*                                                                   *
                                   ;*                                                                   *
                                   ;* DESCRIPTION:                                                      *
                                   ;*       THE INPUT SAMPLE IS QUANTIZED, USING THE GIVEN SCALING FACTOR *
                                   ;*       FOR THE QUANTIZER. THE SCALING FACTOR IS UPDATED ON RETURN.  *
                                   ;*                                                                   *
                                   ;*                                                                   *
                                   ;* PARAMETERS:                                                       *
                                   ;*       INPUT :                                                     *
                                   ;*         XIN    - INPUT SAMPLE(PASSED IN AX REGISTER)              *
                                   ;*         FSV    - SCALING FACTOR FOR THE QUANTIZER. THIS VALUE IS  *
                                   ;*                  UPDATED ON OUTPUT.(INPUTED IN CX REGISTER)       *
                                   ;*                                                                   *
                                   ;*         OUTPUT:                                                   *
                                   ;*         XOUT   - OUTPUT QUANTIZED SAMPLE (PASSED IN CX REGISTER)  *
                                   ;*         FSV    - NEW SCALING FACTOR FOR THE QUANTIZER (OUTPUT     *
                                   ;*                  IN DX REGISTER)                                  *
                                   ;*                                                                   *
                                   ;*       PRE-SPECIFIED:                                              *
                                   ;*         NLEV   - NUMBER OF POSITIVE QUANTIZER LEVELS. THE QUANTIZER *
                                   ;*                 IS ASSUMED TO BE SYMMETRIC ABOUT ZERO. THE TOTAL   *
                                   ;*                 NUMBER OF QUANTIZER LEVELS IS 2*NLEV.             *
                                   ;*                                                                   *
                                   ;* ROUTINES REQUIRED:                                                *
                                   ;*      IQUANTZ - QUANTIZE A POINT                                   *
                                   ;*********************************************************************

                          +1       $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           796
                           797
019E 890E7600              798                      MOV    FSV, CX               ;FSV IS PASSED TO THIS PROCEDURE FROM
                           799                                                   ;REGISTER CX. SAVE FOR FUTURE REFEREN
                                   CE
                           800                                                   ;THIS VALUE IS ALSO USED IN PROCEDURE
                                    IQUANTZ
                           801
                           802                      ;XIN IS PASSED TO THIS PROCEDURE IN REGISTER AX.
                           803
01A2 C70688000000          804                      MOV    SIGN_FLAG, POSITIVE   ;INITIALIZE THE SIGN FLAG
                           805
01A8 A9FFFF                806                      TEST   AX, 0FFFFH            ;GET SIGN
01AB 7908                  807                      JNS    LABEL_1               ;IT IS A POSITIVE NUMBER
                           808
                           809                      ;XIN IS A NEGATIVE NUMBER, GET ABSOLUTE VALUE
                           810
01AD F7D8                  811                      NEG    AX
01AF C70688000100          812                      MOV    SIGN_FLAG, NEGATIVE   ;STORE STATE OF SIGN
                           813
01B5 A37200                814     LABEL_1:         MOV    ABS_XIN, AX           ;SAVE THE ABSOLUTE XIN FOR
                           815                                                   ;USE IN IQUANTZ PROCEDURE
                           816
                           817
                           818                      ;REGISTER AX, CX AND DX ARE NOT PRESERVED IN THIS CALL.
01B8 E86300                819                      CALL   IQUANTZ
                           820
                           821                      ;RESULT 'L2' RETURNED IN CX. THIS VALUE IS DOUBLED THE
                           822                      ;ACTUAL VALUE TO FACILITATE INDEX ADDRESSING.
                           823
01BB 890E8000              824                      MOV    L2, CX                ;SAVE FOR FUTURE REFENRECE
                           825
                           826 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         827                     ;*********************************************
                         828                     ;* GENERATE QUANTIZED VALUE FOR THE CODED BITS *
                         829                     ;*********************************************
                         830
01BF A17600              831                     MOV     AX, FSV
                         832
01C2 3D0F00              833                     CMP     AX, FSVTH
                         834
01C5 730E                835                     JAE     LABEL_MID_RISE
                         836
                         837                     ;FSV < FSVTH
                         838
                         839
01C7 83F902              840                     CMP     CX, 2                   ;L2 = 1?
                         841
01CA 7509                842                     JNE     LABEL_MID_RISE
                         843
                         844                     ;L2 = 1 , USE MID_TREAD QUANTIZER AND SET XOUT = 0
                         845
01CC C70674000000        846                     MOV     XOUT, 0
01D2 EB1B90              847                     JMP     INC_STEP
                         848
                         849
01D5                     850     LABEL_MID_RISE:
                         851
                         852                     ;NEED TO USE BX AT THIS POINT, DISABLE INTERRUPT BEFORE USE
                         853                     ;BX IS USED BY INTERRUPT I/O ROUTINES AS A DEDICATED REGISTER
                         854
01D5 FA                  855                     CLI
01D6 87D9                856                     XCHG    BX, CX                  ;CX CONTAINS 'L2'
                         857
01D8 2EF727              858                     MUL     CS:YQ[BX]               ;AX CONTAINS FSV
                         859
01DB 87D9                860                     XCHG    BX, CX
                         861
01DD FB                  862                     STI
                         863
01DE D1C0                864                     ROL     AX, 1                   ;SCALE BY 2**15
01E0 D1D2                865                     RCL     DX, 1
                         866
                         867                     ;INCORPORATE SIGN BIT
01E2 833EBB0000          868                     CMP     SIGN_FLAG, 0
01E7 7402                869                     JZ      LABEL_2
                         870
                         871                     ;XIN WAS A NEGATIVE NUMBER
                         872
01E9 F7DA                873                     NEG     DX
                         874
01EB 89167400            875     LABEL_2:        MOV     XOUT, DX
                         876
                         877 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           878
                           879                 ;*******************************
                           880                 ;* INCREMENT STEP SIZE OF FSV *
                           881                 ;*******************************
                           882
                           883
01EF BB0E8000              884     INC_STEP:   MOV     CX, L2
                           885
                           886                 ;NEED TO USE BX AGIAN, REFER TO PREVIOUS USAGE FOR DOCUMENTATION
                           887
01F3 FA                    888                 CLI
01F4 87CB                  889                 XCHG    CX, BX
01F6 2E8B470A              890                 MOV     AX, CS:QMLT[BX]
01FA 87CB                  891                 XCHG    CX, BX
01FC FB                    892                 STI
01FD F7267600              893                 MUL     FSV
                           894
0201 D1C0                  895                 ROL     AX, 1                       ;SCALE RESULT BY 2**15
0203 D1D2                  896                 RCL     DX, 1
                           897
0205 81FAE803              898                 CMP     DX, FSVMX
0209 7606                  899                 JBE     LABEL_3
020B BAE803                900                 MOV     DX, FSVMX
020E EB0990                901                 JMP     APCMG_END
0211 83FA0A                902     LABEL_3:    CMP     DX, FSVMN
0214 7303                  903                 JAE     APCMG_END
0216 BA0A00                904                 MOV     DX, FSVMN
                           905
                           906                 ;NEW FSV IN DX
                           907
0219 8B0E7400              908     APCMG_END:  MOV     CX, XOUT
021D C3                    909                 RET
                           910
                           911     APCMG       ENDP
                           912
                           913
                           914
                           915 +1  $EJECT
```

```
LOC  OBJ          SOURCE

021E              IQUANTZ             PROC          NEAR

                       ;*****************************************************************
                       ;* INPUT :           ABS_XIN , FSV PASSED FROM MEMORY            *
                       ;*                                                               *
                       ;* OUTPUT:           L        IN REGISTER CX. THIS VALUE IS     *
                       ;*                            DOUBLE THAT OF ACTUAL VALUE TO     *
                       ;*                            FACILITATE INDEX ADDRESSING        *
                       ;* REGISTER USAGE:                                              *
                       ;*                   BX       TEMPORARILY, VALUE SAVED AND       *
                       ;*                            RESTORED BY THIS PROCEDURE.        *
                       ;*                            INTERRUPT DISABLED WHILE USING THIS*
                       ;*                            REGISTER.                          *
                       ;*                   DX                                          *
                       ;*                                                               *
                       ;* REGISTER ASSIGNMENT IN THIS PROCEDURE:                        *
                       ;*                                                               *
                       ;*                   CX       CONTAINS IU                        *
                       ;*          REFER TO FORTRAN LISTING FOR DESCRIPTION OF THESE    *
                       ;*          VARIABLES.                                           *
                       ;*****************************************************************


              +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           941
021E C7067E000000          942                 MOV    IL, 0                  ;IL = 0
0224 B90400                943                 MOV    CX, NLEV* TYPE XQ      ;IU = NLEV [* 2 FACTOR ADDED]
                           944
                           945                 ;I = (IL + IU) /2
                           946
0227 8B167E00              947     LABEL_100:  MOV    DX, IL
022B 03D1                  948                 ADD    DX, CX
022D D1EA                  949                 SHR    DX, 1                  ;DIVIDE BY 2
022F 81E2FEFF              950                 AND    DX, 0FFFEH             ;MAKE SURE IT IS AN EVEN NUMB
                                   ER
0233 89167C00              951                 MOV    I, DX                  ;SAVE I FOR FUTURE USE
                           952                 ;IF (X .GT. XQ(I)*FSV) GOTO 220
                           953
                           954                 ;NEED TO USE REGISTER BX AT THIS POINT. DISABLE INTERRUPT BEFORE
                           955                 ;USING
                           956
                           957
0237 FA                    958                 CLI
0238 87D3                  959                 XCHG   DX, BX
023A 2E8B4706              960                 MOV    AX, CS:XQ[BX]          ;MOV XQ TO AX REGISTER
023E 87D3                  961                 XCHG   DX, BX
0240 FB                    962                 STI
                           963
0241 F7267600              964                 MUL    FSV                    ;XQ IS SCALED BY 2**15
0245 D1C0                  965                 ROL    AX, 1
0247 D1D2                  966                 RCL    DX, 1
0249 39167200              967                 CMP    ABS_XIN, DX
024D 7707                  968                 JA     LABEL_200
                           969
                           970                 ;IU = I
024F 8B0E7C00              971                 MOV    CX, I
0253 EB0990                972                 JMP    LABEL_300
                           973
0256                       974     LABEL_200:  ;IL = I
0256 8B167C00              975                 MOV    DX, I
025A 89167E00              976                 MOV    IL, DX
                           977
                           978
025E 8B167E00              979     LABEL_300:  MOV    DX, IL
0262 83C202                980                 ADD    DX, 2                  ;IL+ 1
0265 3BCA                  981                 CMP    CX, DX
0267 77BE                  982                 JA     LABEL_100
                           983
                           984                 ;RESULT IU IN CX REGISTER
                           985
                           986
0269 C3                    987                 RET
                           988
                           989     IQUANTZ     ENDP
                           990
                           991
                           992
                           993
                           994
```

```
LOC  OBJ                  LINE     SOURCE

                           995
                           996
                           997
----                       998     CODE                 ENDS
                           999
                          1000
                          1001
                          1002 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          1003
                          1004
                          1005
                          1006
----                      1007     INTERRUPT_CODE          SEGMENT
                          1008
                          1009              ASSUME         CS:INTERRUPT_CODE, DS:DATA, ES:NOTHING, SS:STACK
                          1010
0000                      1011     SOF_INTERRUPT           PROC            FAR
                          1012
                          1013     ;**************************************************************************
                          1014     ;* THIS INTERRUPT ROUTINE IS ACTIVATED BY THE SOF_(START_OF_FRAME) SINGNAL *
                          1015     ;* ON THE CODER BUS(THE SOF_ ON THE DECODER BUS IS SYNCHRONIZED AND OCCURS *
                          1016     ;* AT THE SAME TIME AS THE DECODER BUS SOF_).                              *
                          1017     ;* THIS ROUTINE IS RESPONSIBLE FOR CHECKING THE MULDEM OVERALL FIRMWARE    *
                          1018     ;* STATE, INCLUDING FOLLOWING:                                             *
                          1019     ;*              - FE INPUT AND OUTPUT POINTER VALUES                       *
                          1020     ;*              - PREDICTOR COEFFICIENTS BUFFER FRAME AND OFFSET POINTER   *
                          1021     ;*                VALUES                                                   *
                          1022     ;*                                                                         *
                          1023     ;* IF ANY OF THESE VALUES IS ABNORMAL, THIS ROUTINE WILL ATTEMP TO CORRECT *
                          1024     ;* TO THE BEST OF ITS KNOWLEDGE. AT THE SAME TIME, IT WILL INCREMENT THE   *
                          1025     ;* ERROR LOG VALUE AND OUTPUT TO THE LEAST SIGNIFICANT LED.                *
                          1026     ;* THIS ROUTINE IS ALSO RESPONSIBLE FOR SETTING UP THE PRIDICTION COEFF    *
                          1027     ;* BUFFER POINTERS FOR THE CURRENT FRAME.                                  *
                          1028     ;*                                                                         *
                          1029     ;*             REGISTER USAGES:                                            *
                          1030     ;*                        AX, BX, BP, SI, DI                               *
                          1031     ;*                                                                         *
                          1032     ;**************************************************************************
                          1033
                          1034
                          1035 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                        1036
                        1037                    ;NOTE THAT INTERRUPT FLAG (IF) IS DISABLED IN MOST PART
                        1038                    ;OF THIS ROUTINE.
                        1039
0000 50                 1040                            PUSH    AX                      ;MAY USE AX REGISTER TO
                        1041                                                            ;OUTPUT ERROR_LED IN THIS
                        1042                                                            ;ROUTINE
                        1043                    ;************************************
                        1044                    ;*   FE_BUFFER      CHECK          *
                        1045                    ;************************************
                        1046
                        1047
                        1048
                        1049                            ;CHECK FE INPUT POINTER
                        1050
0001 83FE10             1051                            CMP     SI, 16                  ;THIS IS THE NORMAL OFFSET
                        1052                                                            ;VALUE POINTING TO FEL(4)
0004 7413               1053                            JE      LABEL_NORMAL_1
                        1054
                        1055 +1
                        1056 +1
                        1057 +1
                        1058 +1
0006 A17000             1059 +1                         MOV     AX,LED_DISPLAY_VALUE
0009 FEC0               1060 +1                         INC     AL
000B 240F               1061 +1                         AND     AL,0FH
                        1062
000D 7407               1063                            JZ      LED_ERROR_EXT1
                        1064
                        1065 +1
000F A37000             1066 +1                         MOV     LED_DISPLAY_VALUE, AX
0012 0AC4               1067 +1                         OR      AL,AH
0014 E664               1068 +1                         OUT     ERROR_LED, AL
                        1069
0016 BE1000             1070     LED_ERROR_EXT1:        MOV     SI, 16                  ;CORRECT POINTER VALUE
                        1071
0019                    1072     LABEL_NORMAL_1:        ;CHECK FE OUTPUT POINTER
                        1073
0019 83FB00             1074                            CMP     BX, 0                   ;THIS IS THE NORMAL VALUE
                        1075                                                            ;POINTING TO FEL(0)
001C 7413               1076                            JE      LABEL_NORMAL_2
                        1077
                        1078 +1
                        1079 +1
                        1080 +1
                        1081 +1
001E A17000             1082 +1                         MOV     AX,LED_DISPLAY_VALUE
0021 FEC0               1083 +1                         INC     AL
0023 240F               1084 +1                         AND     AL,0FH
                        1085
0025 7407               1086                            JZ      LED_ERROR_EXT2
                        1087
                        1088 +1
0027 A37000             1089 +1                         MOV     LED_DISPLAY_VALUE, AX
002A 0AC4               1090 +1                         OR      AL,AH
```

```
LOC  OBJ                  LINE     SOURCE

002C E664                 1091 +1                  OUT    ERROR_LED, AL
                          1092
002E BB0000               1093     LED_ERROR_EXT2:      MOV    BX, 0                        ;CORRECT POINTER VALUE
                          1094
                          1095                     ;************************************
                          1096                     ;*   PREDICTOR BUFFER CHECK         *
                          1097                     ;************************************
                          1098
0031                      1099     LABEL_NORMAL_2:      ;CHECK PREDICTOR INPUT POINTER
                          1100
0031 83FF10               1101                     CMP    DI, LEN_P_BUFFER_FRAME * (TYPE PREDICTOR_BUFFER)
                          1102                                                  ;THIS IS THE NORMAL VALUE
                          1103                                                  ;POINTING TO ELEMENT AFTER
                          1104                                                  ;THE LAST ELEMENT OF A FRAME
                          1105
0034 7410                 1106                     JE     LABEL_NORMAL_3
                          1107
                          1108                     ;THE POINTER CONTAINS ILLEGAL VALUE
                          1109
                          1110
                          1111 +1
                          1112 +1
                          1113 +1
                          1114 +1
0036 A17000               1115 +1                  MOV    AX,LED_DISPLAY_VALUE
0039 FEC0                 1116 +1                  INC    AL
003B 240F                 1117 +1                  AND    AL,0FH
                          1118
003D 7407                 1119                     JZ     LABEL_NORMAL_3
                          1120
                          1121 +1
003F A37000               1122 +1                  MOV    LED_DISPLAY_VALUE, AX
0042 0AC4                 1123 +1                  OR     AL,AH
0044 E664                 1124 +1                  OUT    ERROR_LED, AL
                          1125
0046                      1126     LABEL_NORMAL_3:      ;SET THE PREDICTOR POINTERS FOR NEXT FRAME OUTPUT
                          1127
0046 BF0000               1128                     MOV    DI, 0
0049 83C510               1129                     ADD    BP, LEN_P_BUFFER_FRAME *(TYPE PREDICTOR_BUFFER)
                          1130                                                  ;INCREMENT TO NEXT FRAME
                          1131
004C 83FD20               1132                     CMP    BP, LAST_P_BUFFER_FRAME ;DETERMINE IF BP POINTS TO
                          1133                                                  ;LAST FRAME IN BUFFER
                          1134
004F 7603                 1135                     JBE    LABEL_SOF_END
                          1136
                          1137                     ;BP WAS POINTING TO LAST FRAME IN BUFFER BEFORE INCREMENT
                          1138
0051 BD0000               1139                     MOV    BP, 0                        ;MOVE TO THE FIRST FRAME IN B
                                   UFFER
                          1140
0054                      1141 +1  LABEL_SOF_END:
                          1142 +1
                          1143 +1
                          1144 +1
```

```
LOC  OBJ                  LINE     SOURCE

0054 FA                   1145 +1                          CLI
0055 B8E000               1146 +1                          MOV    AX, ENABLE_NORMAL_MASK
0058 E67A                 1147 +1                          OUT    PIC_PORT_1, AL
005A FB                   1148 +1                          STI
005B 58                   1149                             POP    AX
005C CF                   1150                             IRET
                          1151
                          1152
                          1153     SOF_INTERRUPT           ENDP
                          1154
                          1155
                          1156
                          1157 +1  $EJECT
```

```
LOC  OBJ          LINE     SOURCE


005D                       PREDICTOR_OUTPUT_INTERRUPT      PROC            FAR


                           ;*****************************************************************************
                           ;* THIS PROCEDURE IS ACTIVATED BY THE SPEN_ SIGNAL ON THE DECODER BUS.       *
                           ;* THIS ROUTINE IS RESPONSIBLE FOR OUTPUT THE PREDICTOR COEFFICIENTS FROM    *
                           ;* THE PREDICTOR COEFF BUFFER.                                               *
                           ;*                                                                           *
                           ;*                   REGISTER USAGES:                                        *
                           ;*                           AX, DI                                          *
                           ;*                                                                           *
                           ;*****************************************************************************


005D 50                                            PUSH    AX                          ;AX IS USED DURING THIS ROUTI
                           NE.




005E B8FE00                                        MOV     AX,ENABLE_SOF_MASK
0061 E678                                          OUT     PIC_PORT_1, AL
0063 FB                                            STI


                                                   ;CONFIGURE THE DECODER DATA BUS TO BE OUTPUT

0064 B89999                                        MOV     AX, D_DATA_BUS_OUT
0067 E74E                                          OUT     DECODER_BUS_SPVR, AX

                                                   ;OUTPUT DATA

0069 3E8B03                                        MOV     AX, DS:PREDICTOR_BUFFER[BP][DI]

006D E744                                          OUT     D_DATA_BUS, AX

                                                   ;INCREMENT BUFFER POINTER

006E 83C702                                        ADD     DI, TYPE PREDICTOR_BUFFER
0071 83FF10                                        CMP     DI, LEN_P_BUFFER_FRAME * TYPE PREDICTOR_BUFFER
0074 7203                                          JB      LABEL_P_EN

                                                   ;DI POINTS BEYOND END OF CURRENT FRAME, END OF OUTPUT
                                                   ;OF A FRAME(THE N - 2ND FRAME) PREDICTOR COEFFICIENTS.
                                                   ;SET UP THE POINTER FOR INPUT NEW PREDICTOR
                                                   ;COEFFICIENTS AT END OF CURRENT FRAME(FRAME N).
                                                   ;THE INPUT DATA WILL BE DEPOSITED INTO THE SAME FRAME
                                                   ;BUFFER, THEREFORE BP REGISTER NEEDS NOT BE CHANGED.

0076 BF0000                                        MOV     DI, 0


0079                       LABEL_P_EN:             ;CHECK FOR END OF OUTPUT CYCLE
0079 E54C                                          IN      AX, D_CONTROL_BUS
```

```
LOC  OBJ                 LINE     SOURCE

007B 250400              1212                         AND      AX, SPEN
007E 74F9                1213                         JZ       LABEL_P_EN
                         1214
                         1215                         ;NOW SPEN IS FALSE(I.E. SIGNAL IS HIGH), TAKE DATA OFF THE BU
                                  S
                         1216
0080 B89B9B              1217                         MOV      AX, D_DATA_BUS_IN
0083 E74E                1218                         OUT      DECODER_BUS_SPVR, AX
                         1219
                         1220 +1
                         1221 +1
                         1222 +1
                         1223 +1
0085 FA                  1224 +1                      CLI
0086 B8FD00              1225 +1                      MOV      AX, ENABLE_NORMAL_MASK
0089 E67A                1226 +1                      OUT      PIC_PORT_1, AL
008B FB                  1227 +1                      STI
                         1228
008C 58                  1229                         POP      AX
008D CF                  1230                         IRET
                         1231
                         1232
                         1233     PREDICTOR_OUTPUT_INTERRUPT       ENDP
                         1234
                         1235
                         1236
                         1237 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                        1238
                        1239
008E                    1240     FE_OUTPUT_INTERRUPT      PROC     FAR
                        1241
                        1242
                        1243     ;***************************************************************************
                        1244     ;* THIS INTERRUPT HANDLER IS ACTIVATED BY FE ENABLE_ SIGNAL ON THE DECODER *
                        1245     ;* BUS. IT IS RESPONSIBLE FOR OUTPUTING FEH AND FEL TO THE DECODER  BUS    *
                        1246     ;* ON ENTRY TO THE HANDLER, AX IS PUSHED ONTO STACK, DATA BUS PORT IS      *
                        1247     ;* CONVERTED FROM INPUT STATE TO OUTPUT STATE, THE APPROPRIATE FEX IS      *
                        1248     ;* OUTPUTED TO THE BUS, POINTER TO BUFFER IS INCREMENTED TO NEXT FEX TO BE *
                        1249     ;* OUTPUT. THIS ROUTINE WILL LOOP AROUND UNTIL FE ENDABLE_ IS HIGH, THEN   *
                        1250     ;* IT WILL PULL DATA OFF FROM THE DATA BUS, CONVERT PORT TO INPUT STATE    *
                        1251     ;* AND EXIT                                                                *
                        1252     ;*                                                                         *
                        1253     ;* REGISTER USAGE:                                                         *
                        1254     ;*                  AX, BX                                                 *
                        1255     ;***************************************************************************
                        1256
                        1257
008E 50                 1258                              PUSH     AX
                        1259
                        1260 +1
                        1261 +1
                        1262 +1
                        1263 +1
008F B8FE00             1264 +1                           MOV      AX,ENABLE_SOF_MASK
0092 E621               1265 +1                           OUT      PIC_PORT_1, AL
0094 FB                 1266 +1                           STI
                        1267
                        1268
                        1269                              ;CONFIGURE DATA BUS PORT TO BE OUTPUT
                        1270
0095 B89999             1271                              MOV      AX, D_DATA_BUS_OUT
0098 E74E               1272                              OUT      DECODER_BUS_SPVR, AX
                        1273
009A 8B4730             1274                              MOV      AX, FE_BUFFER[BX]
                        1275
                        1276                              ;BX IS THE POINTER TO THE NEXT WORD TO BE OUTPUT IN FE BUFFER
                        1277
009D E74A               1278                              OUT      D_DATA_BUS, AX
                        1279
                        1280                              ;INCREMENT THE OUTPUT POINTER
                        1281
009F 83C302             1282                              ADD      BX, TYPE FE_BUFFER
00A2 83FB40             1283                              CMP      BX, LEN_FE_BUFFER * TYPE FE_BUFFER
00A5 7203               1284                              JB       LABEL_FE_ENABLE
00A7 BB0000             1285                              MOV      BX, 0
                        1286
                        1287
                        1288
00AA E54C               1289     LABEL_FE_ENABLE:         IN       AX, D_CONTROL_BUS
00AC 251000             1290                              AND      AX, SFEEN
00AF 74F9               1291                              JZ       LABEL_FE_ENABLE
                        1292
```

```
LOC  OBJ               LINE     SOURCE

                       1293                          ;NOW SFEEN_ IS FALSE(SIGNAL IS HIGH), TAKE DATA OFF BUS
00B1 B89B9B            1294                          MOV    AX, D_DATA_BUS_IN
00B4 E74E              1295                          OUT    DECODER_BUS_SPVR, AX
                       1296
                       1297 +1
                       1298 +1
                       1299 +1
                       1300 +1
00B6 FA                1301 +1                       CLI
00B7 B8E000            1302 +1                       MOV    AX, ENABLE_NORMAL_MASK
00BA E67A              1303 +1                       OUT    PIC_PORT_1, AL
00BC FB                1304 +1                       STI
                       1305
00BD 58                1306                          POP    AX
                       1307
00BE CF                1308                          IRET
                       1309
                       1310     FE_OUTPUT_INTERRUPT  ENDP
                       1311
                       1312
                       1313
                       1314 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          1315
00BF                      1316     FE_INPUT_INTERRUPT      PROC    FAR
                          1317
                          1318     ;*************************************************************************
                          1319     ;* THIS INTERRUPT ROUTINE IS ACTIVATED BY THE FE ENABLE_ SIGNAL ON THE     *
                          1320     ;* CODER BUS. IT IS RESPONSIBLE FOR TAKING THE FEX(FEH AND FEL) DATA FROM  *
                          1321     ;* THE CODER DATA BUS AND STORE THEM INTO THE FE BUFFER.                   *
                          1322     ;* AFTER ENTERING, THE ROUTINE PUSHES AX REGISTER ONTO STACK, AND CHECK    *
                          1323     ;* FOR FE CLOCK. ONCE FE CLOCK IS VALID, THE FEX DATA IS TAKEN FROM THE    *
                          1324     ;* AND STORED IN THE FE_BUFFER. THE ROUTINE THEN INCREMENTS THE BUFFER     *
                          1325     ;* POINTER AND EXIT.                                                       *
                          1326     ;*                                                                         *
                          1327     ;*               REGISTER USAGES:                                          *
                          1328     ;*                        AX,SI                                            *
                          1329     ;*                                                                         *
                          1330     ;*************************************************************************
                          1331
                          1332
                          1333                            ;THIS ROUTINE IS ACTIVATED BY FE_ENABLE SIGNAL ON THE
                          1334                            ;CODER CONTROL BUS
                          1335
                          1336
                          1337
00BF 50                   1338                            PUSH    AX
                          1339 +1
                          1340 +1
                          1341 +1
                          1342 +1
00C0 B8FD00               1343 +1                         MOV     AX,ENABLE_SOF_MASK
00C3 E67A                 1344 +1                         OUT     PIC_PORT_1, AL
00C5 FB                   1345 +1                         STI
                          1346
00C6                      1347     LABEL_FE_CLK:          ;CHECK FOR FE CLOCK VALIDITY
                          1348
00C6 E544                 1349                            IN      AX, C_CONTROL_BUS
00C8 250008               1350                            AND     AX, CFECLK
00CB 74F9                 1351                            JZ      LABEL_FE_CLK
                          1352                            ;FE CLOCK IS TRUE(I.E. SIGNAL IS HIGH), READ IN FEX FROM DATA
                                    BUS
                          1353
00CD E542                 1354                            IN      AX, C_DATA_BUS
                          1355
                          1356
00CF 894430               1357                            MOV     FE_BUFFER[SI], AX
                          1358
                          1359                            ;SI IS THE OFFSET POINTER FOR DEPOSITION
                          1360
                          1361                            ;INCREMENT POINTERS
                          1362
                          1363
00D2 83C602               1364                            ADD     SI, TYPE FE_BUFFER
00D5 FF066200             1365                            INC     FE_SERVICE_COUNTER      ;INCREMENT THE SERVICE
                          1366                                                            ;COUNTER TO ENABLE NON-INTERR
                                    UPT
                          1367                                                            ;DRIVEN QUANTIZATION PROCESS
```

```
LOC  OBJ               LINE     SOURCE

                       1368
00D9 83FE40            1369                           CMP    SI, LEN_FE_BUFFER * TYPE FE_BUFFER        ;END OF BUFFE
                                R?
00DC 7203              1370                           JB     LABEL_FE_INPUT_END
00DE BE0000            1371                           MOV    SI, 0                      ;RESET POINTER TO BEGINNING
                       1372                                                             ;OF CIRCULAR BUFFER.
                       1373
00E1                   1374 +1  LABEL_FE_INPUT_END:
                       1375 +1
                       1376 +1
                       1377 +1
00E1 FA                1378 +1                        CLI
00E2 B8E000            1379 +1                        MOV    AX, ENABLE_NORMAL_MASK
00E5 E67A              1380 +1                        OUT    PIC_PORT_1, AL
00E7 FB                1381 +1                        STI
00E8 58                1382                           POP    AX
                       1383
00E9 CF                1384                           IRET
                       1385
                       1386     FE_INPUT_INTERRUPT    ENDP
                       1387
                       1388
                       1389 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                        1390
                        1391
                        1392
00EA                    1393     PREDICTOR_INPUT_INTERRUPT       PROC    FAR
                        1394
                        1395
                        1396
                        1397     ;***************************************************************************
                        1398     ;* THIS PROCEDURE IS ACTIVATED BY THE PEN_  SIGNAL ON THE CODER BUS.       *
                        1399     ;* IT INPUTS THE PREDICTOR COEFFICIENTS FROM THE CODER BUS AND STORES THE  *
                        1400     ;* DATA IN THE PREDICTOR_COEFF BUFFER.                                      *
                        1401     ;*                                                                          *
                        1402     ;*                REGISTER USAGES:                                          *
                        1403     ;*                        AX, DI                                            *
                        1404     ;*                                                                          *
                        1405     ;***************************************************************************
00EA 50                 1406                             PUSH    AX
                        1407
                        1408 +1
                        1409 +1
                        1410 +1
                        1411 +1
00EB B8FE00             1412 +1                          MOV     AX,ENABLE_SOF_MASK
00EE E67A               1413 +1                          OUT     PIC_PORT_1, AL
00F0 FB                 1414 +1                          STI
                        1415
                        1416                             ;TEST FOR P CLOCK VALID
                        1417
                        1418
00F1 ED44               1419     LABEL_P_CLK:             IN      AX, C_CONTROL_BUS
                        1420
00F3 252000             1421                             AND     AX, CPCLK
00F6 74F9               1422                             JZ      LABEL_P_CLK
                        1423
                        1424
                        1425                             ;P CLOCK IS VALID, READ IN PREDICTOR COEFFICIENTS FROM
                        1426                             ;DATA BUS
                        1427
                        1428
00F8 E542               1429                             IN      AX, C_DATA_BUS
                        1430
                        1431                             ;BP SPECIFIES THE FRAME STARTING OFFSET(0, 16, 32 OR 48)
                        1432                             ;DI SPECIFIES THE OFFSET WITHIN EACH FRAME (0, 2, 4...14)
00FA 3E8903             1433                             MOV     DS:PREDICTOR_BUFFER[BP][DI], AX
                        1434
00FD 83C702             1435                             ADD     DI, TYPE PREDICTOR_BUFFER       ;INCREMENT OFFSET
                        1436                                                                     ;VALUE WITHIN ONE FRA
                                 ME
                        1437
                        1438
                        1439 +1
                        1440 +1
                        1441 +1
                        1442 +1
0100 FA                 1443 +1                          CLI
```

```
LOC  OBJ                  LINE     SOURCE

0101 B8E000               1444 +1                          MOV     AX, ENABLE_NORMAL_MASK
0104 E676                 1445 +1                          OUT     PIC_PORT_1, AL
0106 FB                   1446 +1                          STI
                          1447
0107 58                   1448                             POP     AX
0108 CF                   1449                             IRET
                          1450
                          1451     PREDICTOR_INPUT_INTERRUPT       ENDP
                          1452
                          1453
                          1454
                          1455 +1  $EJECT
```

```
LOC  OBJ              LINE     SOURCE

                      1456
0109                  1457     EXCEPTION_INT           PROC    FAR
                      1458
                      1459                  ;************************************************************
                      1460                  ;* THIS IS THE EXCEPTION HANDLER FOR THE 8086 CPU.          *
                      1461                  ;* UNDER NORMAL OPERATION ENVIRONMENT, THIS CODE SHOULD *
                      1462                  ;* NEVER BE EXECUTED.                                       *
                      1463                  ;* IN THE UNLIKELY EVENT THAT THIS HANDLER IS ACTIVATED,*
                      1464                  ;* IT SETS THE LED_DISPLAY_VALUE TO OFFH AND OUTPUT         *
                      1465                  ;* TO THE ERROR_LED.                                        *
                      1466                  ;*              REGISTER USAGES:                            *
                      1467                  ;*                      AX,                                 *
                      1468                  ;*                                                          *
                      1469                  ;************************************************************
                      1470
                      1471
                      1472                  ;EXTERNAL INTERRUPTS ARE DISABLED DURING EXECUTION OF THIS HANDLER
                      1473
                      1474
0109 50               1475                             PUSH    AX
010A B8FFFF           1476                             MOV     AX, OFFFFH
010D A30000           1477                             MOV     LED_DISPLAY_VALUE, AX
0110 E654             1478                             OUT     ERROR_LED, AL
0112 58               1479                             POP     AX
0113 FB               1480                             STI
0114 CF               1481                             IRET
                      1482
                      1483
                      1484     EXCEPTION_INT           ENDP
                      1485
                      1486
                      1487
----                  1488     INTERRUPT_CODE          ENDS
                      1489                             END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

B2. 5-BIT ADAPTIVE RESIDUAL QUANTIZER

```
VAX/VMS 8086/8087/8088 MACRO ASSEMBLER V1.0VX ASSEMBLY OF MODULE M5LDEM
OBJECT MODULE PLACED IN M5LDEM.OBJ
NO INVOCATION LINE CONTROLS


LOC  OBJ                  LINE     SOURCE

                             1
                             2
                             3                      NAME    M5LDEM
                             4
                             5
                             6
                             7         ;*******************************************************************
                             8         ;*                                                                 *
                             9         ;* PROJECT:        498A RELP CODEC                                 *
                            10         ;* BOARD:          MULDEM SIMULATOR                                *
                            11         ;* DEVICE:         2 FIRMWARE EPROMS (2732A-2)                     *
                            12         ;* BOARD LOCATION: E15, E29                                        *
                            13         ;* PROGRAMMABLE                                                    *
                            14         ;* DEVICE NUMBER:  60-0670, 60-0671                                *
                            15         ;*                                                                 *
                            16         ;*******************************************************************
                            17
                            18 +1  $EJECT
```

```
LOC  OBJ             LINE     SOURCE

                       19
                       20
                       21
                       22
                       23             ;*****************************************************
                       24             ;*                                                   *
                       25             ;* THIS FIRMWARE SIMULATES THE MULtiplexor/DEMulti-  *
                       26             ;* plexor IN THE RELP CODEC SYSTEM. REFER TO MDA     *
                       27             ;* DOCUMENT 00 - 3035 - R01 FOR DETAIL.              *
                       28             ;*                                                   *
                       29             ;*****************************************************
                       30
                       31
                       32
                       33
                       34     ;************************
                       35     ;*  SYSTEM CONSTANTS   *
                       36     ;************************
                       37
                       38
                       39
 0020                  40     LEN_FE_BUFFER           EQU     32     ;ACCOMODATE 16 PAIRS OF FEX
                       41
 0008                  42     LEN_P_BUFFER_FRAME      EQU     8      ;EACH FRAME ACCOMODATES 8 PREDICTOR COEFFICIE
                              NTS
                       43
 0020                  44     LAST_P_BUFFER_FRAME     EQU     32     ;THE BASE OFFSET ADDRESS OF LAST P FRAME
                       45                                            ;FROM THE BEGINNING OF P BUFFER. THIS VALUE
                       46                                            ;MUST BE IN MULTIPLE OF LEN_P_BUFFER_FRAME *
                              TYPE P_BUFFER
                       47                                            ;THE FRAME DELAY IS EQUAL TO 1 +
                       48                                            ;(LAST_P_BUFFER_FRAME / 16)
                       49
                       50
 0001                  51     FE_HIGH                 EQU     1      ;THIS REPRESENTS THE FEH'S TYPE
 0000                  52     FE_LOW                  EQU     0      ;THIS REPRESENTS THE FEL'S TYPE
                       53
 0010                  54     NLEV                    EQU     16     ;NUMBER OF QUANTIZER REGIONS
                       55
 0050                  56     FSVMN                   EQU     80     ;FSVMN  - MINIMUM SCALING FACTOR
                       57
 1F40                  58     FSVMX                   EQU     8000   ;FSVMX  - MAXIMUM SCALING FACTOR
                       59
 0078                  60     FSVTH                   EQU     120    ;FSVTH  - THRESHOLD VALUE. IF THE
                       61                                            ;SCALE FACTOR IS LESS THAN FSVTH, A
                       62                                            ;MID-TREAD QUANTIZER IS USED INSTEAD
                       63                                            ;OF A MID-RISE QUANTIZER. THE MID-TREAD
                       64                                            ;QUANTIZER USES ZERO AS AN OUTPUT LEVEL
                       65                                            ;INSTEAD OF YQ(1).
                       66
 0000                  67     POSITIVE                EQU     0      ;POSITIVE SIGN
                       68
 0001                  69     NEGATIVE                EQU     1      ;NEGATIVE SIGN
                       70
                       71
```

```
LOC  OBJ          LINE     SOURCE

                           ;     HARDWARE DEPENDENT CONSTANTS

0042                       C_DATA_BUS            EQU      42H     ;THIS IS THE PORT(B) ADDRESS ON THE 8255'S
                                                                  ;FOR THE 16 BIT CODER DATA BUS

0044                       C_CONTROL_BUS         EQU      44H     ;THIS IS THE PORT(C) ADDRESS ON THE 8255'S
                                                                  ;FOR THE 16 BIT CODER CONTROL BUS

0046                       CODER_BUS_SPVR        EQU      46H     ;THIS IS THE SUPERVISOR PORT(CONTROL PORT)
                                                                  ;FOR THE CODER BUS(THE CONTROL BUS AND
                                                                  ;THE DATA BUS)



004A                       D_DATA_BUS            EQU      4AH     ;THIS IS THE PORT(B) ADDRESS ON THE 8255'S
                                                                  ;FOR THE 16 BIT DECODER DATA BUS


004C                       D_CONTROL_BUS         EQU      4CH     ;THIS IS THE PORT(C) ADDRESS ON THE 8255'S
                                                                  ;FOR THE 16 BIT DECODER CONTROL BUS


004E                       DECODER_BUS_SPVR      EQU      4EH     ;THIS IS THE SUPERVISOR PORT(CONTROL PORT)
                                                                  ;FOR THE DECODER BUS(THE CONTROL BUS AND
                                                                  ;THE DATA BUS)






0064                       ERROR_LED             EQU      64H     ;

0066                       ERROR_BUS_SPVR        EQU      66H     ;


                                ;8259A PIC DEPENDENT CONSTANTS

0078                       PIC_PORT_0            EQU      78H     ;8259A PROGRAMMABLE INTERRUPT CONTROLLER
007A                       PIC_PORT_1            EQU      7AH     ;CONTROL PORTS.

0013                       ICW1                  EQU      13H     ;SINGLE PIC, ICW4 NEEDED
0008                       ICW2                  EQU      08H     ;STARTING INTERRUPT VECTOR = 8
0003                       ICW4                  EQU      03H     ;SFNM = 0, AEOI = 1, NON-BUFFERED MODE
                                                                  ;SP/EN WILL HAVE INPUT = 1 =>MASTER
                                                                  ;UPM = 1 =>8086/88 SYSTEM.
00FE                       ENABLE_SOF_MASK       EQU      0FEH    ;MASK OUT ALL EXCEPT
                                                                  ;SOF INTERRUPT(R0)

00E0                       ENABLE_NORMAL_MASK    EQU      0E0H    ;MASK OUT UNUSED
                                                                  ;INTERRUPT(R7,R6,R5)



                                ;8255A PPI DEPENTENT CONTSTANTS
```

```
LOC  OBJ                LINE     SOURCE

                         127
9B9B                     128     C_DATA_BUS_IN         EQU     9B9BH    ;PROGRAM THE CODER DATA BUS TO BE INPUT, ALL
                         129                                            ;OTHER PORTS ON CODER BUS ARE INPUT.
                         130                                            ;MSB AND LSB OF THIS WORD EACH ADDRESSES TO O
                                 NE PPI
                         131
9999                     132     C_DATA_BUS_OUT        EQU     9999H    ;PROGRAM THE CODER DATA BUS TO BE OUTPUT(IE.
                         133                                            ;TAKE CONTROL OF THE DATA BUS), ALL OTHER
                         134                                            ;PORTS ON THE CODER BUS ARE INPUT. MSB
                         135                                            ;AND LSB OF THIS WORD EACH ADDRESSES ONE PPI.
                         136
9B9B                     137     D_DATA_BUS_IN         EQU     9B9BH    ;PROGRAM THE DECODER DATA BUS TO BE INPUT, AL
                                 L
                         138                                            ;OTHER PORTS ON DECODER BUS ARE INPUT.
                         139                                            ;MSB AND LSB OF THIS WORD EACH ADDRESSES TO O
                                 NE PPI
                         140
9999                     141     D_DATA_BUS_OUT        EQU     9999H    ;PROGRAM THE DECODER DATA BUS TO BE OUTPUT(IE
                                 .
                         142                                            ;TAKE CONTROL OF THE DATA BUS), ALL OTHER
                         143                                            ;PORTS ON THE DECODER BUS ARE INPUT. MSB
                         144                                            ;AND LSB OF THIS WORD EACH ADDRESSES ONE PPI.
                         145
0092                     146     ERROR_LED_ON          EQU     0092H    ;PROGRAM THE ERROR BUS TO BE:
                         147                                            ;  PORT A - INPUT } NOT USED HERE
                         148                                            ;  PORT B - INPUT } NOT USED HERE
                         149                                            ;  PORT C - OUTPUT } TO ERROR_LED
                         150
009B                     151     ERROR_LED_OFF         EQU     009BH    ;PROGRAM THE ERROR BUS TO BE:
                         152                                            ;  PORT A - INPUT } NOT USED HERE
                         153                                            ;  PORT B - INPUT } NOT USED HERE
                         154                                            ;  PORT C - INPUT } TO ERROR_LED
                         155                                            ;THE LED DISPLAY IN THIS STATE WILL
                         156                                            ;BE 'FF'
                         157
                         158
                         159           ;CONTROL BUS DEPENDENT CONSTANTS
                         160
                         161           ;DECODER BUS
                         162           ;-----------
                         163
0004                     164     SPEN                  EQU     0004H    ;MASK FOR SPEN SIGNAL(LOW TRUE)
0010                     165     SFEEN                 EQU     0010H    ;MASK FOR SFEEN SIGNAL(LOW TRUE)
                         166
                         167
                         168           ;CODER BUS
                         169           ;---------
                         170
0800                     171     CFECLK                EQU     0800H    ;MASK FOR CFECLK SIGNAL(HIGH TRUE)
0020                     172     CPCLK                 EQU     0020H    ;MASK FOR CPCLK SIGNAL(HIGH TRUE)
                         173
                         174
                         175
                         176     ;MDA IN HOUSE MONITOR ENTRY POINT
                         177
```

```
LOC  OBJ                LINE     SOURCE

 0010                    178     MONT_86_IP              EQU     0010H   ;OFFSET ADDRESS
 FE9F                    179     MONT_86_CS              EQU     0FE9FH  ;SEGMENT ADDRESS
                         180
                         181 +1  $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          182
                          183
----                      184     INTERRUPT_VECTOR        SEGMENT         WORD AT OH
                          185
                          186
                          187
                          188     ;                 *****************************
                          189     ;                 *  INTERRUPT VECTOR TABLE   *
                          190     ;                 *****************************
                          191
                          192     ;INTEL RESERVES INT 5 TO 32 FOR INTERNAL USES. CURRENT IMPLEMENTATION
                          193     ;                    VIOLATES THIS RESTRICTION.
                          194
                          195
                          196     ;8086 PREDIFINED INTERRUPTS: (INT 0 TO 4)
                          197
0000 (1                   198         DIVIDE_INT_IP      DW  1 DUP(?)
     ????
     )
0002 (1                   199         DIVIDE_INT_CS      DW  1 DUP(?)
     ????
     )
0004 (1                   200         SINGLE_STEP_IP     DW  1 DUP(?)
     ????
     )
0006 (1                   201         SINGLE_STEP_CS     DW  1 DUP(?)
     ????
     )
0008 (1                   202         NMI_IP             DW  1 DUP(?)
     ????
     )
000A (1                   203         NMI_CS             DW  1 DUP(?)
     ????
     )
000C (1                   204         BREAKPOINT_IP      DW  1 DUP(?)
     ????
     )
000E (1                   205         BREAKPOINT_CS      DW  1 DUP(?)
     ????
     )
0010 (1                   206         OVERFLOW_IP        DW  1 DUP(?)
     ????
     )
0012 (1                   207         OVERFLOW_CS        DW  1 DUP(?)
     ????
     )
                          208
                          209
                          210     ;MULDEM APPLICATION INTERRUPTS: (INT 8 TO 15)
                          211
0020                      212          ORG         20H
0020 (1                   213          INT_8_IP        DW  1 DUP(?)
     ????
     )
0022 (1                   214          INT_8_CS        DW  1 DUP(?)
LOC  OBJ                  LINE     SOURCE

     ????
     )
0024 (1                   215          INT_9_IP       DW  1 DUP(?)
     ????
     )
0026 (1                   216          INT_9_CS       DW  1 DUP(?)
     ????
     )
0028 (1                   217          INT_10_IP      DW  1 DUP(?)
     ????
     )
002A (1                   218          INT_10_CS      DW  1 DUP(?)
     ????
     )
002C (1                   219          INT_11_IP      DW  1 DUP(?)
     ????
     )
002E (1                   220          INT_11_CS      DW  1 DUP(?)
     ????
     )
0030 (1                   221          INT_12_IP      DW  1 DUP(?)
     ????
     )
0032 (1                   222          INT_12_CS      DW  1 DUP(?)
     ????
     )
0034 (1                   223          INT_13_IP      DW  1 DUP(?)
     ????
     )
0036 (1                   224          INT_13_CS      DW  1 DUP(?)
     ????
     )
0038 (1                   225          INT_14_IP      DW  1 DUP(?)
     ????
     )
003A (1                   226          INT_14_CS      DW  1 DUP(?)
     ????
     )
003C (1                   227          INT_15_IP      DW  1 DUP(?)
     ????
     )
003E (1                   228          INT_15_CS      DW  1 DUP(?)
     ????
     )
                          229
                          230
----                      231     INTERRUPT_VECTOR          ENDS
                          232
                          233
                          234
                          235 +1  $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          236
                          237
----                      238     DATA    SEGMENT
                          239
                          240          ;THIS SEGMENT WILL RESIDE IN RAM
                          241
0000 (24                  242          PREDICTOR_BUFFER   DW      (((LAST_P_BUFFER_FRAME/16)+1) * LEN_P_BUFFER_FRAME) D
                                  UP(?)
     ????
     )
0030 (32                  243          FE_BUFFER          DW      (LEN_FE_BUFFER) DUP(?)
     ????
     )
                          244
0070 (1                   245          LED_DISPLAY_VALUE  DW      1 DUP(?)
     ????
     )
                          246
0072 (1                   247          ABS_XIN            DW      1 DUP(?)
     ????
     )
0074 (1                   248          XOUT               DW      1 DUP(?)
     ????
     )
                          249
0076 (1                   250          FSV                DW      1 DUP(?)
     ????
     )
0078 (1                   251          FSVL               DW      1 DUP(?)
     ????
     )
007A (1                   252          FSVH               DW      1 DUP(?)
     ????
     )
                          253
007C (1                   254          I                  DW      1 DUP(?)
     ????
     )
007E (1                   255          IL                 DW      1 DUP(?)
     ????
     )
0080 (1                   256          L2                 DW      1 DUP(?)
     ????
     )
                          257
0082 (1                   258          FE_SERVICE_COUNTER DW      1 DUP(?)
     ????
     )
                          259
0084 (1                   260          FE_SERVICE_PTR     DW      1 DUP(?)
     ????
     )
                          261
0086 (1                   262          FE_TYPE            DW      1 DUP(?)
     ????
LOC  OBJ                  LINE     SOURCE

     )
                           263
0038 (1                    264          SIGN_FLAG           DW      1 DUP(?)
     ????
     )
                           265
                           266
----                       267     DATA    ENDS
                           268
                           269
                           270
                           271
                           272
                           273
                           274
                           275
                           276
----                       277     STACK                    SEGMENT
                           278
                           279             ;*********************************************
                           280             ;* THE STACK IS SOLELY USED BY THE 8086 TO   *
                           281             ;* STORE RETURN ADDRESS IN INTERRUPT ROUTINES*
                           282             ;*********************************************
0000 (60                   283                              DW      60 DUP (?)
     ????
     )
0078                       284     TOS                      LABEL   WORD
                           285
----                       286     STACK                    ENDS
                           287
                           288 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           289
                           290
                           291
                           292     ;SYSTEM MACROS:
                           293     ;--------------
                           294
                           295
                           296     ;THE MACROS ARE NOT LISTED IN THE ASSEMBLER GENERATED LISTING, REFER
                           297     ;TO THE SOURCE FILE FOR MACRO CONTENTS.
                           298
                           299
                           300
                           301
                           302
                           303
                           304
                           305
                           306
                           307
                           308
                           309
                           310
                           311
                           312
                           313
                           314
                           315 +1  $EJECT
```

```



            ;*******************************************************
            ;*                                                     *
            ;*   REGISTER USAGE IN M5LDEM SIMULATOR SYSTEM :       *
            ;*                                                     *
            ;*******************************************************


;DEDICATED USAGE FOR ALL PARTS OF THE SYSTEM AT ALL TIME:

            ;SI     - FE_BUFFER INPUT POINTER
            ;BX     - FE_BUFFER OUTPUT POINTER [THIS REGISTER IS ALSO USED UNDER
            ;         NON-INTERRUPT DRIVEN QUANTIZATION PROCESS, HOWEVER, THE
            ;         ORIGINAL REGISTER IS PRESERVED]
            ;BP     - PREDICTOR_BUFFER FRAME POINTER
            ;DI     - PREDICTOR_BUFFER OFFSET POINTER


            ;CS     - CODE SEGMENT
            ;DS     - DATA SEGMENT
            ;ES     - DATA SEGMENT OR INTERRUPT_VECTOR SEGMENT
            ;SS/SP  - STACK OPERATION


;UNASSIGNED REGISTERS:

            ;THESE REGISTERS DO NOT CARRY DEDICATED FUNCTIONS IN THE M5LDEM
            ;SIMULATOR:

            ;AX, CX, DX

+1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         350
                         351
                         352           PUBLIC          START_ADDR
                         353
                         354
----                     355     CODE                  SEGMENT
                         356
                         357           ASSUME          CS:CODE, DS:DATA, SS:STACK, ES:INTERRUPT_VECTOR
                         358
                         359
                         360
                         361           ;************************************
                         362           ;*    STATIC    VARIABLES          *
                         363           ;************************************
                         364
                         365
                         366                      ;YQ  -   ARRAY OF NLEV NORMALIZED QUANTIZER OUTPUT VALUES
                         367                      ;        (IN INCREASING ORDER)
                         368                      ;        VALUE SCALED BY ** IN ** REPRESENTATION
0000 0000                369           YQ              DW      0, 1024, 3072, 5120, 7168, 9216, 11264
0002 0004
0004 000C
0006 0014
0008 001C
000A 0024
000C 002C
000E 0034                370                           DW      13312, 15360, 17408, 19456, 21504, 23552
0010 003C
0012 0044
0014 004C
0016 0054
0018 005C
001A 0064                371                           DW      25600, 27648, 29696, 31744
001C 006C
001E 0074
0020 007C
                         372
                         373                      ;XQ  -   ARRAY OF NLEV-1 NORMALIZED QUANTIZER BREAK POINTS
                         374                      ;        (IN INCREASING ORDER)
                         375                      ;        VALUE SCALED BY ** IN ** REPRESENTATION
0022 0000                376           XQ              DW      0, 2048, 4096, 6144, 8192, 10240, 12288
0024 0008
0026 0010
0028 0018
002A 0020
002C 0028
002E 0030
0030 0038                377                           DW      14336, 16384, 18432, 20480, 22528, 24576
0032 0040
0034 0048
0036 0050
0038 0058
003A 0060
003C 0068                378                           DW      26624, 28672, 30720
003E 0070
```

```
LOC  OBJ                  LINE     SOURCE

0040 0076
                           379
                           380                   ;QMLT   - ARRAY OF NLEV, QUANTIZER MULTIPLIERS
                           381                   ;         VALUE SCALED BY ** IN ** REPRESENTATION
0042 0000                  382           QMLT            DW      0, 13926, 13926, 13926, 13926, 13926, 13926
0044 6636
0046 6636
0048 6636
004A 6636
004C 6636
004E 6636
0050 6636                  383                           DW      13926, 13926, 19661, 22938, 26214, 29491
0052 6636
0054 CD4C
0056 9A59
0058 6666
005A 3373
005C 0080                  384                           DW      32768, 36045, 39322, 42598
005E CD8C
0060 9A99
0062 66A6
                           385
                           386
                           387 +1  $EJECT
```

```
LOC  OBJ                   SOURCE


                      ;    ************************************************************************
                      ;    *                  M A I N     P R O G R A M                          *
                      ;    ************************************************************************


                      ;         ********************************************************
                      ;         *                                                      *
                      ;         *         REGISTER VALUES ARE NOT PRESERVED            *
                      ;         *         IN ALL PROCEDURES. THEY SHOULD BE            *
                      ;         *              SAVED BEFORE ENTERING.                  *
                      ;         *                                                      *
                      ;         ********************************************************


0064 FA                    START_ADDR:   CLI                        ;DISABLE EXTERNAL INTERRUPT
0065 B8----         R                    MOV    AX, DATA            ;CANNOT MOVE IMMED. VALUE TO
0068 8ED8                                MOV    DS, AX              ;SEGMENT REGISTER.

006A 8EC0                                MOV    ES, AX              ;ES AND DS ARE REFERING TO SAME
                                                                    ;SEGMENT

                                         ;RESET THE LED_DISPLAY_VALUE
006C C70667000000                        MOV    LED_DISPLAY_VALUE, 0

0072 B89200                              MOV    AX, ERROR_LED_ON
0075 E666                                OUT    ERROR_BUS_SPVR, AL

                                         ;OUTPUT ERROR DISPLAY VALUE

0077 B80000                              MOV    AX, 0
007A E664                                OUT    ERROR_LED, AL

                      ;         ************************************************
                      ;         * INITIALIZE THE INTERRUPT VECTOR  TABLE       *
                      ;         *                                              *
                      ;         * THIS SUBSYSTEM USES FOLLOWING SIGNALS FROM *
                      ;         * THE SYSTEM BUS:                              *
                      ;         *                                              *
                      ;         *      - START OF FRAME (SOF_)                 *
                      ;         *      - FE ENABLE (SFEEN_)                    *
                      ;         *      - P ENABLE (SPEN_)                      *
                      ;         *      - FE CLOCK (CFECLK)                     *
                      ;         *      - P CLOCK (CPCLK)                       *
                      ;         ************************************************




007C B80000                ERROR_ENTRY:  MOV    AX, INTERRUPT_VECTOR
007F 8EC0                                MOV    ES, AX                      ;USE EXTRA SEGMENT TO ADDRESS
                                                                            ;THE INTERRUPT VECTOR TABLE

0081 26C70608001000                      MOV    NMI_IP, MONT_86_IP
0088 26C7060A009FFE                      MOV    NMI_CS, MONT_86_CS
```

```
LOC  OBJ                  LINE     SOURCE

                           443
008F B80901                444                    MOV     AX, OFFSET EXCEPTION_INT
0092 BB----90        R     445                    MOV     BX, SEG EXCEPTION_INT
0096 26A30000              446                    MOV     DIVIDE_INT_IP, AX
009A 26891E0200            447                    MOV     DIVIDE_INT_CS, BX
009F 26A31000              448                    MOV     OVERFLOW_IP, AX
00A3 26891E1200            449                    MOV     OVERFLOW_CS, BX
                           450
                           451
00A8 26C70620000000        452                    MOV     INT_8_IP, OFFSET SOF_INTERRUPT
00AF 26C7062200---- R      453                    MOV     INT_8_CS, SEG SOF_INTERRUPT
                           454
00B6 26C70624005D00        455                    MOV     INT_9_IP, OFFSET PREDICTOR_OUTPUT_INTERRUPT
00BD 26C7062600---- R      456                    MOV     INT_9_CS, SEG PREDICTOR_OUTPUT_INTERRUPT
                           457
00C4 26C7062800BE00        458                    MOV     INT_10_IP, OFFSET FE_OUTPUT_INTERRUPT
00CB 26C7062A00---- R      459                    MOV     INT_10_CS, SEG FE_OUTPUT_INTERRUPT
                           460
00D2 26C7062C00E400        461                    MOV     INT_11_IP, OFFSET PREDICTOR_INPUT_INTERRUPT
00D9 26C7062E00---- R      462                    MOV     INT_11_CS, SEG PREDICTOR_INPUT_INTERRUPT
                           463
00E0 26C70630008F00        464                    MOV     INT_12_IP, OFFSET FE_INPUT_INTERRUPT
00E7 26C7063200---- R      465                    MOV     INT_12_CS, SEG FE_INPUT_INTERRUPT
                           466 +1  $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          467
                          468
                          469           ASSUME ES:DATA
                          470
                          471
                          472
                          473           ;INITIALIZE 8086 PROCESSOR ENVIRONMENT
                          474
                          475
00EE B8----          R    476                   MOV     AX, DATA        ;CANNOT MOVE IMMED. VALUE TO
00F1 8ED8                 477                   MOV     DS, AX          ;SEGMENT REGISTER.
                          478
00F3 8EC0                 479                   MOV     ES, AX          ;ES AND DS ARE REFERING TO SAME
                          480                                           ;SEGMENT
00F5 B8----          R    481                   MOV     AX, STACK
00F8 8ED0                 482                   MOV     SS, AX
00FA BC7800               483                   MOV     SP, OFFSET TOS
                          484
                          485
                          486           ;INITIALIZE ALL SYSTEM HARDWARE
                          487
                          488
                          489
                          490           ;       ******************
                          491           ;       *  P I C         *
                          492           ;       ******************
                          493
                          494
                          495           ;CAUTION:  AUTOMATIC EOI IS USED HERE. REFER TO INTEL APPLICATION
                          496           ;          NOTE AP-59 'USING THE 8259A PROGRAMMABLE INTERRUPT
                          497           ;          CONTROLLER', UNDER HEADING 'AUTOMATIC EOI MODE'
                          498
                          499
00FD B013                 500                   MOV     AL, ICW1
00FF E67E                 501                   OUT     PIC_PORT_0, AL
                          502
0101 BA7A00               503                   MOV     DX, PIC_PORT_1
0104 B008                 504                   MOV     AL, ICW2
0106 EE                   505                   OUT     DX, AL
                          506
                          507                   ;ICW 3 IS NOT NEEDED FOR CURRENT HARDWARE CONFIGURATION
                          508
0107 B003                 509                   MOV     AL, ICW4
0109 EE                   510                   OUT     DX, AL
                          511
010A B0FE                 512                   MOV     AL, ENABLE_SOF_MASK
010C EE                   513                   OUT     DX, AL
                          514
                          515
                          516           ;       ******************
                          517           ;       *  P P I         *
                          518           ;       ******************
                          519
                          520
                          521           ;INITIALLY ALL PORT ARE PROGRAMMED TO BE INPUT PORTS IN MODE 0
```

```
LOC  OBJ                LINE     SOURCE

                         522              ;EXCEPT THE ERROR LED PORT, WHICH WILL BE OUTPUT ALL THE TIME
                         523
010D B87B9B              524                      MOV     AX, C_DATA_BUS_IN
0110 E746                525                      OUT     CODER_BUS_SPVR, AX
                         526
                         527
0112 B89B9B              528                      MOV     AX, D_DATA_BUS_IN
0115 E74E                529                      OUT     DECODER_BUS_SPVR, AX
                         530 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         531
                         532           ;***********************************************
                         533           ;*  INITIALIZE APPLICATION PROGRAM ENVIRONMENT *
                         534           ;***********************************************
                         535
                         536
                         537
                         538
                         539
                         540                  ; FILL BUFFER AREAS WITH ZEROS
                         541
0117 B80000              542                  MOV     AX, 0                          ;FILL VALUE
011A B91800              543                  MOV     CX, LENGTH PREDICTOR_BUFFER    ;ITERATION COUNT
011D BF0000              544                  MOV     DI, OFFSET PREDICTOR_BUFFER
                         545
                         546                  ;ES SETS TO THE SEGMENT BASE OF DATA SEGMENT
                         547
0120 F3                  548           REP    STOS    PREDICTOR_BUFFER
0121 AB
                         549
                         550
0122 B92000              551                  MOV     CX, LENGTH FE_BUFFER           ;ITERATION COUNT
0125 BF3000              552                  MOV     DI, OFFSET FE_BUFFER
                         553
0128 F3                  554           REP    STOS    FE_BUFFER
0129 AB
                         555
                         556
                         557                  ;RESET SERVICE COUNTER
012A C70682000000        558                  MOV     FE_SERVICE_COUNTER, 0
                         559
                         560                  ;INITIALIZE FE TYPE TO LOW
0130 C70686000000        561                  MOV     FE_TYPE, FE_LOW
                         562
                         563
                         564                  ;INITIALIZE BUFFER POINTERS TO APPROPRIATE VALUES
                         565
                         566                  ;PREDICTOR BUFFER , BP IS THE FRAME POINTER,
                         567                  ;DI IS THE OFFSET POINTER WITHIN A FRAME.
                         568
0136 BD2000              569                  MOV     BP, LAST_P_BUFFER_FRAME        ;ON FIRST SOF INTERRUPT AFTER
                         570                                                         ;POWER UP, SOF_INTERRUPT ROUT
                                 INE
                         571                                                         ;WILL SET BP = 0, DI = 0
                         572
0139 BF1000              573                  MOV     DI, LEN_P_BUFFER_FRAME * (TYPE PREDICTOR_BUFFER)
                         574
                         575                  ;FE BUFFER INITIALIZATION
                         576
013C BE1000              577                  MOV     SI, 16                         ;POINTS TO FEL(4)
013F BB0000              578                  MOV     BX, 0
                         579
                         580                  ;QUANTIZATION SERVICE
                         581
0142 C70684001000        582                  MOV     FE_SERVICE_PTR, 16
```

```
LOC  OBJ                  LINE     SOURCE

0148 C70678005000          583                     MOV     FSVL, FSVMN
014E C7067A005000          584                     MOV     FSVH, FSVMN
                           585
                           586
                           587
0154 FB                    588                     STI                                ;ENABLE EXTERNAL INTERRUPT
                           589
0155 F4                    590                     HLT                                ;WAIT UNTIL THE FIRST
                           591                                                        ;START OF FRAME INTERRUPT
                           592
                           593 +1  $EJECT
```

```
LOC  OBJ              LINE    SOURCE

                       594
                       595
                       596            ;*********************************************
                       597            ;*                                           *
                       598            ;*    START   OF   APPLICATION   PROGRAM      *
                       599            ;*                                           *
                       600            ;* AFTER POWER UP, THE FOLLOWING APPLICATION *
                       601            ;* PROGRAM WILL NOT START EXECUTION UNTIL THE *
                       602            ;* FIRST SOF_INTERRUPT HAS BEEN SERVICED.     *
                       603            ;*********************************************
                       604
                       605
0156                   606     LABEL_WAIT_FOR_NEW_FE:
                       607
                       608                      ;LOOP AROUND UNTIL THERE IS A NEW FE TO SERVICE
                       609
0156 39368400          610                      CMP     FE_SERVICE_PTR, SI
015A 74FA              611                      JE      LABEL_WAIT_FOR_NEW_FE
                       612
                       613                      ;TWO POINTERS ARE NOT EQUAL, POSSIBLE NEW FE'S
                       614
                       615
                       616                      ;CHECK SERVICE COUNTER
015C 833E820000        617                      CMP     FE_SERVICE_COUNTER, 0
0161 7470              618                      JE      LABEL_PTR_ERROR         ;THE SERVICE PTR AND
                       619                                                      ;INPUT POINTER(SI) ARE NOT
                       620                                                      ;EQUAL, SERVICE COUNTER = 0
                       621                                                      ;SYNC ERROR
                       622
                       623
                       624                      ;SERVICE COUNTER >=1 , NORMAL CONDITION
                       625
0163 833E820008        626                      CMP     FE_SERVICE_COUNTER, 8
0168 7375              627                      JAE     LABEL_PTR_ERROR         ;THERE ARE MORE THAN 4 PAIRS
                       628                                                      ;OF FEX TO BE SERVICED. CPU
                       629                                                      ;IS RUNNING TOO SLOW
                       630
                       631                      ;1 =< SERVICE COUNTER < 8, NORMAL CONDITION.
                       632                      ; BEGIN SERVICE THE FE'S
                       633
                       634
                       635
                       636 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         637
016A                     638     LABEL_SERVICE_FE:
                         639                       ;MOVE THE NEW FEX INTO AX AND CORRESPONDING FSV TO CX
                         640
                         641                       ;NEED TO USE THE BX REGISTER TO ACCESS THE FE_BUFFER
016A 8B0E8400            642                       MOV     CX, FE_SERVICE_PTR
                         643
016E FA                  644                       CLI
016F 87CB                645                       XCHG    CX, BX
0171 8B4730              646                       MOV     AX, FE_BUFFER[BX]
0174 87CB                647                       XCHG    CX, BX
0176 FB                  648                       STI
                         649
0177 833E860000          650                       CMP     FE_TYPE, FE_LOW
017C 7520                651                       JNE     LABEL_FE_HIGH
                         652
                         653
                         654
                         655
                         656                       ;IT IS FEL TO BE PROCESSED
017E 8B0E7B00            657                       MOV     CX, FSVL
0182 E86D00              658                       CALL    APCMQ
                         659
                         660                       ;RESULT FSV IN DX, XOUT IN CX
                         661
0185 89167B00            662                       MOV     FSVL, DX                  ;SAVE THE NEW FSVL
                         663                       ;STORE THE NEW XOUT INTO THE FE_BUFFER
0189 A18400              664                       MOV     AX, FE_SERVICE_PTR
018C FA                  665                       CLI
018D 93                  666                       XCHG    AX, BX
                         667
018E 894F30              668                       MOV     FE_BUFFER[BX], CX
0191 FF0E8200            669                       DEC     FE_SERVICE_COUNTER
                         670
0195 93                  671                       XCHG    AX, BX
0196 FB                  672                       STI
                         673
                         674                       ;UPDATE THE FE TYPE TO BE SERVICED NEXT
0197 C70686000100        675                       MOV     FE_TYPE, FE_HIGH
                         676
                         677
                         678                       ;INCREMENT THE SERVICE POINTER, AX CONTAINS FE_SERVICE_PTR
019D 050200              679                       ADD     AX, TYPE FE_BUFFER
01A0 3D4000              680                       CMP     AX, LEN_FE_BUFFER * TYPE FE_BUFFER      ;END OF BUFFER?
01A3 7331                681                       JAE     LABEL_WRAP_AROUND
01A5 A38400              682                       MOV     FE_SERVICE_PTR, AX
01A8 EBAC                683                       JMP     LABEL_WAIT_FOR_NEW_FE
                         684
                         685 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           686
                           687
01AA                       688     LABEL_FE_HIGH:  ;THE FEH IS TO BE PROCESSED.
01AA BB0E7A00              689                     MOV     CX, FSVH
01AE E84100                690                     CALL    APCMQ
                           691
                           692                     ;RESULT FSV IN DX, XOUT IN CX
                           693
01B1 89167A00              694                     MOV     FSVH, DX                 ;SAVE THE NEW FSVH
                           695
                           696                     ;STORE THE NEW XOUT INTO THE FE_BUFFER
01B5 A18400                697                     MOV     AX, FE_SERVICE_PTR
                           698
01B8 FA                    699                     CLI
01B9 93                    700                     XCHG    AX, BX
                           701
01BA 894F30                702                     MOV     FE_BUFFER[BX], CX
01BD FF0E8200              703                     DEC     FE_SERVICE_COUNTER
                           704
01C1 93                    705                     XCHG    AX, BX
01C2 FB                    706                     STI
                           707
                           708                     ;UPDATE THE FE TYPE TO BE SERVICED NEXT
01C3 C70686000000          709                     MOV     FE_TYPE, FE_LOW
                           710
                           711
                           712                     ;INCREMENT THE SERVICE POINTER
01C9 050200                713                     ADD     AX, TYPE FE_BUFFER
01CC 3D4000                714                     CMP     AX, LEN_FE_BUFFER * TYPE FE_BUFFER      ;END OF BUFFER?
01CF 7305                  715                     JAE     LABEL_WRAP_AROUND
01D1 A38400                716                     MOV     FE_SERVICE_PTR, AX
                           717
01D4 EB80                  718                     JMP     LABEL_WAIT_FOR_NEW_FE
                           719
                           720
01D6                       721     LABEL_WRAP_AROUND:
                           722                     ;SERVICE POINTER WAS POINTING TO THE LAST ELEMENT IN FE
                           723                     ;ARRAY, MOVE TO THE FIRST ELEMENT.
                           724
01D6 C70684000000          725                     MOV     FE_SERVICE_PTR, 0
01DC E977FF                726                     JMP     LABEL_WAIT_FOR_NEW_FE
                           727
                           728 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         729
01DF                     730     LABEL_PTR_ERROR:
                         731
                         732             ;*****************************************************************
                         733             ;* THIS PART OF CODE HANDLES THE SYNCHRONIZATION ERROR           *
                         734             ;* BETWEEN THE QUANTIZATION PROCESS AND THE VARIOUS              *
                         735             ;* INTERRUPT DRIVEN I/O PROCESSES.                               *
                         736             ;*                                                               *
                         737             ;* AN ERROR CONDITION MAY BE ONE OR MORE OF THE FOLLOWINGS:      *
                         738             ;*                                                               *
                         739             ;* (1) BOTH INPUT(REGISTER SI - DEDICATED) AND SERVICE POINTER *
                         740             ;*     ARE POINTING TO DIFFERENT ELEMENT IN THE FE_BUFFER        *
                         741             ;*     AND THE SERVICE COUNTER HAS A VALUE OF ZERO.              *
                         742             ;* (2) THE SERVICE COUNTER HAS A VALUE HIGHER THAN 8. THIS       *
                         743             ;*     MEANS THAT FOUR PAIRS OF FEX OR MORE ARE NOT QUANTIZED  *
                         744             ;*     YET.                                                      *
                         745             ;* THE MULDEM SIMULATOR PROCESS WILL CONTINUE ONCE THE           *
                         746             ;* ABOVE ERROR(S) OCCURS. THE FOLLOWING ROUTINE WILL CAUSE THE *
                         747             ;* MOST SIGNIFICANT ERROR_LED TO INCREMENT BY 1.                 *
                         748             ;*****************************************************************
                         749
                         750
01DF FA                  751                     CLI                             ;DISABLE EXTERNAL INTERRUPT
01E0 A17000              752                     MOV     AX,LED_DISPLAY_VALUE
01E3 80C410              753                     ADD     AH,10H
01E6 7207                754                     JC      LABEL_PTR_EXT
01E8 A37000              755                     MOV     LED_DISPLAY_VALUE,AX
01EB 0AC4                756                     OR      AL,AH
01ED E664                757                     OUT     ERROR_LED,AL
01EF E96AFF              758     LABEL_PTR_EXT:  JMP     ERROR_ENTRY
                         759
                         760                     ;***********************************
                         761                     ;*    END  OF  MAIN  PROGRAM    *
                         762                     ;***********************************
                         763
                         764
                         765
                         766 +1  $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          767
                          768
01F2                      769     APCMQ           PROC            NEAR
                          770
                          771
                          772     ;*************************************************************************
                          773     ;* PURPOSE:                                                              *
                          774     ;*       THIS ROUTINE QUANTIZES A SAMPLE USING AN ADAFIVE QUANTIZER.      *
                          775     ;*                                                                       *
                          776     ;*                                                                       *
                          777     ;* DESCRIPTION:                                                          *
                          778     ;*       THE INPUT SAMPLE IS QUANTIZED, USING THE GIVEN SCALING FACTOR   *
                          779     ;*       FOR THE QUANTIZER. THE SCALING FACTOR IS UPDATED ON RETURN.     *
                          780     ;*                                                                       *
                          781     ;*                                                                       *
                          782     ;* PARAMETERS:                                                           *
                          783     ;*       INPUT :                                                         *
                          784     ;*         XIN    - INPUT SAMPLE(PASSED IN AX REGISTER)                  *
                          785     ;*         FSV    - SCALING FACTOR FOR THE QUANTIZER. THIS VALUE IS      *
                          786     ;*                  UPDATED ON OUTPUT.(INPUTED IN CX REGISTER)           *
                          787     ;*                                                                       *
                          788     ;*        OUTPUT:                                                        *
                          789     ;*         XOUT   - OUTPUT QUANTIZED SAMPLE (PASSED IN CX REGISTER)      *
                          790     ;*         FSV    - NEW SCALING FACTOR FOR THE QUANTIZER (OUTPUT         *
                          791     ;*                  IN DX REGISTER)                                      *
                          792     ;*                                                                       *
                          793     ;*       PRE-SPECIFIED:                                                  *
                          794     ;*         NLEV   - NUMBER OF POSITIVE QUANTIZER LEVELS. THE QUANTIZER   *
                          795     ;*                 IS ASSUMED TO BE SYMMETRIC ABOUT ZERO. THE TOTAL       *
                          796     ;*                 NUMBER OF QUANTIZER LEVELS IS 2*NLEV.                  *
                          797     ;*                                                                       *
                          798     ;* ROUTINES REQUIRED:                                                    *
                          799     ;*      IQUANTZ - QUANTIZE A POINT                                       *
                          800     ;*************************************************************************
                          801
                          802 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           803
                           804
01F2 890E7600              805                  MOV    FSV, CX                ;FSV IS PASSED TO THIS PROCEDURE FROM
                           806                                                ;REGISTER CX. SAVE FOR FUTURE REFEREN
                                   CE
                           807                                                ;THIS VALUE IS ALSO USED IN PROCEDURE
                                    IQUANTZ
                           808
                           809                  ;XIN IS PASSED TO THIS PROCEDURE IN REGISTER AX.
                           810
01F6 C7068B000000          811                  MOV    SIGN_FLAG, POSITIVE    ;INITIALIZE THE SIGN FLAG
                           812
01FC A9FFFF                813                  TEST   AX, 0FFFFH             ;GET SIGN
01FF 7908                  814                  JNS    LABEL_1                ;IT IS A POSITIVE NUMBER
                           815
                           816                  ;XIN IS A NEGATIVE NUMBER, GET ABSOLUTE VALUE
                           817
0201 F7D8                  818                  NEG    AX
0203 C7068B000100          819                  MOV    SIGN_FLAG, NEGATIVE    ;STORE STATE OF SIGN
                           820
0209 A37200                821     LABEL_1:     MOV    ABS_XIN, AX            ;SAVE THE ABSOLUTE XIN FOR
                           822                                                ;USE IN IQUANTZ PROCEDURE
                           823
                           824
                           825                  ;REGISTER AX, CX AND DX ARE NOT PRESERVED IN THIS CALL.
020C E86700                826                  CALL   IQUANTZ
                           827
                           828                  ;RESULT 'L2' RETURNED IN CX. THIS VALUE IS DOUBLED THE
                           829                  ;ACTUAL VALUE TO FACILITATE INDEX ADDRESSING.
                           830
020F 890E8000              831                  MOV    L2, CX                 ;SAVE FOR FUTURE REFENRECE
                           832
                           833 +1  $EJECT
```

```
LOC  OBJ                  LINE    SOURCE

                           834                  ;**********************************************
                           835                  ;* GENERATE QUANTIZED VALUE FOR THE CODED BITS *
                           836                  ;**********************************************
                           837
0213 A17600                838                  MOV    AX, FSV
                           839
0216 3D7B00                840                  CMP    AX, FSVTH
                           841
0219 730E                  842                  JAE    LABEL_MID_RISE
                           843
                           844                  ;FSV < FSVTH
                           845
                           846
021B 83F902                847                  CMP    CX, 2                 ;L2 = 1?
                           848
021E 7509                  849                  JNE    LABEL_MID_RISE
                           850
                           851                  ;L2 = 1 , USE MID_TREAD QUANTIZER AND SET XOUT = 0
                           852
0220 C70674000000          853                  MOV    XOUT, 0
0226 EB1B90                854                  JMP    INC_STEP
                           855
                           856
0229                       857    LABEL_MID_RISE:
                           858
                           859                  ;NEED TO USE BX AT THIS POINT, DISABLE INTERRUPT BEFORE USE
                           860                  ;BX IS USED BY INTERRUPT I/O ROUTINES AS A DEDICATED REGISTER
                           861
0229 FA                    862                  CLI
022A 87D9                  863                  XCHG   BX, CX                ;CX CONTAINS 'L2'
                           864
022C 2EF727                865                  MUL    CS:YQ[BX]             ;AX CONTAINS FSV
                           866
022F 87D9                  867                  XCHG   BX, CX
                           868
0231 FB                    869                  STI
                           870
0232 D1D0                  871                  ROL    AX, 1                 ;SCALE BY 2**15
0234 D1D2                  872                  RCL    DX, 1
                           873
                           874                  ;INCORPORATE SIGN BIT
0236 833E860000            875                  CMP    SIGN_FLAG, 0
023B 7402                  876                  JZ     LABEL_2
                           877
                           878                  ;XIN WAS A NEGATIVE NUMBER
                           879
023D F7DA                  880                  NEG    DX
                           881
023F 89167400              882    LABEL_2:      MOV    XOUT, DX
                           883
                           884 +1 $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           885
                           886                        ;******************************
                           887                        ;* INCREMENT STEP SIZE OF FSV *
                           888                        ;******************************
                           889
                           890
0243 8B0E8000              891     INC_STEP:          MOV     CX, L2
                           892
                           893                        ;NEED TO USE BX AGIAN, REFER TO PREVIOUS USAGE FOR DOCUMENTATION
                           894
0247 FA                    895                        CLI
0248 87CB                  896                        XCHG    CX, BX
024A 2E8B4742              897                        MOV     AX, CS:QMLT[BX]
024E 87CB                  898                        XCHG    CX, BX
0250 FB                    899                        STI
0251 F7267600              900                        MUL     FSV
                           901
0255 D1C0                  902                        ROL     AX, 1                    ;DIVIDE RESULT BY 2**14
0257 D1D2                  903                        RCL     DX, 1
0259 D1C0                  904                        ROL     AX, 1
025B D1D2                  905                        RCL     DX, 1
                           906
025D 81FA401F              907                        CMP     DX, FSVMX
0261 7606                  908                        JBE     LABEL_3
0263 BA401F                909                        MOV     DX, FSVMX
0266 EB0990                910                        JMP     APCMQ_END
0269 83FA50                911     LABEL_3:           CMP     DX, FSVMN
026C 7303                  912                        JAE     APCMQ_END
026E BA5000                913                        MOV     DX, FSVMN
                           914
                           915                        ;NEW FSV IN DX
                           916
0271 8B0E7400              917     APCMQ_END:         MOV     CX, XOUT
0275 C3                    918                        RET
                           919
                           920     APCMQ              ENDP
                           921
                           922
                           923
                           924 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         925
0276                     926     IQUANTZ             PROC          NEAR
                         927
                         928            ;*************************************************************
                         929            ;* INPUT :          ABS_XIN , FSV PASSED FROM MEMORY           *
                         930            ;*                                                             *
                         931            ;* OUTPUT:          L       IN REGISTER CX. THIS VALUE IS      *
                         932            ;*                          DOUBLE THAT OF ACTUAL VALUE TO     *
                         933            ;*                          FACILITATE INDEX ADDRESSING        *
                         934            ;* REGISTER USAGE:                                             *
                         935            ;*                  BX      TEMPORARILY, VALUE SAVED AND       *
                         936            ;*                          RESTORED BY THIS PROCEDURE.        *
                         937            ;*                          INTERRUPT DISABLED WHILE USING THIS*
                         938            ;*                          REGISTER.                          *
                         939            ;*                  DX                                         *
                         940            ;*                                                             *
                         941            ;* REGISTER ASSIGNMENT IN THIS PROCEDURE:                      *
                         942            ;*                                                             *
                         943            ;*                  CX      CONTAINS IU                        *
                         944            ;*          REFER TO FORTRAN LISTING FOR DESCRIPTION OF THESE  *
                         945            ;*          VARIABLES.                                         *
                         946            ;*************************************************************
                         947
                         948
                         949 +1  $EJECT
```

```
LOC  OBJ               LINE   SOURCE


0276 C7067E000000                              MOV     IL, 0                           ;IL = 0
027C B92000                                    MOV     CX, NLEV* TYPE XQ               ;IU = NLEV [* 2 FACTOR ADDED]

                                               ;I = (IL + IU) /2

027F 8B167E00                 LABEL_100:       MOV     DX, IL
0283 03D1                                      ADD     DX, CX
0285 D1EA                                      SHR     DX, 1                           ;DIVIDE BY 2
0287 81E2FEFF                                  AND     DX, 0FFFEH                      ;MAKE SURE IT IS AN EVEN NUMB
                              ER
028B 89167C00                                  MOV     I, DX                           ;SAVE I FOR FUTURE USE
                                               ;IF (X .GT. XQ(I)*FSV) GOTO 220

                                               ;NEED TO USE REGISTER BX AT THIS POINT. DISABLE INTERRUPT BEFORE
                                               ;USING


028F FA                                        CLI
0290 87D3                                      XCHG    DX, BX
0292 2E8B4722                                  MOV     AX, CS:XQ[BX]                   ;MOV XQ TO AX REGISTER
0296 87D3                                      XCHG    DX, BX
0298 FB                                        STI

0299 F7267600                                  MUL     FSV                             ;XQ IS SCALED BY 2**15
029D D1C0                                      ROL     AX, 1
029F D1D2                                      RCL     DX, 1
02A1 39167200                                  CMP     ABS_XIN, DX
02A5 7707                                      JA      LABEL_200

                                               ;IU = I
02A7 8B0E7C00                                  MOV     CX, I
02AB EB0990                                    JMP     LABEL_300

02AE                          LABEL_200:       ;IL = I
02AE 8B167C00                                  MOV     DX, I
02B2 89167E00                                  MOV     IL, DX


02B6 8B167E00                 LABEL_300:       MOV     DX, IL
02BA 83C202                                    ADD     DX, 2                           ;IL+ 1
02BD 3BCA                                      CMP     CX, DX
02BF 77BE                                      JA      LABEL_100

                                               ;RESULT IU IN CX REGISTER


02C1 C3                                        RET

                              IQUANTZ          ENDP





```

```
LOC  OBJ                  LINE     SOURCE

                          1004
                          1005
                          1006
----                      1007     CODE                ENDS
                          1008
                          1009
                          1010
                          1011 +1  $EJECT
```

```
LOC  OBJ          LINE    SOURCE




----                      INTERRUPT_CODE          SEGMENT

                                  ASSUME          CS:INTERRUPT_CODE, DS:DATA, ES:NOTHING, SS:STACK

0000                      SOF_INTERRUPT           PROC            FAR

                          ;*************************************************************************
                          ;* THIS INTERRUPT ROUTINE IS ACTIVATED BY THE SOF_(START_OF_FRAME) SINGNAL *
                          ;* ON THE CODER BUS(THE SOF_ ON THE DECODER BUS IS SYNCHRONIZED AND OCCURS *
                          ;* AT THE SAME TIME AS THE DECODER BUS SOF_).                              *
                          ;* THIS ROUTINE IS RESPONSIBLE FOR CHECKING THE MULDEM OVERALL FIRMWARE    *
                          ;* STATE, INCLUDING FOLLOWING:                                             *
                          ;*              - FE INPUT AND OUTPUT POINTER VALUES                       *
                          ;*              - PREDICTOR COEFFICIENTS BUFFER FRAME AND OFFSET POINTER   *
                          ;*                VALUES                                                   *
                          ;*                                                                         *
                          ;* IF ANY OF THESE VALUES IS ABNORMAL, THIS ROUTINE WILL ATTEMP TO CORRECT *
                          ;* TO THE BEST OF ITS KNOWLEDGE. AT THE SAME TIME, IT WILL INCREMENT THE   *
                          ;* ERROR LOG VALUE AND OUTPUT TO THE LEAST SIGNIFICANT LED.                *
                          ;* THIS ROUTINE IS ALSO RESPONSIBLE FOR SETTING UP THE PRIDICTION COEFF    *
                          ;* BUFFER POINTERS FOR THE CURRENT FRAME.                                  *
                          ;*                                                                         *
                          ;*             REGISTER USAGES:                                            *
                          ;*                     AX, BX, BP, SI, DI                                  *
                          ;*                                                                         *
                          ;*************************************************************************


                     +1   $EJECT
```

```
LOC  OBJ                  LINE      SOURCE

                          1045
                          1046                  ;NOTE THAT INTERRUPT FLAG (IF) IS DISABLED IN MOST PART
                          1047                  ;OF THIS ROUTINE.
                          1048
0000 50                   1049                          PUSH    AX                      ;MAY USE AX REGISTER TO
                          1050                                                          ;OUTPUT ERROR_LED IN THIS
                          1051                                                          ;ROUTINE
                          1052                  ;*************************************
                          1053                  ;*   FE_BUFFER        CHECK          *
                          1054                  ;*************************************
                          1055
                          1056
                          1057
                          1058                          ;CHECK FE INPUT POINTER
                          1059
0001 83FE10               1060                          CMP     SI, 16                  ;THIS IS THE NORMAL OFFSET
                          1061                                                          ;VALUE POINTING TO FEL(4)
0004 7413                 1062                          JE      LABEL_NORMAL_1
                          1063
                          1064 +1
                          1065 +1
                          1066 +1
                          1067 +1
0006 A17000               1068 +1                       MOV     AX,LED_DISPLAY_VALUE
0009 FEC0                 1069 +1                       INC     AL
000B 240F                 1070 +1                       AND     AL,0FH
                          1071
000D 7407                 1072                          JZ      LED_ERROR_EXT1
                          1073
                          1074 +1
000F A37000               1075 +1                       MOV     LED_DISPLAY_VALUE, AX
0012 0AC4                 1076 +1                       OR      AL,AH
0014 E664                 1077 +1                       OUT     ERROR_LED, AL
                          1078
0016 BE1000               1079      LED_ERROR_EXT1:     MOV     SI, 16                  ;CORRECT POINTER VALUE
                          1080
0019                      1081      LABEL_NORMAL_1:     ;CHECK FE OUTPUT POINTER
                          1082
0019 83FB00               1083                          CMP     BX, 0                   ;THIS IS THE NORMAL VALUE
                          1084                                                          ;POINTING TO FEL(0)
001C 7413                 1085                          JE      LABEL_NORMAL_2
                          1086
                          1087 +1
                          1088 +1
                          1089 +1
                          1090 +1
001E A17000               1091 +1                       MOV     AX,LED_DISPLAY_VALUE
0021 FEC0                 1092 +1                       INC     AL
0023 240F                 1093 +1                       AND     AL,0FH
                          1094
0025 7407                 1095                          JZ      LED_ERROR_EXT2
                          1096
                          1097 +1
0027 A37000               1098 +1                       MOV     LED_DISPLAY_VALUE, AX
002A 0AC4                 1099 +1                       OR      AL,AH
```

```
LOC  OBJ                LINE     SOURCE

002C E664               1100 +1                        OUT     ERROR_LED, AL
                        1101
002E BB0000             1102     LED_ERROR_EXT2:       MOV     BX, 0                       ;CORRECT POINTER VALUE
                        1103
                        1104                     ;************************************
                        1105                     ;*   PREDICTOR BUFFER CHECK         *
                        1106                     ;************************************
                        1107
0031                    1108     LABEL_NORMAL_2:       ;CHECK PREDICTOR INPUT POINTER
                        1109
0031 83FF10             1110                           CMP     DI, LEN_P_BUFFER_FRAME * (TYPE PREDICTOR_BUFFER)
                        1111                                                       ;THIS IS THE NORMAL VALUE
                        1112                                                       ;POINTING TO ELEMENT AFTER
                        1113                                                       ;THE LAST ELEMENT OF A FRAME
                        1114
0034 7410               1115                           JE      LABEL_NORMAL_3
                        1116
                        1117                           ;THE POINTER CONTAINS ILLEGAL VALUE
                        1118
                        1119
                        1120 +1
                        1121 +1
                        1122 +1
                        1123 +1
0036 A17000             1124 +1                        MOV     AX,LED_DISPLAY_VALUE
0039 FEC0               1125 +1                        INC     AL
003B 240F               1126 +1                        AND     AL,0FH
                        1127
003D 7407               1128                           JZ      LABEL_NORMAL_3
                        1129
                        1130 +1
003F A37000             1131 +1                        MOV     LED_DISPLAY_VALUE, AX
0042 0AC4               1132 +1                        OR      AL,AH
0044 E664               1133 +1                        OUT     ERROR_LED, AL
                        1134
0046                    1135     LABEL_NORMAL_3:       ;SET THE PREDICTOR POINTERS FOR NEXT FRAME OUTPUT
                        1136
0046 BF0000             1137                           MOV     DI, 0
0049 83C510             1138                           ADD     BP, LEN_P_BUFFER_FRAME *(TYPE PREDICTOR_BUFFER)
                        1139                                                       ;INCREMENT TO NEXT FRAME
                        1140
004C 83FD20             1141                           CMP     BP, LAST_P_BUFFER_FRAME ;DETERMINE IF BP POINTS TO
                        1142                                                       ;LAST FRAME IN BUFFER
                        1143
004F 7603               1144                           JBE     LABEL_SOF_END
                        1145
                        1146                           ;BP WAS POINTING TO LAST FRAME IN BUFFER BEFORE INCREMENT
                        1147
0051 BD0000             1148                           MOV     BP, 0                       ;MOVE TO THE FIRST FRAME IN B
                                 UFFER
                        1149
0054                    1150 +1  LABEL_SOF_END:
                        1151 +1
                        1152 +1
                        1153 +1
```

```
LOC  OBJ                  LINE     SOURCE

0054 FA                   1154 +1                       CLI
0055 B8E000               1155 +1                       MOV      AX, ENABLE_NORMAL_MASK
0058 E67A                 1156 +1                       OUT      PIC_PORT_1, AL
005A FB                   1157 +1                       STI
005B 58                   1158                          POP      AX
005C CF                   1159                          IRET
                          1160
                          1161
                          1162     SOF_INTERRUPT        ENDP
                          1163
                          1164
                          1165
                          1166 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          1167
005D                      1168    PREDICTOR_OUTPUT_INTERRUPT      PROC            FAR
                          1169
                          1170
                          1171    ;*************************************************************************
                          1172    ;* THIS PROCEDURE IS ACTIVATED BY THE SPEN_ SIGNAL ON THE DECODER BUS.    *
                          1173    ;* THIS ROUTINE IS RESPONSIBLE FOR OUTPUT THE PREDICTOR COEFFICIENTS FROM  *
                          1174    ;* THE PREDICTOR COEFF BUFFER.                                            *
                          1175    ;*                                                                        *
                          1176    ;*              REGISTER USAGES:                                          *
                          1177    ;*                       AX, DI                                           *
                          1178    ;*                                                                        *
                          1179    ;*************************************************************************
                          1180
                          1181
005D 50                   1182                            PUSH     AX                    ;AX IS USED DURING THIS ROUTI
                                  NE.
                          1183 +1
                          1184 +1
                          1185 +1
                          1186 +1
005E B8FE00               1187 +1                         MOV      AX,ENABLE_SOF_MASK
0061 E67A                 1188 +1                         OUT      PIC_PORT_1, AL
0063 FB                   1189 +1                         STI
                          1190
                          1191
                          1192                            ;CONFIGURE THE DECODER DATA BUS TO BE OUTPUT
                          1193
0064 B89999               1194                            MOV      AX, D_DATA_BUS_OUT
0067 E74E                 1195                            OUT      DECODER_BUS_SPVR, AX
                          1196
                          1197                            ;OUTPUT DATA
                          1198
0069 3E8B03               1199                            MOV      AX, DS:PREDICTOR_BUFFER[BP][DI]
                          1200
006C E74A                 1201                            OUT      D_DATA_BUS, AX
                          1202
                          1203                            ;INCREMENT BUFFER POINTER
                          1204
006E 83C702               1205                            ADD      DI, TYPE PREDICTOR_BUFFER
0071 83FF10               1206                            CMP      DI, LEN_P_BUFFER_FRAME * TYPE PREDICTOR_BUFFER
0074 7203                 1207                            JB       LABEL_P_EN
                          1208
                          1209                            ;DI POINTS BEYOND END OF CURRENT FRAME, END OF OUTPUT
                          1210                            ;OF A FRAME(THE N - 2ND FRAME) PREDICTOR COEFFICIENTS.
                          1211                            ;SET UP THE POINTER FOR INPUT NEW PREDICTOR
                          1212                            ;COEFFICIENTS AT END OF CURRENT FRAME(FRAME N).
                          1213                            ;THE INPUT DATA WILL BE DEPOSITED INTO THE SAME FRAME
                          1214                            ;BUFFER, THEREFORE BP REGISTER NEEDS NOT BE CHANGED.
                          1215
0076 BF0000               1216                            MOV      DI, 0
                          1217
                          1218
0079                      1219    LABEL_P_EN:             ;CHECK FOR END OF OUTPUT CYCLE
0079 E54C                 1220                            IN       AX, D_CONTROL_BUS
```

```
LOC  OBJ                  LINE     SOURCE

007B 250400               1221                         AND     AX, SPEN
007E 74F9                 1222                         JZ      LABEL_P_EN
                          1223
                          1224                         ;NOW SPEN IS FALSE(I.E. SIGNAL IS HIGH), TAKE DATA OFF THE BU
                                   S
                          1225
0080 B89B9B               1226                         MOV     AX, D_DATA_BUS_IN
0083 E74E                 1227                         OUT     DECODER_BUS_SPVR, AX
                          1228
                          1229 +1
                          1230 +1
                          1231 +1
                          1232 +1
0085 FA                   1233 +1                      CLI
0086 B8E000               1234 +1                      MOV     AX, ENABLE_NORMAL_MASK
0089 E67A                 1235 +1                      OUT     PIC_PORT_1, AL
008B FB                   1236 +1                      STI
                          1237
008C 58                   1238                         POP     AX
008D CF                   1239                         IRET
                          1240
                          1241
                          1242     PREDICTOR_OUTPUT_INTERRUPT      ENDP
                          1243
                          1244
                          1245
                          1246 +1  $EJECT
```

```
LOC  OBJ            LINE      SOURCE

                    1247
                    1248
008E                1249      FE_OUTPUT_INTERRUPT     PROC     FAR
                    1250
                    1251
                    1252      ;*****************************************************************************
                    1253      ;* THIS INTERRUPT HANDLER IS ACTIVATED BY FE ENABLE_ SIGNAL ON THE DECODER *
                    1254      ;* BUS. IT IS RESPONSIBLE FOR OUTPUTING FEH AND FEL TO THE DECODER  BUS    *
                    1255      ;* ON ENTRY TO THE HANDLER, AX IS PUSHED ONTO STACK, DATA BUS PORT IS      *
                    1256      ;* CONVERTED FROM INPUT STATE TO OUTPUT STATE, THE APPROPRIATE FEX IS      *
                    1257      ;* OUTPUTED TO THE BUS, POINTER TO BUFFER IS INCREMENTED TO NEXT FEX TO BE *
                    1258      ;* OUTPUT. THIS ROUTINE WILL LOOP AROUND UNTIL FE ENDABLE_ IS HIGH, THEN   *
                    1259      ;* IT WILL PULL DATA OFF FROM THE DATA BUS, CONVERT PORT TO INPUT STATE    *
                    1260      ;* AND EXIT                                                                *
                    1261      ;*                                                                         *
                    1262      ;* REGISTER USAGE:                                                         *
                    1263      ;*                AX, BX                                                   *
                    1264      ;*****************************************************************************
                    1265
                    1266
008E 50             1267                             PUSH     AX
                    1268
                    1269 +1
                    1270 +1
                    1271 +1
                    1272 +1
008F B8FE00         1273 +1                          MOV      AX,ENABLE_SOF_MASK
0092 E67A           1274 +1                          OUT      PIC_PORT_1, AL
0094 FB             1275 +1                          STI
                    1276
                    1277
                    1278                             ;CONFIGURE DATA BUS PORT TO BE OUTPUT
                    1279
0095 B89999         1280                             MOV      AX, D_DATA_BUS_OUT
0098 E74E           1281                             OUT      DECODER_BUS_SPVR, AX
                    1282
009A 8B4730         1283                             MOV      AX, FE_BUFFER[BX]
                    1284
                    1285                             ;BX IS THE POINTER TO THE NEXT WORD TO BE OUTPUT IN FE BUFFER
                    1286
009D E74A           1287                             OUT      D_DATA_BUS, AX
                    1288
                    1289                             ;INCREMENT THE OUTPUT POINTER
                    1290
009F 83C302         1291                             ADD      BX, TYPE FE_BUFFER
00A2 83FB40         1292                             CMP      BX, LEN_FE_BUFFER * TYPE FE_BUFFER
00A5 7203           1293                             JB       LABEL_FE_ENABLE
00A7 BB0000         1294                             MOV      BX, 0
                    1295
                    1296
                    1297
00AA E54C           1298      LABEL_FE_ENABLE:       IN       AX, D_CONTROL_BUS
00AC 251000         1299                             AND      AX, SFEEN
00AF 74F9           1300                             JZ       LABEL_FE_ENABLE
                    1301
```

```
LOC  OBJ                  LINE     SOURCE

                          1302                                  ;NOW SFEEN_ IS FALSE(SIGNAL IS HIGH), TAKE DATA OFF BUS
00B1 B89B9B               1303                                  MOV     AX, D_DATA_BUS_IN
00B4 E74E                 1304                                  OUT     DECODER_BUS_SPVR, AX
                          1305
                          1306 +1
                          1307 +1
                          1308 +1
                          1309 +1
00B6 FA                   1310 +1                               CLI
00B7 B8E000               1311 +1                               MOV     AX, ENABLE_NORMAL_MASK
00BA E67A                 1312 +1                               OUT     PIC_PORT_1, AL
00BC FB                   1313 +1                               STI
                          1314
00BD 58                   1315                                  POP     AX
                          1316
00BE CF                   1317                                  IRET
                          1318
                          1319     FE_OUTPUT_INTERRUPT          ENDP
                          1320
                          1321
                          1322
                          1323 +1  $EJECT
```

```
LOC  OBJ              LINE     SOURCE

                      1377
00D9 83FE40           1378                           CMP    SI, LEN_FE_BUFFER * TYPE FE_BUFFER       ;END OF BUFFE
                               R?
00DC 7203             1379                           JB     LABEL_FE_INPUT_END
00DE BE0000           1380                           MOV    SI, 0                      ;RESET POINTER TO BEGINNING
                      1381                                                             ;OF CIRCULAR BUFFER.
                      1382
00E1                  1383 +1  LABEL_FE_INPUT_END:
                      1384 +1
                      1385 +1
                      1386 +1
00E1 FA               1387 +1                        CLI
00E2 B8EC00           1388 +1                        MOV    AX, ENABLE_NORMAL_MASK
00E5 E67A             1389 +1                        OUT    PIC_PORT_1, AL
00E7 FB               1390 +1                        STI
00E8 58               1391                           POP    AX
                      1392
00E9 CF               1393                           IRET
                      1394
                      1395     FE_INPUT_INTERRUPT    ENDP
                      1396
                      1397
                      1398 +1  $EJECT
```

```
LOC  OBJ             LINE     SOURCE

                     1399
                     1400
                     1401
00EA                 1402     PREDICTOR_INPUT_INTERRUPT       PROC    FAR
                     1403
                     1404
                     1405
                     1406     ;*************************************************************************
                     1407     ;* THIS PROCEDURE IS ACTIVATED BY THE PEN_  SIGNAL ON THE CODER BUS.      *
                     1408     ;* IT INPUTS THE PREDICTOR COEFFICIENTS FROM THE CODER BUS AND STORES THE  *
                     1409     ;* DATA IN THE PREDICTOR_COEFF BUFFER.                                     *
                     1410     ;*                                                                         *
                     1411     ;*                REGISTER USAGES:                                         *
                     1412     ;*                          AX, DI                                         *
                     1413     ;*                                                                         *
                     1414     ;*************************************************************************
00EA 50              1415                             PUSH    AX
                     1416
                     1417 +1
                     1418 +1
                     1419 +1
                     1420 +1
00EB B8FE00          1421 +1                          MOV     AX,ENABLE_SOF_MASK
00EE E67A            1422 +1                          OUT     PIC_PORT_1, AL
00F0 FB              1423 +1                          STI
                     1424
                     1425                             ;TEST FOR P CLOCK VALID
                     1426
                     1427
00F1 E544            1428     LABEL_P_CLK:            IN      AX, C_CONTROL_BUS
                     1429
00F3 252000          1430                             AND     AX, CPCLK
00F6 74F9            1431                             JZ      LABEL_P_CLK
                     1432
                     1433
                     1434                             ;P CLOCK IS VALID, READ IN PREDICTOR COEFFICIENTS FROM
                     1435                             ;DATA BUS
                     1436
                     1437
00F8 E542            1438                             IN      AX, C_DATA_BUS
                     1439
                     1440                             ;BP SPECIFIES THE FRAME STARTING OFFSET(0, 16, 32 OR 48)
                     1441                             ;DI SPECIFIES THE OFFSET WITHIN EACH FRAME (0, 2, 4...14)
00FA 3E8903          1442                             MOV     DS:PREDICTOR_BUFFER[BP][DI], AX
                     1443
00FD 83C702          1444                             ADD     DI, TYPE PREDICTOR_BUFFER       ;INCREMENT OFFSET
                     1445                                                                     ;VALUE WITHIN ONE FRA
                              ME
                     1446
                     1447
                     1448 +1
                     1449 +1
                     1450 +1
                     1451 +1
0100 FA              1452 +1                          CLI
```

```
LOC  OBJ                  LINE     SOURCE

0101 B8E000               1453 +1                        MOV     AX, ENABLE_NORMAL_MASK
0104 E67A                 1454 +1                        OUT     PIC_PORT_1, AL
0106 FB                   1455 +1                        STI
                          1456
0107 58                   1457                           POP     AX
0108 CF                   1458                           IRET
                          1459
                          1460     PREDICTOR_INPUT_INTERRUPT       ENDP
                          1461
                          1462
                          1463
                          1464 +1  $EJECT
```

```
LOC  OBJ           LINE     SOURCE

0109                        EXCEPTION_INT          PROC    FAR

                                    ;*********************************************************
                                    ;* THIS IS THE EXCEPTION HANDLER FOR THE 8086 CPU.       *
                                    ;* UNDER NORMAL OPERATION ENVIRONMENT, THIS CODE SHOULD *
                                    ;* NEVER BE EXECUTED.                                    *
                                    ;* IN THE UNLIKELY EVENT THAT THIS HANDLER IS ACTIVATED,*
                                    ;* IT SETS THE LED_DISPLAY_VALUE TO OFFH AND OUTPUT      *
                                    ;* TO THE ERROR_LED.                                     *
                                    ;*             REGISTER USAGES:                          *
                                    ;*                     AX,                               *
                                    ;*                                                       *
                                    ;*********************************************************


                                    ;EXTERNAL INTERRUPTS ARE DISABLED DURING EXECUTION OF THIS HANDLER


0109 50                                                PUSH    AX
010A B8FFFF                                            MOV     AX, OFFFFH
010D A37000                                            MOV     LED_DISPLAY_VALUE, AX
0110 E664                                              OUT     ERROR_LED, AL
0112 58                                                POP     AX
0113 FB                                                STI
0114 CF                                                IRET


                            EXCEPTION_INT          ENDP



----                        INTERRUPT_CODE         ENDS
                                                   END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

B3. NO RESIDUAL QUANTIZER

```
VAX/VMS 8086/8087/8088 MACRO ASSEMBLER V1.0VX ASSEMBLY OF MODULE MQLDEM
OBJECT MODULE PLACED IN MQLDEM.OBJ
NO INVOCATION LINE CONTROLS

LOC  OBJ                  LINE     SOURCE

                            1
                            2
                            3
                            4                          NAME    MQLDEM
                            5
                            6
                            7
                            8          ;**************************************************************
                            9          ;*                                                            *
                           10          ;* PROJECT:       472A RELP CODEC                             *
                           11          ;* BOARD:         MULDEM SIMULATOR                            *
                           12          ;* DEVICE:        2 FIRMWARE EPROMS (2732A-2)                 *
                           13          ;* BOARD LOCATION: E15, E29                                   *
                           14          ;* PROGRAMMABLE                                               *
                           15          ;* DEVICE NUMBER:  60-0670, 60-0671                           *
                           16          ;*                                                            *
                           17          ;**************************************************************
                           18
                           19 +1   $EJECT
```

```
LOC  OBJ          LINE     SOURCE

                    20
                    21
                    22
                    23            ;*************************************************************
                    24            ;* THIS FIRMWARE IS IDENTICAL TO THE MQLDEM VERSION EXCEPT  *
                    25            ;* IN THE FOLLOWING AREAS:                                  *
                    26            ;*                                                          *
                    27            ;*          - THE FEX COEFFICIENTS ARE NOT QUANTIZED.       *
                    28            ;*          - NOISE SUPPRESSION IS IMPLEMENTED IN SUCH A WAY *
                    29            ;*            THAT WHEN INPUT FEX IS BELOW THE THRESHOLD    *
                    30            ;*            VALUE FSVTH, THE FEX VALUE WILL BE SUPPRESSED  *
                    31            ;*            TO ZERO.                                      *
                    32            ;*************************************************************
                    33
                    34
                    35
                    36
                    37              ;*****************************************************
                    38              ;*                                                   *
                    39              ;* THIS FIRMWARE SIMULATES THE MULtiplexor/DEMulti-  *
                    40              ;* plexor IN THE RELP CODEC SYSTEM. REFER TO MDA     *
                    41              ;* DOCUMENT 00 - 3035 - R01 FOR DETAIL.              *
                    42              ;*                                                   *
                    43              ;*****************************************************
                    44
                    45
                    46
                    47
                    48
                    49     ;***********************
                    50     ;*  SYSTEM CONSTANTS   *
                    51     ;***********************
                    52
                    53
                    54
 0020               55     LEN_FE_BUFFER         EQU    32     ;ACCOMODATE 16 PAIRS OF FEX
                    56
 0008               57     LEN_P_BUFFER_FRAME    EQU    8      ;EACH FRAME ACCOMODATES 8 PREDICTOR COEFFICIE
                           NTS
                    58
 0020               59     LAST_P_BUFFER_FRAME   EQU    32     ;THE BASE OFFSET ADDRESS OF LAST P FRAME
                    60                                          ;FROM THE BEGINNING OF P BUFFER. THIS VALUE
                    61                                          ;MUST BE IN MULTIPLE OF LEN_P_BUFFER_FRAME *
                           TYPE P_BUFFER
                    62                                          ;THE FRAME DELAY IS EQUAL TO 1 +
                    63                                          ;(LAST_P_BUFFER_FRAME / 16)
                    64
                    65
 0001               66     FE_HIGH               EQU    1      ;THIS REPRESENTS THE FEH'S TYPE
 0000               67     FE_LOW                EQU    0      ;THIS REPRESENTS THE FEL'S TYPE
                    68
 0002               69     NLEV                  EQU    2      ;NUMBER OF QUANTIZER REGIONS
                    70
 000A               71     FSVMN                 EQU    10     ;FSVMN  - MINIMUM SCALING FACTOR
                    72
```

```
LOC  OBJ            LINE     SOURCE

03E8                  73     FSVMX               EQU     1000    ;FSVMX  - MAXIMUM SCALING FACTOR
                      74
000F                  75     FSVTH               EQU     15      ;FSVTH  - THRESHOLD VALUE. IF THE
                      76                                         ;SCALE FACTOR IS LESS THAN FSVTH, A
                      77                                         ;MID-TREAD QUANTIZER IS USED INSTEAD
                      78                                         ;OF A MID-RISE QUANTIZER. THE MID-TREAD
                      79                                         ;QUANTIZER USES ZERO AS AN OUTPUT LEVEL
                      80                                         ;INSTEAD OF YQ(1).
                      81
0000                  82     POSITIVE            EQU     0       ;POSITIVE SIGN
                      83
0001                  84     NEGATIVE            EQU     1       ;NEGATIVE SIGN
                      85
                      86
                      87     ;    HARDWARE DEPENDENT CONSTANTS
                      88
0042                  89     C_DATA_BUS          EQU     42H     ;THIS IS THE PORT(B) ADDRESS ON THE 8255'S
                      90                                         ;FOR THE 16 BIT CODER DATA BUS
                      91
0044                  92     C_CONTROL_BUS       EQU     44H     ;THIS IS THE PORT(C) ADDRESS ON THE 8255'S
                      93                                         ;FOR THE 16 BIT CODER CONTROL BUS
                      94
0046                  95     CODER_BUS_SPVR      EQU     46H     ;THIS IS THE SUPERVISOR PORT(CONTROL PORT)
                      96                                         ;FOR THE CODER BUS(THE CONTROL BUS AND
                      97                                         ;THE DATA BUS)
                      98
                      99
                     100
004A                 101     D_DATA_BUS          EQU     4AH     ;THIS IS THE PORT(B) ADDRESS ON THE 8255'S
                     102                                         ;FOR THE 16 BIT DECODER DATA BUS
                     103
                     104
004C                 105     D_CONTROL_BUS       EQU     4CH     ;THIS IS THE PORT(C) ADDRESS ON THE 8255'S
                     106                                         ;FOR THE 16 BIT DECODER CONTROL BUS
                     107
                     108
004E                 109     DECODER_BUS_SPVR    EQU     4EH     ;THIS IS THE SUPERVISOR PORT(CONTROL PORT)
                     110                                         ;FOR THE DECODER BUS(THE CONTROL BUS AND
                     111                                         ;THE DATA BUS)
                     112
                     113
                     114
                     115
                     116
                     117
0064                 118     ERROR_LED           EQU     64H     ;
                     119
0066                 120     ERROR_BUS_SPVR      EQU     66H     ;
                     121
                     122
                     123         ;8259A PIC DEPENDENT CONSTANTS
                     124
0078                 125     PIC_PORT_0          EQU     78H     ;8259A PROGRAMMABLE INTERRUPT CONTROLLER
007A                 126     PIC_PORT_1          EQU     7AH     ;CONTROL PORTS.
                     127
```

```
LOC  OBJ                  LINE     SOURCE

 0013                      128     ICW1                  EQU     13H      ;SINGLE PIC, ICW4 NEEDED
 0008                      129     ICW2                  EQU     08H      ;STARTING INTERRUPT VECTOR = 8
 0003                      130     ICW4                  EQU     03H      ;SFNM = 0, AEOI = 1, NON-BUFFERED MODE
                           131                                            ;SP/EN WILL HAVE INPUT = 1 =>MASTER
                           132                                            ;UPM = 1 =>8086/88 SYSTEM.
 00FE                      133     ENABLE_SOF_MASK       EQU     0FEH     ;MASK OUT ALL EXCEPT
                           134                                            ;SOF INTERRUPT(R0)
                           135
 00E0                      136     ENABLE_NORMAL_MASK    EQU     0E0H     ;MASK OUT UNUSED
                           137                                            ;INTERRUPT(R7,R6,R5)
                           138
                           139
                           140
                           141         ;8255A PPI DEPENTENT CONTSTANTS
                           142
 9B9B                      143     C_DATA_BUS_IN         EQU     9B9BH    ;PROGRAM THE CODER DATA BUS TO BE INPUT, ALL
                           144                                            ;OTHER PORTS ON CODER BUS ARE INPUT.
                           145                                            ;MSB AND LSB OF THIS WORD EACH ADDRESSES TO O
                                   NE PPI
                           146
 9999                      147     C_DATA_BUS_OUT        EQU     9999H    ;PROGRAM THE CODER DATA BUS TO BE OUTPUT(IE.
                           148                                            ;TAKE CONTROL OF THE DATA BUS), ALL OTHER
                           149                                            ;PORTS ON THE CODER BUS ARE INPUT. MSB
                           150                                            ;AND LSB OF THIS WORD EACH ADDRESSES ONE PPI.
                           151
 9B9B                      152     D_DATA_BUS_IN         EQU     9B9BH    ;PROGRAM THE DECODER DATA BUS TO BE INPUT, AL
                                   L
                           153                                            ;OTHER PORTS ON DECODER BUS ARE INPUT.
                           154                                            ;MSB AND LSB OF THIS WORD EACH ADDRESSES TO O
                                   NE PPI
                           155
 9999                      156     D_DATA_BUS_OUT        EQU     9999H    ;PROGRAM THE DECODER DATA BUS TO BE OUTPUT(IE
                                   .
                           157                                            ;TAKE CONTROL OF THE DATA BUS), ALL OTHER
                           158                                            ;PORTS ON THE DECODER BUS ARE INPUT. MSB
                           159                                            ;AND LSB OF THIS WORD EACH ADDRESSES ONE PPI.
                           160
 0092                      161     ERROR_LED_ON          EQU     0092H    ;PROGRAM THE ERROR BUS TO BE:
                           162                                            ;  PORT A - INPUT } NOT USED HERE
                           163                                            ;  PORT B - INPUT } NOT USED HERE
                           164                                            ;  PORT C - OUTPUT } TO ERROR_LED
                           165
 009B                      166     ERROR_LED_OFF         EQU     009BH    ;PROGRAM THE ERROR BUS TO BE:
                           167                                            ;  PORT A - INPUT } NOT USED HERE
                           168                                            ;  PORT B - INPUT } NOT USED HERE
                           169                                            ;  PORT C - INPUT } TO ERROR_LED
                           170                                            ;THE LED DISPLAY IN THIS STATE WILL
                           171                                            ;BE 'FF'
                           172
                           173
                           174         ;CONTROL BUS DEPENDENT CONSTANTS
                           175
                           176         ;DECODER BUS
                           177         ;-----------
                           178
 OBJ                 LINE    SOURCE

0004                  179    SPEN                  EQU     0004H   ;MASK FOR SPEN SIGNAL(LOW TRUE)
0010                  180    SFEEN                 EQU     0010H   ;MASK FOR SFEEN SIGNAL(LOW TRUE)
                      181
                      182
                      183        ;CODER BUS
                      184        ;---------
                      185
0800                  186    CFECLK                EQU     0800H   ;MASK FOR CFECLK SIGNAL(HIGH TRUE)
0020                  187    CPCLK                 EQU     0020H   ;MASK FOR CPCLK SIGNAL(HIGH TRUE)
                      188
                      189
                      190
                      191    ;MDA IN HOUSE MONITOR ENTRY POINT
                      192
0010                  193    MONT_86_IP            EQU     0010H   ;OFFSET ADDRESS
FE9F                  194    MONT_86_CS            EQU     0FE9FH  ;SEGMENT ADDRESS
                      195
                      196 +1 $EJECT
```

```
LOC  OBJ          LINE     SOURCE

                   197
                   198
----               199     INTERRUPT_VECTOR        SEGMENT         WORD AT 0H
                   200
                   201
                   202
                   203     ;               ******************************
                   204     ;               *  INTERRUPT VECTOR TABLE   *
                   205     ;               ******************************
                   206
                   207     ;INTEL RESERVES INT 5 TO 32 FOR INTERNAL USES. CURRENT IMPLEMENTATION
                   208     ;                   VIOLATES THIS RESTRICTION.
                   209
                   210
                   211     ;8086 PREDIFINED INTERRUPTS: (INT 0 TO 4)
                   212
0000 (1            213         DIVIDE_INT_IP      DW  1 DUP(?)
     ????
     )
0002 (1            214         DIVIDE_INT_CS      DW  1 DUP(?)
     ????
     )
0004 (1            215         SINGLE_STEP_IP     DW  1 DUP(?)
     ????
     )
0006 (1            216         SINGLE_STEP_CS     DW  1 DUP(?)
     ????
     )
0008 (1            217         NMI_IP             DW  1 DUP(?)
     ????
     )
000A (1            218         NMI_CS             DW  1 DUP(?)
     ????
     )
000C (1            219         BREAKPOINT_IP      DW  1 DUP(?)
     ????
     )
000E (1            220         BREAKPOINT_CS      DW  1 DUP(?)
     ????
     )
0010 (1            221         OVERFLOW_IP        DW  1 DUP(?)
     ????
     )
0012 (1            222         OVERFLOW_CS        DW  1 DUP(?)
     ????
     )
                   223
                   224
                   225     ;MULDEM APPLICATION INTERRUPTS: (INT 8 TO 15)
                   226
0020               227          ORG        20H
0020 (1            228          INT_8_IP       DW  1 DUP(?)
     ????
     )
0022 (1            229          INT_8_CS       DW  1 DUP(?)
LOC  OBJ                  LINE     SOURCE

     ????
     )
0024 (1                    230          INT_9_IP       DW  1 DUP(?)
     ????
     )
0026 (1                    231          INT_9_CS       DW  1 DUP(?)
     ????
     )
0028 (1                    232          INT_10_IP      DW  1 DUP(?)
     ????
     )
002A (1                    233          INT_10_CS      DW  1 DUP(?)
     ????
     )
002C (1                    234          INT_11_IP      DW  1 DUP(?)
     ????
     )
002E (1                    235          INT_11_CS      DW  1 DUP(?)
     ????
     )
0030 (1                    236          INT_12_IP      DW  1 DUP(?)
     ????
     )
0032 (1                    237          INT_12_CS      DW  1 DUP(?)
     ????
     )
0034 (1                    238          INT_13_IP      DW  1 DUP(?)
     ????
     )
0036 (1                    239          INT_13_CS      DW  1 DUP(?)
     ????
     )
0038 (1                    240          INT_14_IP      DW  1 DUP(?)
     ????
     )
003A (1                    241          INT_14_CS      DW  1 DUP(?)
     ????
     )
003C (1                    242          INT_15_IP      DW  1 DUP(?)
     ????
     )
003E (1                    243          INT_15_CS      DW  1 DUP(?)
     ????
     )
                           244
                           245
----                       246     INTERRUPT_VECTOR         ENDS
                           247
                           248
                           249
                           250 +1  $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          251
                          252
----                      253     DATA    SEGMENT
                          254
                          255          ;THIS SEGMENT WILL RESIDE IN RAM
                          256
0000 (24                  257          PREDICTOR_BUFFER   DW      (((LAST_P_BUFFER_FRAME/16)+1) * LEN_P_BUFFER_FRAME) D
                                  UP(?)
     ????
     )
0030 (32                  258          FE_BUFFER          DW      (LEN_FE_BUFFER) DUP(?)
     ????
     )
                          259
0070 (1                   260          LED_DISPLAY_VALUE  DW      1 DUP(?)
     ????
     )
                          261
0072 (1                   262          ABS_XIN            DW      1 DUP(?)
     ????
     )
0074 (1                   263          XOUT               DW      1 DUP(?)
     ????
     )
                          264
0076 (1                   265          FSV                DW      1 DUP(?)
     ????
     )
0078 (1                   266          FSVL               DW      1 DUP(?)
     ????
     )
007A (1                   267          FSVH               DW      1 DUP(?)
     ????
     )
                          268
007C (1                   269          I                  DW      1 DUP(?)
     ????
     )
007E (1                   270          IL                 DW      1 DUP(?)
     ????
     )
0080 (1                   271          L2                 DW      1 DUP(?)
     ????
     )
                          272
0082 (1                   273          FE_SERVICE_COUNTER DW      1 DUP(?)
     ????
     )
                          274
0084 (1                   275          FE_SERVICE_PTR     DW      1 DUP(?)
     ????
     )
                          276
0086 (1                   277          FE_TYPE            DW      1 DUP(?)
     ????
LOC  OBJ                 LINE     SOURCE

       )
                          278
0088 (1                   279          SIGN_FLAG          DW       1 DUP(?)
       ????
       )
                          280
                          281
----                      282     DATA    ENDS
                          283
                          284
                          285
                          286
                          287
                          288
                          289
                          290
                          291
----                      292     STACK                   SEGMENT
                          293
                          294             ;**********************************************
                          295             ;* THE STACK IS SOLELY USED BY THE 8086 TO   *
                          296             ;* STORE RETURN ADDRESS IN INTERRUPT ROUTINES*
                          297             ;**********************************************
0000 (60                  298                             DW       60 DUP (?)
       ????
       )
0078                      299     TOS                     LABEL    WORD
                          300
----                      301     STACK                   ENDS
                          302
                          303 +1  $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          304
                          305
                          306
                          307     ;SYSTEM MACROS:
                          308     ;--------------
                          309
                          310
                          311     ;THE MACROS ARE NOT LISTED IN THE ASSEMBLER GENERATED LISTING, REFER
                          312     ;TO THE SOURCE FILE FOR MACRO CONTENTS.
                          313
                          314
                          315
                          316
                          317
                          318
                          319
                          320
                          321
                          322
                          323
                          324
                          325
                          326
                          327
                          328
                          329
                          330
                          331 +1  $EJECT
```

```



          ;*****************************************************
          ;*                                                   *
          ;*   REGISTER USAGE IN MQLDEM SIMULATOR SYSTEM :     *
          ;*                                                   *
          ;*****************************************************


;DEDICATED USAGE FOR ALL PARTS OF THE SYSTEM AT ALL TIME:

          ;SI     - FE_BUFFER INPUT POINTER
          ;BX     - FE_BUFFER OUTPUT POINTER [THIS REGISTER IS ALSO USED UNDER
          ;          NON-INTERRUPT DRIVEN QUANTIZATION PROCESS, HOWEVER, THE
          ;          ORIGINAL REGISTER IS PRESERVED]
          ;BP     - PREDICTOR_BUFFER FRAME POINTER
          ;DI     - PREDICTOR_BUFFER OFFSET POINTER


          ;CS     - CODE SEGMENT
          ;DS     - DATA SEGMENT
          ;ES     - DATA SEGMENT OR INTERRUPT_VECTOR SEGMENT
          ;SS/SP  - STACK OPERATION


;UNASSIGNED REGISTERS:

          ;THESE REGISTERS DO NOT CARRY DEDICATED FUNCTIONS IN THE MQLDEM
          ;SIMULATOR:

          ;AX, CX, DX

$EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           366
                           367
                           368            PUBLIC         START_ADDR
                           369
                           370
----                       371     CODE                  SEGMENT
                           372
                           373            ASSUME         CS:CODE, DS:DATA, SS:STACK, ES:INTERRUPT_VECTOR
                           374
                           375
                           376
                           377            ;**********************************
                           378            ;*    STATIC    VARIABLES         *
                           379            ;**********************************
                           380
                           381
                           382                   ;YQ  -   ARRAY OF NLEV NORMALIZED QUANTIZER OUTPUT VALUES
                           383                   ;        (IN INCREASING ORDER)
                           384                   ;        VALUE SCALED BY ** IN ** REPRESENTATION
0000 0000                  385            YQ             DW     0, 8192,  24576
0002 0020
0004 0060
                           386
                           387                   ;XQ  -   ARRAY OF NLEV-1 NORMALIZED QUANTIZER BREAK POINTS
                           388                   ;        (IN INCREASING ORDER)
                           389                   ;        VALUE SCALED BY ** IN ** REPRESENTATION
0006 0000                  390            XQ             DW     0, 16384
0008 0040
                           391
                           392                   ;QMLT   - ARRAY OF NLEV, QUANTIZER MULTIPLIERS
                           393                   ;        VALUE SCALED BY ** IN ** REPRESENTATION
000A 0000                  394            QMLT           DW     0, 27853, 62259
000C CD6C
000E 33F3
                           395
                           396
                           397 +1  $EJECT
```

```
LOC  OBJ                LINE    SOURCE

                         398
                         399    ;       ******************************************************************
                         400    ;   *                    M A I N    P R O G R A M                        *
                         401    ;       ******************************************************************
                         402
                         403
                         404    ;               *****************************************************
                         405    ;               *                                                   *
                         406    ;               *         REGISTER VALUES ARE NOT PRESERVED         *
                         407    ;               *         IN ALL PROCEDURES. THEY SHOULD BE         *
                         408    ;               *             SAVED BEFORE ENTERING.                *
                         409    ;               *                                                   *
                         410    ;               *****************************************************
                         411
                         412
0010 FA                  413    START_ADDR:     CLI                             ;DISABLE EXTERNAL INTERRUPT
                         414
                         415
                         416    ;               *********************************************
                         417    ;               * INITIALIZE THE INTERRUPT VECTOR  TABLE    *
                         418    ;               *                                           *
                         419    ;               * THIS SUBSYSTEM USES FOLLOWING SIGNALS FROM *
                         420    ;               * THE SYSTEM BUS:                           *
                         421    ;               *                                           *
                         422    ;               *     - START OF FRAME (SOF_)               *
                         423    ;               *     - FE ENABLE (SFEEN_)                  *
                         424    ;               *     - P ENABLE (SPEN_)                    *
                         425    ;               *     - FE CLOCK (CFECLK)                   *
                         426    ;               *     - P CLOCK (CPCLK)                     *
                         427    ;               *********************************************
                         428
                         429
                         430
0011 B8----         R    431                    MOV     AX, DATA        ;CANNOT MOVE IMMED. VALUE TO
0014 8ED8                432                    MOV     DS, AX          ;SEGMENT REGISTER.
                         433
0016 8EC0                434                    MOV     ES, AX          ;ES AND DS ARE REFERING TO SAME
                         435                                            ;SEGMENT
                         436
                         437                    ;RESET THE LED_DISPLAY_VALUE
0018 C70670000000        438                    MOV     LED_DISPLAY_VALUE, 0
                         439
001E B89200              440                    MOV     AX, ERROR_LED_ON
0021 E666                441                    OUT     ERROR_BUS_SPVR, AL
                         442
                         443                    ;OUTPUT ERROR DISPLAY VALUE
                         444
0023 B80000              445                    MOV     AX, 0
0026 E664                446                    OUT     ERROR_LED, AL
                         447
0028 B80000              448    ERROR_ENTRY:    MOV     AX, INTERRUPT_VECTOR
002B 8EC0                449                    MOV     ES, AX                  ;USE EXTRA SEGMENT TO ADDRESS
                         450                                                    ;THE INTERRUPT VECTOR TABLE
                         451
002D 26C70608001000      452                    MOV     NMI_IP, MONT_86_IP
```

```
LOC  OBJ                  LINE     SOURCE

0034 26C7060A009FFE        453                   MOV    NMI_CS, MONT_86_CS
                           454
003B B80301                455                   MOV    AX, OFFSET EXCEPTION_INT
003E BB----90         R    456                   MOV    BX, SEG EXCEPTION_INT
0042 26A30000              457                   MOV    DIVIDE_INT_IP, AX
0046 26891E0200            458                   MOV    DIVIDE_INT_CS, BX
004B 26A31000              459                   MOV    OVERFLOW_IP, AX
004F 26891E1200            460                   MOV    OVERFLOW_CS, BX
                           461
                           462
0054 26C70620000000        463                   MOV    INT_8_IP, OFFSET SOF_INTERRUPT
005B 26C7062200----   R    464                   MOV    INT_8_CS, SEG SOF_INTERRUPT
                           465
0062 26C70624005700        466                   MOV    INT_9_IP, OFFSET PREDICTOR_OUTPUT_INTERRUPT
0069 26C7062600----   R    467                   MOV    INT_9_CS, SEG PREDICTOR_OUTPUT_INTERRUPT
                           468
0070 26C70628008E00        469                   MOV    INT_10_IP, OFFSET FE_OUTPUT_INTERRUPT
0077 26C7062A00----   R    470                   MOV    INT_10_CS, SEG FE_OUTPUT_INTERRUPT
                           471
007E 26C7062C00E400        472                   MOV    INT_11_IP, OFFSET PREDICTOR_INPUT_INTERRUPT
0085 26C7062E00----   R    473                   MOV    INT_11_CS, SEG PREDICTOR_INPUT_INTERRUPT
                           474
008C 26C70630008900        475                   MOV    INT_12_IP, OFFSET FE_INPUT_INTERRUPT
0093 26C7063200----   R    476                   MOV    INT_12_CS, SEG FE_INPUT_INTERRUPT
                           477 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           478
                           479
                           480            ASSUME ES:DATA
                           481
                           482
                           483
                           484            ;INITIALIZE 8086 PROCESSOR ENVIRONMENT
                           485
                           486
009A B8----          R     487                  MOV     AX, DATA        ;CANNOT MOVE IMMED. VALUE TO
009D 8ED8                  488                  MOV     DS, AX          ;SEGMENT REGISTER.
                           489
009F 8EC0                  490                  MOV     ES, AX          ;ES AND DS ARE REFERING TO SAME
                           491                                          ;SEGMENT
00A1 B8----          R     492                  MOV     AX, STACK
00A4 8ED0                  493                  MOV     SS, AX
00A6 BC7B00                494                  MOV     SP, OFFSET TOS
                           495
                           496
                           497            ;INITIALIZE ALL SYSTEM HARDWARE
                           498
                           499
                           500
                           501            ;       *******************
                           502            ;       *   P I C         *
                           503            ;       *******************
                           504
                           505
                           506            ;CAUTION:  AUTOMATIC EOI IS USED HERE. REFER TO INTEL APPLICATION
                           507            ;          NOTE AP-59 "USING THE 8259A PROGRAMMABLE INTERRUPT
                           508            ;          CONTROLLER', UNDER HEADING 'AUTOMATIC EOI MODE'
                           509
                           510
00A9 B013                  511                  MOV     AL, ICW1
00AB E67E                  512                  OUT     PIC_PORT_0, AL
                           513
00AD BA7A00                514                  MOV     DX, PIC_PORT_1
00B0 B008                  515                  MOV     AL, ICW2
00B2 EE                    516                  OUT     DX, AL
                           517
                           518                  ;ICW 3 IS NOT NEEDED FOR CURRENT HARDWARE CONFIGURATION
                           519
00B3 B003                  520                  MOV     AL, ICW4
00B5 EE                    521                  OUT     DX, AL
                           522
00B6 B0FE                  523                  MOV     AL, ENABLE_SOF_MASK
00B6 EE                    524                  OUT     DX, AL
                           525
                           526
                           527            ;       *******************
                           528            ;       *   P P I         *
                           529            ;       *******************
                           530
                           531
                           532            ;INITIALLY ALL PORT ARE PROGRAMMED TO BE INPUT PORTS IN MODE 0
```

```
LOC  OBJ                  LINE     SOURCE

                           533            ;EXCEPT THE ERROR LED PORT, WHICH WILL BE OUTPUT ALL THE TIME
                           534
00B9 B89B9E                535                    MOV     AX, C_DATA_BUS_IN
00BC E746                  536                    OUT     CODER_BUS_SPVR, AX
                           537
                           538
00BE B89B9B                539                    MOV     AX, D_DATA_BUS_IN
00C1 E74E                  540                    OUT     DECODER_BUS_SPVR, AX
                           541 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         542
                         543            ;***********************************************
                         544            ;*  INITIALIZE APPLICATION PROGRAM ENVIRONMENT *
                         545            ;***********************************************
                         546
                         547
                         548
                         549
                         550
                         551                     ; FILL BUFFER AREAS WITH ZEROS
                         552
00C3 B80000              553                     MOV    AX, 0                          ;FILL VALUE
00C6 B91800              554                     MOV    CX, LENGTH PREDICTOR_BUFFER    ;ITERATION COUNT
00C9 BF0000              555                     MOV    DI, OFFSET PREDICTOR_BUFFER
                         556
                         557                     ;ES SETS TO THE SEGMENT BASE OF DATA SEGMENT
                         558
00CC F3                  559            REP      STOS   PREDICTOR_BUFFER
00CD AB
                         560
                         561
00CE B92000              562                     MOV    CX, LENGTH FE_BUFFER           ;ITERATION COUNT
00D1 BF3000              563                     MOV    DI, OFFSET FE_BUFFER
                         564
00D4 F3                  565            REP      STOS   FE_BUFFER
00D5 AB
                         566
                         567
                         568                     ;RESET SERVICE COUNTER
00D6 C70682000000        569                     MOV    FE_SERVICE_COUNTER, 0
                         570
                         571                     ;INITIALIZE FE TYPE TO LOW
00DC C70656000000        572                     MOV    FE_TYPE, FE_LOW
                         573
                         574
                         575                     ;INITIALIZE BUFFER POINTERS TO APPROPRIATE VALUES
                         576
                         577                     ;PREDICTOR BUFFER , BP IS THE FRAME POINTER,
                         578                     ;DI IS THE OFFSET POINTER WITHIN A FRAME.
                         579
00E2 BD2000              580                     MOV    BP, LAST_P_BUFFER_FRAME        ;ON FIRST SOF INTERRUPT AFTER
                         581                                                           ;POWER UP, SOF_INTERRUPT ROUT
                                 INE
                         582                                                           ;WILL SET BP = 0, DI = 0
                         583
00E5 BF1000              584                     MOV    DI, LEN_P_BUFFER_FRAME * (TYPE PREDICTOR_BUFFER)
                         585
                         586                     ;FE BUFFER INITIALIZATION
                         587
00E8 BE1000              588                     MOV    SI, 16                         ;POINTS TO FEL(4)
00EB BB0000              589                     MOV    BX, 0
                         590
                         591                     ;QUANTIZATION SERVICE
                         592
00EE C70684001000        593                     MOV    FE_SERVICE_PTR, 16
```

```
LOC  OBJ                  LINE     SOURCE

00F4 C7067B000A00          594                 MOV     FSVL, FSVMN
00FA C7067A000A00          595                 MOV     FSVH, FSVMN
                           596
                           597
                           598
0100 FB                    599                 STI                                     ;ENABLE EXTERNAL INTERRUPT
                           600
0101 F4                    601                 HLT                                     ;WAIT UNTIL THE FIRST
                           602                                                         ;START OF FRAME INTERRUPT
                           603
                           604 +1  $EJECT
```

```
LOC  OBJ           LINE    SOURCE



                                   ;*********************************************
                                   ;*                                           *
                                   ;*    START   OF   APPLICATION   PROGRAM     *
                                   ;*                                           *
                                   ;* AFTER POWER UP, THE FOLLOWING APPLICATION *
                                   ;* PROGRAM WILL NOT START EXECUTION UNTIL THE *
                                   ;* FIRST SOF_INTERRUPT HAS BEEN SERVICED.    *
                                   ;*********************************************


0102                       LABEL_WAIT_FOR_NEW_FE:

                                          ;LOOP AROUND UNTIL THERE IS A NEW FE TO SERVICE

0102 39360400                             CMP     FE_SERVICE_PTR, SI
0106 74FA                                 JE      LABEL_WAIT_FOR_NEW_FE

                                          ;TWO POINTERS ARE NOT EQUAL, POSSIBLE NEW FE'S


                                          ;CHECK SERVICE COUNTER
0108 833E820000                           CMP     FE_SERVICE_COUNTER, 0
010D 744C                                 JE      LABEL_PTR_ERROR         ;THE SERVICE PTR AND
                                                                          ;INPUT POINTER(SI) ARE NOT
                                                                          ;EQUAL, SERVICE COUNTER = 0
                                                                          ;SYNC ERROR


                                          ;SERVICE COUNTER >=1 , NORMAL CONDITION

010F 833E820008                           CMP     FE_SERVICE_COUNTER, 8
0114 7345                                 JAE     LABEL_PTR_ERROR         ;THERE ARE MORE THAN 4 PAIRS
                                                                          ;OF FEX TO BE SERVICED. CPU
                                                                          ;IS RUNNING TOO SLOW

                                          ;1 =< SERVICE COUNTER < 8, NORMAL CONDITION.
                                          ; BEGIN SERVICE THE FE'S



                         +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         648
                         649
0116                     650     LABEL_SERVICE_FE:
                         651                             ;MOVE THE NEW FEX INTO AX AND CORRESPONDING FSV TO CX
                         652
                         653                             ;NEED TO USE THE BX REGISTER TO ACCESS THE FE_BUFFER
0116 8B0E8400            654                             MOV     CX, FE_SERVICE_PTR
                         655
011A FA                  656                             CLI
011B 87CB                657                             XCHG    CX, BX
011D 8B4730              658                             MOV     AX, FE_BUFFER[BX]
0120 87CB                659                             XCHG    CX, BX
0122 FB                  660                             STI
                         661
0123 833E860000          662                             CMP     FE_TYPE, FE_LOW
0128 7547                663                             JNE     LABEL_FE_HIGH
                         664
                         665
                         666
                         667
                         668                             ;IT IS FEL TO BE PROCESSED
012A 8B0E7800            669                             MOV     CX, FSVL
012E E87600              670                             CALL    APCMQ
                         671
                         672                             ;RESULT FSV IN DX, XOUT IN CX
                         673
0131 89167800            674                             MOV     FSVL, DX                ;SAVE THE NEW FSVL
                         675
0135 A19400              676                             MOV     AX, FE_SERVICE_PTR      ;FE_SERVICE_PTR WILL BE USED LATER
                         677
                         678                             ;IF XOUT IS ZERO THEN, STORE THE NEW XOUT INTO THE FE_BUFFER
                         679                             ;BACKGROUND NOISE LEVEL IS SUPPRESSED THROUGH THIS MACHANISM
                         680                             ;IF XOUT IS NON-ZERO , THE ORIGINAL VALUE IS PRESERVED. NO
                         681                             ;QUANTIZATION IS DONE TO THE FEL ITSELF.
                         682
0138 83F900              683                             CMP     CX, 0                   ;CX CONTAINS THE XOUT VALUE
013B 7507                684                             JNE     LABEL_NO_FEL_UPDATE
                         685
                         686                             ;UPDATE THE FEL, XOUT = 0 IN CX
013D FA                  687                             CLI
013E 93                  688                             XCHG    AX, BX
013F 894F30              689                             MOV     FE_BUFFER[BX], CX
0142 93                  690                             XCHG    AX, BX
0143 FB                  691                             STI
                         692
0144                     693     LABEL_NO_FEL_UPDATE:
0144 FF0E8200            694                             DEC     FE_SERVICE_COUNTER
                         695
                         696                             ;UPDATE THE FE TYPE TO BE SERVICED NEXT
0148 C70686000100        697                             MOV     FE_TYPE, FE_HIGH
                         698
                         699
                         700                             ;INCREMENT THE SERVICE POINTER, AX CONTAINS FE_SERVICE_PTR
014E 050200              701                             ADD     AX, TYPE FE_BUFFER
0151 3D4000              702                             CMP     AX, LEN_FE_BUFFER * TYPE FE_BUFFER      ;END OF BUFFER?
```

```
LOC  OBJ                  LINE     SOURCE

0154 7348                  703                      JAE     LABEL_WRAP_AROUND
0156 A33400                704                      MOV     FE_SERVICE_PTR, AX
0159 EBA7                  705                      JMP     LABEL_WAIT_FOR_NEW_FE
                           706
                           707
                           708 +1  $EJECT
```

```
LOC  OBJ              LINE     SOURCE

                       709
                       710
                       711     ;*******************************************************************************
                       712
                       713     ;THIS LABEL_PTR_ERROR HANDLER IS POSITIONED HERE SO THAT IT IS WITHIN
                       714     ;127 BYTES FROM THE ORINGIN OF JUMP(OR RELATED) INSTRUCTIONS
                       715
015B                   716     LABEL_PTR_ERROR:
                       717
                       718             ;*************************************************************
                       719             ;* THIS PART OF CODE HANDLES THE SYNCHRONIZATION ERROR        *
                       720             ;* BETWEEN THE QUANTIZATION PROCESS AND THE VARIOUS           *
                       721             ;* INTERRUPT DRIVEN I/O PROCESSES.                            *
                       722             ;*                                                            *
                       723             ;* AN ERROR CONDITION MAY BE ONE OR MORE OF THE FOLLOWINGS:   *
                       724             ;*                                                            *
                       725             ;* (1) BOTH INPUT(REGISTER SI - DEDICATED) AND SERVICE POINTER *
                       726             ;*     ARE POINTING TO DIFFERENT ELEMENT IN THE FE_BUFFER     *
                       727             ;*     AND THE SERVICE COUNTER HAS A VALUE OF ZERO.           *
                       728             ;* (2) THE SERVICE COUNTER HAS A VALUE HIGHER THAN 8. THIS    *
                       729             ;*     MEANS THAT FOUR PAIRS OF FEX OR MORE ARE NOT QUANTIZED *
                       730             ;*     YET.                                                   *
                       731             ;* THE MOLDEM SIMULATOR PROCESS WILL CONTINUE AFTER THE       *
                       732             ;* ABOVE ERROR(S) OCCURS. THE FOLLOWING ROUTINE WILL CAUSE THE *
                       733             ;* MOST SIGNIFICANT ERROR_LED TO INCREMENT.                   *
                       734             ;*************************************************************
                       735
                       736
015B FA                737                     CLI                          ;DISABLE EXTERNAL INTERRUPT
                       738
015C A17000            739                     MOV     AX, LED_DISPLAY_VALUE
015F 80C410            740                     ADD     AH, 10H
0162 A37000            741                     MOV     LED_DISPLAY_VALUE, AX
0165 0AC4              742                     OR      AL, AH
0167 E664              743                     OUT     ERROR_LED, AL
0169 E9BCFE            744                     JMP     ERROR_ENTRY
                       745
                       746
                       747
                       748      ;*****************************************************************************
                       749
                       750 +1  $EJECT
```

```
LOC  OBJ                 LINE     SOURCE

                          751
016C                      752     LABEL_FE_HIGH:  ;THE FEH IS TO BE PROCESSED.
016C 8B0E7A00             753                     MOV     CX, FSVH
0170 E83400               754                     CALL    APCMQ
                          755
                          756                     ;RESULT FSV IN DX, XOUT IN CX
                          757
0173 89167A00             758                     MOV     FSVH, DX                ;SAVE THE NEW FSVH
                          759
0177 A1E400               760                     MOV     AX, FE_SERVICE_PTR      ;FE_SERVICE_PTR WILL BE USED LATER
                          761
                          762
                          763                     ;IF XOUT IS ZERO THEN, STORE THE NEW XOUT INTO THE FE_BUFFER
                          764                     ;BACKGROUND NOISE LEVEL IS SUPPRESSED THROUGH THIS MACHANISM
                          765                     ;IF XOUT IS NON-ZERO , THE ORIGINAL VALUE IS PRESERVED. NO
                          766                     ;QUANTIZATION IS DONE TO THE FEH ITSELF.
                          767
017A 83F900               768                     CMP     CX, 0                   ;CX CONTAINS XOUT
017D 7507                 769                     JNE     LABEL_NO_FEH_UPDATE
                          770
                          771                     ;UPDATE THE FEH , XOUT = 0 IN CX
017F FA                   772                     CLI
0180 93                   773                     XCHG    AX, BX
0181 894F3C               774                     MOV     FE_BUFFER[BX], CX
0184 93                   775                     XCHG    AX, BX
0185 FB                   776                     STI
                          777
0186                      778     LABEL_NO_FEH_UPDATE:
0186 FF0E8200             779                     DEC     FE_SERVICE_COUNTER
                          780
                          781                     ;UPDATE THE FE TYPE TO BE SERVICED NEXT
018A C70686000000         782                     MOV     FE_TYPE, FE_LOW
                          783
                          784
                          785                     ;INCREMENT THE SERVICE POINTER
0190 050200               786                     ADD     AX, TYPE FE_BUFFER
0193 3D4000               787                     CMP     AX, LEN_FE_BUFFER * TYPE FE_BUFFER      ;END OF BUFFER?
0196 7306                 788                     JAE     LABEL_WRAP_AROUND
0198 A3E400               789                     MOV     FE_SERVICE_PTR, AX
019B E964FF               790                     JMP     LABEL_WAIT_FOR_NEW_FE
                          791
                          792 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           793
                           794
                           795
                           796
019E                       797     LABEL_WRAP_AROUND:
                           798                         ;SERVICE POINTER WAS POINTING TO THE LAST ELEMENT IN FE
                           799                         ;ARRAY, MOVE TO THE FIRST ELEMENT.
                           800
019E C70684000000          801                         MOV     FE_SERVICE_PTR, 0
01A4 E95BFF                802                         JMP     LABEL_WAIT_FOR_NEW_FE
                           803
                           804
                           805
                           806                         ;***********************************
                           807                         ;*     END  OF  MAIN  PROGRAM      *
                           808                         ;***********************************
                           809
                           810
                           811
                           812 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         813
                         814
01A7                     815     APCMQ          PROC          NEAR
                         816
                         817
                         818     ;***************************************************************************
                         819     ;* PURPOSE:                                                                *
                         820     ;*      THIS ROUTINE QUANTIZES A SAMPLE USING AN ADAPIVE QUANTIZER.        *
                         821     ;*                                                                         *
                         822     ;*                                                                         *
                         823     ;* DESCRIPTION:                                                            *
                         824     ;*      THE INPUT SAMPLE IS QUANTIZED, USING THE GIVEN SCALING FACTOR      *
                         825     ;*      FOR THE QUANTIZER. THE SCALING FACTOR IS UPDATED ON RETURN.        *
                         826     ;*                                                                         *
                         827     ;*                                                                         *
                         828     ;* PARAMETERS:                                                             *
                         829     ;*      INPUT :                                                            *
                         830     ;*        XIN   - INPUT SAMPLE(PASSED IN AX REGISTER)                      *
                         831     ;*        FSV   - SCALING FACTOR FOR THE QUANTIZER. THIS VALUE IS          *
                         832     ;*                UPDATED ON OUTPUT.(INPUTED IN CX REGISTER)               *
                         833     ;*                                                                         *
                         834     ;*        OUTPUT:                                                          *
                         835     ;*        XOUT  - OUTPUT QUANTIZED SAMPLE (PASSED IN CX REGISTER)          *
                         836     ;*        FSV   - NEW SCALING FACTOR FOR THE QUANTIZER (OUTPUT             *
                         837     ;*                 IN DX REGISTER)                                         *
                         838     ;*                                                                         *
                         839     ;*      PRE-SPECIFIED:                                                     *
                         840     ;*        NLEV  - NUMBER OF POSITIVE QUANTIZER LEVELS. THE QUANTIZER       *
                         841     ;*               IS ASSUMED TO BE SYMMETRIC ABOUT ZERO. THE TOTAL          *
                         842     ;*               NUMBER OF QUANTIZER LEVELS IS 2*NLEV.                     *
                         843     ;*                                                                         *
                         844     ;* ROUTINES REQUIRED:                                                      *
                         845     ;*     IQUANTZ - QUANTIZE A POINT                                          *
                         846     ;***************************************************************************
                         847
                         848 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         849
                         850
01A7 890E7600            851                 MOV    FSV, CX               ;FSV IS PASSED TO THIS PROCEDURE FROM
                         852                                              ;REGISTER CX. SAVE FOR FUTURE REFEREN
                                 CE
                         853                                              ;THIS VALUE IS ALSO USED IN PROCEDURE
                                  IQUANTZ
                         854
                         855                 ;XIN IS PASSED TO THIS PROCEDURE IN REGISTER AX.
                         856
01AB C70688000000        857                 MOV    SIGN_FLAG, POSITIVE   ;INITIALIZE THE SIGN FLAG
                         858
01B1 A9FFFF              859                 TEST   AX, OFFFFH            ;GET SIGN
01B4 7908                860                 JNS    LABEL_1               ;IT IS A POSITIVE NUMBER
                         861
                         862                 ;XIN IS A NEGATIVE NUMBER, GET ABSOLUTE VALUE
                         863
01B6 F7D8                864                 NEG    AX
01B8 C70688000100        865                 MOV    SIGN_FLAG, NEGATIVE   ;STORE STATE OF SIGN
                         866
01BE A37200              867     LABEL_1:    MOV    ABS_XIN, AX           ;SAVE THE ABSOLUTE XIN FOR
                         868                                              ;USE IN IQUANTZ PROCEDURE
                         869
                         870
                         871                 ;REGISTER AX, CX AND DX ARE NOT PRESERVED IN THIS CALL.
01C1 E86300              872                 CALL   IQUANTZ
                         873
                         874                 ;RESULT 'L2' RETURNED IN CX. THIS VALUE IS DOUBLED THE
                         875                 ;ACTUAL VALUE TO FACILITATE INDEX ADDRESSING.
                         876
01C4 890E8000            877                 MOV    L2, CX                ;SAVE FOR FUTURE REFENRECE
                         878
                         879 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                         880                      ;**********************************************
                         881                      ;* GENERATE QUANTIZED VALUE FOR THE CODED BITS *
                         882                      ;**********************************************
                         883
01C8 A17600              884                      MOV     AX, FSV
                         885
01CB 3D0F00              886                      CMP     AX, FSVTH
                         887
01CE 730E                888                      JAE     LABEL_MID_RISE
                         889
                         890                      ;FSV < FSVTH
                         891
                         892
01D0 83F902              893                      CMP     CX, 2                  ;L2 = 1?
                         894
01D3 7509                895                      JNE     LABEL_MID_RISE
                         896
                         897                      ;L2 = 1 , USE MID_TREAD QUANTIZER AND SET XOUT = 0
                         898
01D5 C70674000000        899                      MOV     XOUT, 0
01DB EB1B90              900                      JMP     INC_STEP
                         901
                         902
01DE                     903      LABEL_MID_RISE:
                         904
                         905                      ;NEED TO USE BX AT THIS POINT, DISABLE INTERRUPT BEFORE USE
                         906                      ;BX IS USED BY INTERRUPT I/O ROUTINES AS A DEDICATED REGISTER
                         907
01DE FA                  908                      CLI
01DF 87D9                909                      XCHG    BX, CX                 ;CX CONTAINS 'L2'
                         910
01E1 2EF727              911                      MUL     CS:YQ[BX]              ;AX CONTAINS FSV
                         912
01E4 87D9                913                      XCHG    BX, CX
                         914
01E6 FB                  915                      STI
                         916
01E7 D1C0                917                      ROL     AX, 1                  ;SCALE BY 2**15
01E9 D1D2                918                      RCL     DX, 1
                         919
                         920                      ;INCORPORATE SIGN BIT
01EB 833E880000          921                      CMP     SIGN_FLAG, 0
01F0 7402                922                      JZ      LABEL_2
                         923
                         924                      ;XIN WAS A NEGATIVE NUMBER
                         925
01F2 F7DA                926                      NEG     DX
                         927
01F4 89167400            928      LABEL_2:        MOV     XOUT, DX
                         929
                         930 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           931
                           932                       ;******************************
                           933                       ;* INCREMENT STEP SIZE OF FSV *
                           934                       ;******************************
                           935
                           936
01F8 8B0E8000              937     INC_STEP:         MOV     CX, L2
                           938
                           939                       ;NEED TO USE BX AGIAN, REFER TO PREVIOUS USAGE FOR DOCUMENTATION
                           940
01FC FA                    941                       CLI
01FD 87CB                  942                       XCHG    CX, BX
01FF 2E8B4709              943                       MOV     AX, CS:QMLT[BX]
0203 87CB                  944                       XCHG    CX, BX
0205 FB                    945                       STI
0206 F7267600              946                       MUL     FSV
                           947
020A D1C0                  948                       ROL     AX, 1                       ;SCALE RESULT BY 2**15
020C D1D2                  949                       RCL     DX, 1
                           950
020E 81FAE803              951                       CMP     DX, FSVMX
0212 7605                  952                       JBE     LABEL_3
0214 BAE803                953                       MOV     DX, FSVMX
0217 E90800                954                       JMP     APCMQ_END
021A 83FA0A                955     LABEL_3:          CMP     DX, FSVMN
021D 7303                  956                       JAE     APCMQ_END
021F BA0A00                957                       MOV     DX, FSVMN
                           958
                           959                       ;NEW FSV IN DX
                           960
0222 8B0E7400              961     APCMQ_END:        MOV     CX, XOUT
0226 C3                    962                       RET
                           963
                           964     APCMQ             ENDP
                           965
                           966
                           967
                           968 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE


0227                               IQUANTZ                PROC             NEAR

                                        ;*****************************************************************
                                        ;* INPUT :          ABS_XIN , FSV PASSED FROM MEMORY            *
                                        ;*                                                              *
                                        ;* OUTPUT:          L        IN REGISTER CX. THIS VALUE IS      *
                                        ;*                           DOUBLE THAT OF ACTUAL VALUE TO     *
                                        ;*                           FACILITATE INDEX ADDRESSING        *
                                        ;* REGISTER USAGE:                                              *
                                        ;*                  BX       TEMPORARILY, VALUE SAVED AND       *
                                        ;*                           RESTORED BY THIS PROCEDURE.        *
                                        ;*                           INTERRUPT DISABLED WHILE USING THIS*
                                        ;*                           REGISTER.                          *
                                        ;*                  DX                                          *
                                        ;*                                                              *
                                        ;* REGISTER ASSIGNMENT IN THIS PROCEDURE:                       *
                                        ;*                                                              *
                                        ;*                  CX       CONTAINS IU                        *
                                        ;*          REFER TO FORTRAN LISTING FOR DESCRIPTION OF THESE   *
                                        ;*          VARIABLES.                                          *
                                        ;*****************************************************************


                                  +1  $EJECT
```

```
LOC  OBJ                  SOURCE

0227 C7067E000000                      MOV     IL, 0                         ;IL = 0
022D B90400                            MOV     CX, NLEV* TYPE XQ             ;IU = NLEV [* 2 FACTOR ADDED]

                                       ;I = (IL + IU) /2

0230 8B167E00             LABEL_100:   MOV     DX, IL
0234 03D1                              ADD     DX, CX
0236 D1EA                              SHR     DX, 1                         ;DIVIDE BY 2
0238 81E2FEFF                          AND     DX, 0FFFEH                    ;MAKE SURE IT IS AN EVEN NUMB
                          ER
023C 89167C00                          MOV     I, DX                         ;SAVE I FOR FUTURE USE
                                       ;IF (X .GT. XQ(I)*FSV) GOTO 220

                                       ;NEED TO USE REGISTER BX AT THIS POINT. DISABLE INTERRUPT BEFORE
                                       ;USING


0240 FA                                CLI
0241 87DA                              XCHG    DX, BX
0243 2E8B4706                          MOV     AX, CS:XQ[BX]                 ;MOV XQ TO AX REGISTER
0247 87DA                              XCHG    DX, BX
0249 FB                                STI

024A F7267600                          MUL     FSV                           ;XQ IS SCALED BY 2**15
024E D1D0                              RCL     AX, 1
0250 D1D2                              RCL     DX, 1
0252 39167200                          CMP     ABS_XIN, DX
0256 7707                              JA      LABEL_200

                                       ;IU = I
0258 8B0E7C00                          MOV     CX, I
025C EB0990                            JMP     LABEL_300

025F                      LABEL_200:   ;IL = I
025F 8B167C00                          MOV     DX, I
0263 89167E00                          MOV     IL, DX


0267 8B167E00             LABEL_300:   MOV     DX, IL
026B 83C202                            ADD     DX, 2                         ;IL+ 1
026E 3BCA                              CMP     CX, DX
0270 77BE                              JA      LABEL_100

                                       ;RESULT IU IN CX REGISTER


0272 C3                                RET

                          IQUANTZ      ENDP
```

```
LOC  OBJ                  LINE     SOURCE

                          1048
                          1049
                          1050
----                      1051     CODE                ENDS
                          1052
                          1053
                          1054
                          1055 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                        1056
                        1057
                        1058
                        1059
----                    1060     INTERRUPT_CODE          SEGMENT
                        1061
                        1062             ASSUME          CS:INTERRUPT_CODE, DS:DATA, ES:NOTHING, SS:STACK
                        1063
0000                    1064     SOF_INTERRUPT           PROC            FAR
                        1065
                        1066     ;*****************************************************************************
                        1067     ;* THIS INTERRUPT ROUTINE IS ACTIVATED BY THE SOF_(START_OF_FRAME) SINGNAL *
                        1068     ;* ON THE CODER BUS(THE SOF_ ON THE DECODER BUS IS SYNCHRONIZED AND OCCURS *
                        1069     ;* AT THE SAME TIME AS THE DECODER BUS SOF_).                              *
                        1070     ;* THIS ROUTINE IS RESPONSIBLE FOR CHECKING THE MULDEM OVERALL FIRMWARE    *
                        1071     ;* STATE, INCLUDING FOLLOWING:                                             *
                        1072     ;*              - FE INPUT AND OUTPUT POINTER VALUES                       *
                        1073     ;*              - PREDICTOR COEFFICIENTS BUFFER FRAME AND OFFSET POINTER   *
                        1074     ;*                VALUES                                                   *
                        1075     ;*                                                                         *
                        1076     ;* IF ANY OF THESE VALUES IS ABNORMAL, THIS ROUTINE WILL ATTEMP TO CORRECT *
                        1077     ;* TO THE BEST OF ITS KNOWLEDGE. AT THE SAME TIME, IT WILL INCREMENT THE   *
                        1078     ;* ERROR LOG VALUE AND OUTPUT TO THE LED.                                  *
                        1079     ;* THIS ROUTINE IS ALSO RESPONSIBLE FOR SETTING UP THE PRIDICTION COEFF    *
                        1080     ;* BUFFER POINTERS FOR THE CURRENT FRAME.                                  *
                        1081     ;*                                                                         *
                        1082     ;*              REGISTER USAGES:                                           *
                        1083     ;*                      AX, BX, BP, SI, DI                                 *
                        1084     ;*                                                                         *
                        1085     ;*****************************************************************************
                        1086
                        1087
                        1088 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                        1089
                        1090                        ;NOTE THAT INTERRUPT FLAG (IF) IS DISABLED IN MOST PART
                        1091                        ;OF THIS ROUTINE.
                        1092
0000 50                 1093                                PUSH    AX                      ;MAY USE AX REGISTER TO
                        1094                                                                ;OUTPUT ERROR_LED IN THIS
                        1095                                                                ;ROUTINE
                        1096                        ;************************************
                        1097                        ;*    FE_BUFFER        CHECK        *
                        1098                        ;************************************
                        1099
                        1100
                        1101
                        1102                                ;CHECK FE INPUT POINTER
                        1103
0001 83FE10             1104                                CMP     SI, 16                  ;THIS IS THE NORMAL OFFSET
                        1105                                                                ;VALUE POINTING TO FEL(4)
0004 7411               1106                                JE      LABEL_NORMAL_1
                        1107
                        1108 +1
                        1109 +1
0006 A17000             1110 +1                             MOV     AX, LED_DISPLAY_VALUE
0009 FEC0               1111 +1                             INC     AL
000B 240F               1112 +1                             AND     AL, 0FH
                        1113
000D 7405               1114                                JZ      LED_ERROR_EXT1
                        1115
                        1116 +1
000F A37000             1117 +1                             MOV     LED_DISPLAY_VALUE, AX
0012 E664               1118 +1                             OUT     ERROR_LED, AL
                        1119
0014 BE1000             1120     LED_ERROR_EXT1:            MOV     SI, 16                  ;CORRECT POINTER VALUE
                        1121
0017                    1122     LABEL_NORMAL_1:            ;CHECK FE OUTPUT POINTER
                        1123
0017 83FB00             1124                                CMP     BX, 0                   ;THIS IS THE NORMAL VALUE
                        1125                                                                ;POINTING TO FEL(0)
001A 7411               1126                                JE      LABEL_NORMAL_2
                        1127
                        1128 +1
                        1129 +1
001C A17000             1130 +1                             MOV     AX, LED_DISPLAY_VALUE
001F FEC0               1131 +1                             INC     AL
0021 240F               1132 +1                             AND     AL, 0FH
                        1133
0023 7405               1134                                JZ      LED_ERROR_EXT2
                        1135
                        1136 +1
0025 A37000             1137 +1                             MOV     LED_DISPLAY_VALUE, AX
0028 E664               1138 +1                             OUT     ERROR_LED, AL
                        1139
002A BB0000             1140     LED_ERROR_EXT2:            MOV     BX, 0                   ;CORRECT POINTER VALUE
                        1141
                        1142                        ;************************************
                        1143                        ;*   PREDICTOR BUFFER CHECK         *
```

```
LOC  OBJ            LINE     SOURCE

                    1144                    ;*************************************
                    1145
002D                1146     LABEL_NORMAL_2:         ;CHECK PREDICTOR INPUT POINTER
                    1147
002D 83FF10         1148                     CMP     DI, LEN_P_BUFFER_FRAME * (TYPE PREDICTOR_BUFFER)
                    1149                                                     ;THIS IS THE NORMAL VALUE
                    1150                                                     ;POINTING TO ELEMENT AFTER
                    1151                                                     ;THE LAST ELEMENT OF A FRAME
                    1152
0030 740E           1153                     JE      LABEL_NORMAL_3
                    1154
                    1155                     ;THE POINTER CONTAINS ILLEGAL VALUE
                    1156
                    1157
                    1158 +1
                    1159 +1
0032 A17000         1160 +1                  MOV     AX, LED_DISPLAY_VALUE
0035 FEC0           1161 +1                  INC     AL
0037 240F           1162 +1                  AND     AL, 0FH
                    1163
0039 7405           1164                     JZ      LABEL_NORMAL_3
                    1165
                    1166 +1
003B A37000         1167 -1                  MOV     LED_DISPLAY_VALUE, AX
003E E614           1168 +1                  OUT     ERROR_LED, AL
                    1169
0040                1170     LABEL_NORMAL_3:         ;SET THE PREDICTOR POINTERS FOR NEXT FRAME OUTPUT
                    1171
0040 BF0000         1172                     MOV     DI, 0
0043 83C510         1173                     ADD     BP, LEN_P_BUFFER_FRAME *(TYPE PREDICTOR_BUFFER)
                    1174                                                     ;INCREMENT TO NEXT FRAME
                    1175
0046 83FD20         1176                     CMP     BP, LAST_P_BUFFER_FRAME ;DETERMINE IF BP POINTS TO
                    1177                                                     ;LAST FRAME IN BUFFER
                    1178
0049 7603           1179                     JBE     LABEL_SOF_END
                    1180
                    1181                     ;BP WAS POINTING TO LAST FRAME IN BUFFER BEFORE INCREMENT
                    1182
004B BD0000         1183                     MOV     BP, 0                   ;MOVE TO THE FIRST FRAME IN B
                             UFFER
                    1184
004E                1185 +1  LABEL_SOF_END:
                    1186 +1
                    1187 +1
                    1188 +1
004E FA             1189 +1                  CLI
004F B8E000         1190 +1                  MOV     AX, ENABLE_NORMAL_MASK
0052 E67A           1191 +1                  OUT     PIC_PORT_1, AL
0054 FB             1192 +1                  STI
0055 58             1193                     POP     AX
0056 CF             1194                     IRET
                    1195
                    1196
                    1197     SOF_INTERRUPT           ENDP
```

```
LOC  OBJ                  LINE     SOURCE

                          1198
                          1199
                          1200
                          1201 +1  $EJECT
```

```
LOC  OBJ               LINE     SOURCE

                       1202
0057                   1203     PREDICTOR_OUTPUT_INTERRUPT      PROC            FAR
                       1204
                       1205
                       1206     ;*************************************************************************
                       1207     ;* THIS PROCEDURE IS ACTIVATED BY THE SPEN_ SIGNAL ON THE DECODER BUS.     *
                       1208     ;* THIS ROUTINE IS RESPONSIBLE FOR OUTPUT THE PREDICTOR COEFFICIENTS FROM  *
                       1209     ;* THE PREDICTOR COEFF BUFFER.                                             *
                       1210     ;*                                                                         *
                       1211     ;*               REGISTER USAGES:                                          *
                       1212     ;*                       AX, DI                                            *
                       1213     ;*                                                                         *
                       1214     ;*************************************************************************
                       1215
                       1216
0057 50                1217                             PUSH    AX                      ;AX IS USED DURING THIS ROUTI
                                NE.
                       1218 +1
                       1219 +1
                       1220 +1
                       1221 +1
0058 B8FE00            1222 +1                          MOV     AX,ENABLE_SOF_MASK
005B E67A              1223 +1                          OUT     PIC_PORT_1, AL
005D FB                1224 +1                          STI
                       1225
                       1226
                       1227                             ;CONFIGURE THE DECODER DATA BUS TO BE OUTPUT
                       1228
005E B89999            1229                             MOV     AX, D_DATA_BUS_OUT
0061 E74E              1230                             OUT     DECODER_BUS_SPVR, AX
                       1231
                       1232                             ;OUTPUT DATA
                       1233
0063 3E8B03            1234                             MOV     AX, DS:PREDICTOR_BUFFER[BP][DI]
                       1235
0066 E74A              1236                             OUT     D_DATA_BUS, AX
                       1237
                       1238                             ;INCREMENT BUFFER POINTER
                       1239
0068 83C702            1240                             ADD     DI, TYPE PREDICTOR_BUFFER
006B 83FF10            1241                             CMP     DI, LEN_P_BUFFER_FRAME * TYPE PREDICTOR_BUFFER
006E 7203              1242                             JB      LABEL_P_EN
                       1243
                       1244                             ;DI POINTS BEYOND END OF CURRENT FRAME, END OF OUTPUT
                       1245                             ;OF A FRAME(THE N - 2ND FRAME) PREDICTOR COEFFICIENTS.
                       1246                             ;SET UP THE POINTER FOR INPUT NEW PREDICTOR
                       1247                             ;COEFFICIENTS AT END OF CURRENT FRAME(FRAME N).
                       1248                             ;THE INPUT DATA WILL BE DEPOSITED INTO THE SAME FRAME
                       1249                             ;BUFFER, THEREFORE BP REGISTER NEEDS NOT BE CHANGED.
                       1250
0070 BF0000            1251                             MOV     DI, 0
                       1252
                       1253
0073                   1254     LABEL_P_EN:             ;CHECK FOR END OF OUTPUT CYCLE
0073 E540              1255                             IN      AX, D_CONTROL_BUS
```

```
LOC  OBJ                  LINE     SOURCE

0075 250400               1256                                  AND    AX, SPEN
0078 74F9                 1257                                  JZ     LABEL_P_EN
                          1258
                          1259                                  ;NOW SPEN IS FALSE(I.E. SIGNAL IS HIGH), TAKE DATA OFF THE BU
                                   S
                          1260
007A B89B9B               1261                                  MOV    AX, D_DATA_BUS_IN
007D E74E                 1262                                  OUT    DECODER_BUS_SPVR, AX
                          1263
                          1264 +1
                          1265 +1
                          1266 +1
                          1267 +1
007F FA                   1268 +1                               CLI
0080 B8E000               1269 +1                               MOV    AX, ENABLE_NORMAL_MASK
0083 E67A                 1270 +1                               OUT    PIC_PORT_1, AL
0085 FB                   1271 +1                               STI
                          1272
0086 58                   1273                                  POP    AX
0087 CF                   1274                                  IRET
                          1275
                          1276
                          1277     PREDICTOR_OUTPUT_INTERRUPT          ENDP
                          1278
                          1279
                          1280
                          1281 +1  $EJECT
```

```
LOC  OBJ          LINE     SOURCE

                  1282
                  1283
0088              1284     FE_OUTPUT_INTERRUPT     PROC    FAR
                  1285
                  1286
                  1287     ;*****************************************************************************
                  1288     ;* THIS INTERRUPT HANDLER IS ACTIVATED BY FE ENABLE_ SIGNAL ON THE DECODER *
                  1289     ;* BUS. IT IS RESPONSIBLE FOR OUTPUTING FEH AND FEL TO THE DECODER  BUS    *
                  1290     ;* ON ENTRY TO THE HANDLER, AX IS PUSHED ONTO STACK, DATA BUS PORT IS      *
                  1291     ;* CONVERTED FROM INPUT STATE TO OUTPUT STATE, THE APPROPRIATE FEX IS      *
                  1292     ;* OUTPUTED TO THE BUS, POINTER TO BUFFER IS INCREMENTED TO NEXT FEX TO BE *
                  1293     ;* OUTPUT. THIS ROUTINE WILL LOOP AROUND UNTIL FE ENDABLE_ IS HIGH, THEN   *
                  1294     ;* IT WILL PULL DATA OFF FROM THE DATA BUS, CONVERT PORT TO INPUT STATE    *
                  1295     ;* AND EXIT                                                                 *
                  1296     ;*                                                                          *
                  1297     ;* REGISTER USAGE:                                                          *
                  1298     ;*              AX, BX                                                      *
                  1299     ;*****************************************************************************
                  1300
                  1301
0088 50           1302                             PUSH    AX
                  1303
                  1304 +1
                  1305 +1
                  1306 +1
                  1307 +1
0089 B8FE00       1308 +1                          MOV     AX,ENABLE_SOF_MASK
008C E67A         1309 +1                          OUT     PIC_PORT_1, AL
008E FB           1310 +1                          STI
                  1311
                  1312
                  1313                             ;CONFIGURE DATA BUS PORT TO BE OUTPUT
                  1314
008F B89999       1315                             MOV     AX, D_DATA_BUS_OUT
0092 E74E         1316                             OUT     DECODER_BUS_SPVR, AX
                  1317
0094 8B4730       1318                             MOV     AX, FE_BUFFER[BX]
                  1319
                  1320                             ;BX IS THE POINTER TO THE NEXT WORD TO BE OUTPUT IN FE BUFFER
                  1321
0097 E74A         1322                             OUT     D_DATA_BUS, AX
                  1323
                  1324                             ;INCREMENT THE OUTPUT POINTER
                  1325
0099 83C302       1326                             ADD     BX, TYPE FE_BUFFER
009C 83FB40       1327                             CMP     BX, LEN_FE_BUFFER * TYPE FE_BUFFER
009F 7203         1328                             JB      LABEL_FE_ENABLE
00A1 BB0000       1329                             MOV     BX, 0
                  1330
                  1331
                  1332
00A4 E54C         1333     LABEL_FE_ENABLE:        IN      AX, D_CONTROL_BUS
00A6 251000       1334                             AND     AX, SFEEN
00A9 74F9         1335                             JZ      LABEL_FE_ENABLE
                  1336
```

```
LOC  OBJ                LINE     SOURCE

                        1324
00BF                    1325     FE_INPUT_INTERRUPT      PROC    FAR
                        1326
                        1327     ;*************************************************************************
                        1328     ;* THIS INTERRUPT ROUTINE IS ACTIVATED BY THE FE ENABLE_ SIGNAL ON THE     *
                        1329     ;* CODER BUS. IT IS RESPONSIBLE FOR TAKING THE FEX(FEH AND FEL) DATA FROM  *
                        1330     ;* THE CODER DATA BUS AND STORE THEM INTO THE FE BUFFER.                   *
                        1331     ;* AFTER ENTERING, THE ROUTINE PUSHES AX REGISTER ONTO STACK, AND CHECK    *
                        1332     ;* FOR FE CLOCK. ONCE FE CLOCK IS VALID, THE FEX DATA IS TAKEN FROM THE    *
                        1333     ;* AND STORED IN THE FE_BUFFER. THE ROUTINE THEN INCREMENTS THE BUFFER     *
                        1334     ;* POINTER AND EXIT.                                                       *
                        1335     ;*                                                                         *
                        1336     ;*               REGISTER USAGES:                                          *
                        1337     ;*                        AX,SI                                            *
                        1338     ;*                                                                         *
                        1339     ;*************************************************************************
                        1340
                        1341
                        1342                             ;THIS ROUTINE IS ACTIVATED BY FE_ENABLE SIGNAL ON THE
                        1343                             ;CODER CONTROL BUS
                        1344
                        1345
                        1346
00BF 50                 1347                             PUSH    AX
                        1348 +1
                        1349 +1
                        1350 +1
                        1351 +1
00C0 B8FE00             1352 +1                          MOV     AX,ENABLE_SOF_MASK
00C3 E67A               1353 +1                          OUT     PIC_PORT_1, AL
00C5 FB                 1354 +1                          STI
                        1355
00C6                    1356     LABEL_FE_CLK:            ;CHECK FOR FE CLOCK VALIDITY
                        1357
00C6 E544               1358                             IN      AX, C_CONTROL_BUS
00C8 250006             1359                             AND     AX, CFECLK
00CB 74F9               1360                             JZ      LABEL_FE_CLK
                        1361                             ;FE CLOCK IS TRUE(I.E. SIGNAL IS HIGH), READ IN FEX FROM DATA
                                  BUS
                        1362
00CD E542               1363                             IN      AX, C_DATA_BUS
                        1364
                        1365
00CF 894430             1366                             MOV     FE_BUFFER[SI], AX
                        1367
                        1368                             ;SI IS THE OFFSET POINTER FOR DEPOSITION
                        1369
                        1370                             ;INCREMENT POINTERS
                        1371
                        1372
00D2 83C602             1373                             ADD     SI, TYPE FE_BUFFER
00D5 FF068200           1374                             INC     FE_SERVICE_COUNTER      ;INCREMENT THE SERVICE
                        1375                                                             ;COUNTER TO ENABLE NON-INTERR
                                  UPT
                        1376                                                             ;DRIVEN QUANTIZATION PROCESS
```

```
LOC  OBJ                LINE     SOURCE

                        1337                           ;NOW SFEEN_ IS FALSE(SIGNAL IS HIGH), TAKE DATA OFF BUS
00AB B89B7B             1338                           MOV     AX, D_DATA_BUS_IN
00AE E74E               1339                           OUT     DECODER_BUS_SPVR, AX
                        1340
                        1341 +1
                        1342 +1
                        1343 +1
                        1344 +1
00B0 FA                 1345 +1                        CLI
00B1 B8E000             1346 +1                        MOV     AX, ENABLE_NORMAL_MASK
00B4 E67A               1347 +1                        OUT     PIC_PORT_1, AL
00B6 FB                 1348 +1                        STI
                        1349
00B7 58                 1350                           POP     AX
                        1351
00B8 CF                 1352                           IRET
                        1353
                        1354     FE_OUTPUT_INTERRUPT   ENDP
                        1355
                        1356
                        1357
                        1358 +1  $EJECT
```

```
LOC  OBJ              SOURCE

00B9                  FE_INPUT_INTERRUPT      PROC    FAR

                      ;*****************************************************************************
                      ;* THIS INTERRUPT ROUTINE IS ACTIVATED BY THE FE ENABLE_ SIGNAL ON THE        *
                      ;* CODER BUS. IT IS RESPONSIBLE FOR TAKING THE FEX(FEH AND FEL) DATA FROM     *
                      ;* THE CODER DATA BUS AND STORE THEM INTO THE FE BUFFER.                      *
                      ;* AFTER ENTERING, THE ROUTINE PUSHES AX REGISTER ONTO STACK, AND CHECK       *
                      ;* FOR FE CLOCK. ONCE FE CLOCK IS VALID, THE FEX DATA IS TAKEN FROM THE       *
                      ;* AND STORED IN THE FE_BUFFER. THE ROUTINE THEN INCREMENTS THE BUFFER        *
                      ;* POINTER AND EXIT.                                                          *
                      ;*                                                                           *
                      ;*               REGISTER USAGES:                                            *
                      ;*                         AX,SI                                             *
                      ;*                                                                           *
                      ;*****************************************************************************


                                              ;THIS ROUTINE IS ACTIVATED BY FE_ENABLE SIGNAL ON THE
                                              ;CODER CONTROL BUS



00B9 50                                       PUSH    AX
                 +1
                 +1
                 +1
                 +1
00BA B8FE00      +1                           MOV     AX,ENABLE_SOF_MASK
00BD E67A        +1                           OUT     PIC_PORT_1, AL
00BF FB          +1                           STI

00C0                  LABEL_FE_CLK:           ;CHECK FOR FE CLOCK VALIDITY

00C0 E544                                     IN      AX, C_CONTROL_BUS
00C2 250008                                   AND     AX, CFECLK
00C5 74F9                                     JZ      LABEL_FE_CLK
                                              ;FE CLOCK IS TRUE(I.E. SIGNAL IS HIGH), READ IN FEX FROM DATA
                       BUS

00C7 E542                                     IN      AX, C_DATA_BUS


00C9 894430                                   MOV     FE_BUFFER[SI], AX

                                              ;SI IS THE OFFSET POINTER FOR DEPOSITION

                                              ;INCREMENT POINTERS


00CC 83C602                                   ADD     SI, TYPE FE_BUFFER
00CF FF068200                                 INC     FE_SERVICE_COUNTER       ;INCREMENT THE SERVICE
                                                                               ;COUNTER TO ENABLE NON-INTERR
                      UPT
                                                                               ;DRIVEN QUANTIZATION PROCESS
```

```
LOC  OBJ                  LINE     SOURCE

                          1412
00D3 83FE40               1413                         CMP    SI, LEN_FE_BUFFER * TYPE FE_BUFFER      ;END OF BUFFE
                                   R?
00D6 7203                 1414                         JB     LABEL_FE_INPUT_END
00D8 BE0000               1415                         MOV    SI, 0                       ;RESET POINTER TO BEGINNING
                          1416                                                             ;OF CIRCULAR BUFFER.
                          1417
00DB                      1418 +1  LABEL_FE_INPUT_END:
                          1419 +1
                          1420 +1
                          1421 +1
00DB FA                   1422 +1                      CLI
00DC B8E000               1423 +1                      MOV    AX, ENABLE_NORMAL_MASK
00DF E67A                 1424 +1                      OUT    PIC_PORT_1, AL
00E1 FB                   1425 +1                      STI
00E2 58                   1426                         POP    AX
                          1427
00E3 CF                   1428                         IRET
                          1429
                          1430     FE_INPUT_INTERRUPT          ENDP
                          1431
                          1432
                          1433 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                        1434
                        1435
                        1436
00E4                    1437     PREDICTOR_INPUT_INTERRUPT          PROC    FAR
                        1438
                        1439
                        1440
                        1441     ;***************************************************************************
                        1442     ;* THIS PROCEDURE IS ACTIVATED BY THE PEN_  SIGNAL ON THE CODER BUS.        *
                        1443     ;* IT INPUTS THE PREDICTOR COEFFICIENTS FROM THE CODER BUS AND STORES THE  *
                        1444     ;* DATA IN THE PREDICTOR_COEFF BUFFER.                                      *
                        1445     ;*                                                                          *
                        1446     ;*                  REGISTER USAGES:                                        *
                        1447     ;*                          AX, DI                                          *
                        1448     ;*                                                                          *
                        1449     ;***************************************************************************
00E4 50                 1450                             PUSH    AX
                        1451
                        1452 +1
                        1453 +1
                        1454 +1
                        1455 +1
00E5 B8FE00             1456 +1                          MOV     AX,ENABLE_SOF_MASK
00E8 E67A               1457 +1                          OUT     PIC_PORT_1, AL
00EA FB                 1458 +1                          STI
                        1459
                        1460                             ;TEST FOR P CLOCK VALID
                        1461
                        1462
00EB E544               1463     LABEL_P_CLK:            IN      AX, C_CONTROL_BUS
                        1464
00ED 252000             1465                             AND     AX, CPCLK
00F0 74F9               1466                             JZ      LABEL_P_CLK
                        1467
                        1468
                        1469                             ;P CLOCK IS VALID, READ IN PREDICTOR COEFFICIENTS FROM
                        1470                             ;DATA BUS
                        1471
                        1472
00F2 E542               1473                             IN      AX, C_DATA_BUS
                        1474
                        1475                             ;BP SPECIFIES THE FRAME STARTING OFFSET(0, 16, 32 OR 48)
                        1476                             ;DI SPECIFIES THE OFFSET WITHIN EACH FRAME (0, 2, 4...14)
00F4 3E8903             1477                             MOV     DS:PREDICTOR_BUFFER[BP][DI], AX
                        1478
00F7 83C702             1479                             ADD     DI, TYPE PREDICTOR_BUFFER       ;INCREMENT OFFSET
                        1480                                                                     ;VALUE WITHIN ONE FRA
                                 ME
                        1481
                        1482
                        1483 +1
                        1484 +1
                        1485 +1
                        1486 +1
00FA FA                 1487 +1                          CLI
```

```
LOC  OBJ                  LINE     SOURCE

00FB B8E000               1488 +1                           MOV     AX, ENABLE_NORMAL_MASK
00FE E67A                 1489 +1                           OUT     PIC_PORT_1, AL
0100 FB                   1490 +1                           STI
                          1491
0101 58                   1492                              POP     AX
0102 CF                   1493                              IRET
                          1494
                          1495     PREDICTOR_INPUT_INTERRUPT        ENDP
                          1496
                          1497
                          1498
                          1499 +1  $EJECT
```

```
LOC  OBJ                LINE     SOURCE

                        1500
0103                    1501     EXCEPTION_INT           PROC    FAR
                        1502
                        1503             ;*******************************************************
                        1504             ;* THIS IS THE EXCEPTION HANDLER FOR THE 8086 CPU.      *
                        1505             ;* UNDER NORMAL OPERATION ENVIRONMENT, THIS CODE SHOULD *
                        1506             ;* NEVER BE EXECUTED.                                   *
                        1507             ;* IN THE UNLIKELY EVENT THAT THIS HANDLER IS ACTIVATED,*
                        1508             ;* IT SETS THE LED_DISPLAY_VALUE TO OFFH AND OUTPUT     *
                        1509             ;* TO THE ERROR_LED.                                    *
                        1510             ;*               REGISTER USAGES:                       *
                        1511             ;*                       AX,                            *
                        1512             ;*                                                      *
                        1513             ;*******************************************************
                        1514
                        1515
                        1516             ;EXTERNAL INTERRUPTS ARE DISABLED DURING EXECUTION OF THIS HANDLER
                        1517
                        1518
0103 50                 1519                             PUSH    AX
0104 B8FFFF             1520                             MOV     AX, OFFFFH
0107 A37000             1521                             MOV     LED_DISPLAY_VALUE, AX
010A E664               1522                             OUT     ERROR_LED, AL
010C 58                 1523                             POP     AX
010D FB                 1524                             STI
010E CF                 1525                             IRET
                        1526
                        1527
                        1528     EXCEPTION_INT           ENDP
                        1529
                        1530
                        1531
----                    1532     INTERRUPT_CODE          ENDS
                        1533                             END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

APPENDIX C

APPENDIX C

DESCRIPTION OF THE ORIGINAL DRT TAPE SUPPLIED BY CRC

APPENDIX C

DESCRIPTION OF THE ORIGINAL DRT TAPE SUPPLIED BY CRC

ANSWERS TO DIAGNOSTIC RHYME TESTS

28-Oct-1982

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
1	01	BOB	GOB	3	7	231	3
1	02	DAUNT	TAUNT	1	1	187	0
1	03	MOOT	BOOT	2	2	130	- 3
1	04	SHEET	CHEAT	2	3	091	- 6
1	05	JAB	GAB	2	4	176	0
1	06	POT	TOT	1	5	231	3
1	07	GHOST	BOAST	1	6	185	0
1	08	RILL	NILL	3	7	154	0
1	09	ZED	SAID	2	1	158	0
1	10	GNAW	DAW	1	2	205	3
1	11	SHOES	CHOOSE	1	3	111	- 3
1	12	CHEEP	KEEP	2	4	083	- 6
1	13	BANK	DANK	1	5	146	0
1	14	GOT	DOT	1	6	209	3
1	15	ROSE	NOSE	3	7	139	- 3
1	16	DINT	TINT	2	1	099	- 6
1	17	NECK	DECK	2	2	190	0
1	18	THONG	TONG	1	3	205	3
1	19	CHEW	COO	2	4	130	- 3
1	20	WEED	REED	1	5	068	- 9
1	21	SHAG	SAG	2	6	222	3
1	22	KNOB	ROB	3	7	195	0
1	23	VOLE	FOAL	1	1	127	- 3
1	24	NIP	DIP	2	2	150	0
1	25	FENCE	PENCE	2	3	086	- 6
1	26	SAW	THAW	1	4	214	3
1	27	POOL	TOOL	2	5	110	- 3
1	28	YIELD	WIELD	1	6	104	- 3
1	29	GNAT	RAT	3	7	153	0
1	30	COOT	TOOT	3	7	116	- 3
1	31	BOND	POND	1	1	189	0
1	32	MOAN	BONE	1	2	159	0
1	33	VILL	BILL	2	3	137	- 3
1	34	JEST	GUEST	2	4	165	0
1	35	FOUGHT	THOUGHT	1	5	246	3
1	36	COOP	POOP	1	6	098	- 6
1	37	NEAP	REAP	3	7	075	- 6
1	38	VAST	FAST	2	1	178	0
1	39	KNOCK	DOCK	1	2	186	0
1	40	THOSE	DOZE	2	3	210	3
1	41	SING	THING	2	4	148	0
1	42	MET	NET	2	5	156	0
1	43	CAUGHT	TAUGHT	1	6	283	6
1	44	NUDE	RUDE	3	7	131	- 3
1	45	BEAN	PEEN	1	1	125	- 3
1	46	MAD	BAD	1	2	165	0
1	47	VOX	BOX	2	3	303	6
1	48	JOE	GO	2	4	210	3
1	49	BID	DID	2	5	160	0
1	50	YEN	WREN	2	6	138	- 3
1	51	ROT	NOT	3	7	287	6

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
1	52	ZOO	SUE	1	1	152	0
1	53	NEED	DEED	1	2	091	- 6
1	54	THAN	DAN	2	3	167	0
1	55	CHOP	COP	2	4	264	3
1	56	FORE	THOR	2	5	171	0
1	57	HIT	FIT	2	6	181	0
1	58	NEST	REST	3	7	199	3

ANSWERS TO DIAGNOSTIC RHYME TESTS

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
2	01	PEST	TEST	3	7	194	0
2	02	VAULT	FAULT	2	1	268	3
2	03	NEWS	DUES	1	2	094	- 6
2	04	VEE	BEE	1	3	123	- 3
2	05	SANK	THANK	1	4	155	0
2	06	WAD	ROD	2	5	230	3
2	07	SHOW	SO	2	6	175	0
2	08	RIP	NIP	3	7	133	- 3
2	09	DENSE	TENSE	1	1	112	- 3
2	10	MOSS	BOSS	2	2	233	3
2	11	FOO	POO	2	3	127	- 3
2	12	ZEE	THEE	1	4	108	- 3
2	13	FAD	THAD	2	5	208	3
2	14	HOP	FOP	2	6	262	3
2	15	ROAD	NODE	3	7	175	0
2	16	GIN	CHIN	1	1	116	- 3
2	17	MEND	BEND	1	2	117	- 3
2	18	SHAW	CHAW	2	3	214	3
2	19	JUICE	GOOSE	1	4	094	- 6
2	20	PEAK	TEAK	2	5	094	- 6
2	21	GAT	BAT	1	6	175	0
2	22	NOT	ROT	3	7	235	3
2	23	GOAT	COAT	2	1	151	0
2	24	MIT	BIT	1	2	131	- 3
2	25	THEN	DEN	1	3	125	- 3
2	26	JAWS	GAUZE	2	4	197	0
2	27	MOON	NOON	1	5	089	- 6
2	28	KEY	TEA	2	6	080	- 6
2	29	RAP	NAP	3	7	184	0
2	30	FAN	PAN	3	7	180	0
2	31	JOCK	CHOCK	2	1	188	0
2	32	NOTE	DOTE	2	2	162	0
2	33	THICK	TICK	1	3	142	0
2	34	CHAIR	CARE	1	4	128	- 3
2	35	BONG	DONG	2	5	193	0
2	36	YOU	RUE	2	6	172	- 3
2	37	WREATH	NEATH	3	7	096	- 6
2	38	GAFF	CALF	1	1	163	0
2	39	MOM	BOMB	2	2	196	0
2	40	THOUGH	DOUGH	1	3	189	0
2	41	JILT	GILT	1	4	166	0
2	42	PENT	TENT	1	5	120	- 3
2	43	YAWL	WALL	2	6	223	3
2	44	ROOSE	NOOSE	3	7	166	0
2	45	VEAL	FEEL	2	1	133	- 3
2	46	NAB	DAB	2	2	170	0
2	47	VON	BON	1	3	196	0
2	48	SOLE	THOLE	1	4	147	0
2	49	FIN	THIN	1	5	125	- 3
2	50	KEG	PEG	1	6	131	- 3
2	51	GNAW	RAW	3	7	199	3

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
2	52	DUNE	TUNE	2	1	095	- 6
2	53	MEAT	BEAT	2	2	076	- 6
2	54	SHAD	CHAD	1	3	151	0
2	55	JOT	GOT	1	4	199	3
2	56	BOWL	DOLE	1	5	148	0
2	57	GILL	DILL	1	6	121	- 3
2	58	RED	NED	3	7	116	- 3

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
3	01	BOB	GOB	3	7	253	3
3	02	DAUNT	TAUNT	2	1	199	3
3	03	MOOT	BOOT	1	2	113	- 3
3	04	SHEET	CHEAT	1	3	062	- 9
3	05	JAB	GAB	1	4	125	- 3
3	06	POT	TOT	2	5	192	0
3	07	GHOST	BOAST	2	6	162	0
3	08	RILL	NILL	3	7	111	- 3
3	09	ZED	SAID	1	1	121	- 3
3	10	GNAW	DAW	2	2	217	3
3	11	SHOES	CHOOSE	2	3	097	- 6
3	12	CHEEP	KEEP	1	4	077	- 6
3	13	BANK	DANK	2	5	118	- 3
3	14	GOT	DOT	2	6	182	0
3	15	ROSE	NOSE	3	7	147	0
3	16	DINT	TINT	1	1	103	- 3
3	17	NECK	DECK	1	2	133	- 3
3	18	THONG	TONG	2	3	183	0
3	19	CHEW	COO	1	4	095	- 6
3	20	WEED	REED	2	5	096	- 6
3	21	SHAG	SAG	1	6	174	0
3	22	KNOB	ROB	3	7	217	3
3	23	VOLE	FOAL	2	1	139	- 3
3	24	NIP	DIP	1	2	128	- 3
3	25	FENCE	PENCE	1	3	115	- 3
3	26	SAW	THAW	2	4	222	3
3	27	POOL	TOOL	1	5	118	- 3
3	28	YIELD	WIELD	2	6	124	- 3
3	29	GNAT	RAT	3	7	180	0
3	30	COOT	TOOT	3	7	119	- 3
3	31	BOND	POND	2	1	203	3
3	32	MOAN	BONE	2	2	158	0
3	33	VILL	BILL	1	3	164	0
3	34	JEST	GUEST	1	4	158	0
3	35	FOUGHT	THOUGHT	2	5	225	3
3	36	COOP	POOP	2	6	181	0
3	37	NEAP	REAP	3	7	118	- 3
3	38	VAST	FAST	1	1	186	0
3	39	KNOCK	DOCK	2	2	198	0
3	40	THOSE	DOZE	1	3	175	0
3	41	SING	THING	1	4	111	- 3
3	42	MET	NET	1	5	110	- 3
3	43	CAUGHT	TAUGHT	2	6	228	3
3	44	NUDE	RUDE	3	7	132	- 3
3	45	BEAN	PEEN	2	1	072	- 6
3	46	MAD	BAD	2	2	170	0
3	47	VOX	BOX	1	3	249	3
3	48	JOE	GO	1	4	179	0
3	49	BID	DID	1	5	117	- 3
3	50	YEN	WREN	1	6	098	- 6
3	51	ROT	NOT	3	7	209	3

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
3	52	ZOO	SUE	2	1	127	- 3
3	53	NEED	DEED	2	2	084	- 6
3	54	THAN	DAN	1	3	138	- 3
3	55	CHOP	COP	1	4	201	3
3	56	FORE	THOR	1	5	165	0
3	57	HIT	FIT	1	6	137	- 3
3	58	NEST	REST	3	7	162	0

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
4	01	PEST	TEST	3	7	216	3
4	02	VAULT	FAULT	1	1	243	3
4	03	NEWS	DUES	2	2	120	- 3
4	04	VEE	BEE	2	3	113	- 3
4	05	SANK	THANK	2	4	146	0
4	06	WAD	ROD	1	5	131	- 3
4	07	SHOW	SO	1	6	156	0
4	08	RIP	NIP	3	7	156	0
4	09	DENSE	TENSE	2	1	104	- 3
4	10	MOSS	BOSS	1	2	236	3
4	11	FOO	POO	1	3	126	- 3
4	12	ZEE	THEE	2	4	135	- 3
4	13	FAD	THAD	1	5	208	3
4	14	HOP	FOP	1	6	256	3
4	15	ROAD	NODE	3	7	126	- 3
4	16	GIN	CHIN	2	1	092	- 6
4	17	MEND	BEND	2	2	119	- 3
4	18	SHAW	CHAW	1	3	203	3
4	19	JUICE	GOOSE	2	4	133	- 3
4	20	PEAK	TEAK	1	5	095	- 6
4	21	GAT	BAT	2	6	187	0
4	22	NOT	ROT	3	7	216	3
4	23	GOAT	COAT	1	1	182	0
4	24	MIT	BIT	2	2	151	0
4	25	THEN	DEN	2	3	118	- 3
4	26	JAWS	GAUZE	1	4	240	3
4	27	MOON	NOON	2	5	091	- 6
4	28	KEY	TEA	1	6	095	- 6
4	29	RAP	NAP	3	7	167	0
4	30	FAN	PAN	3	7	192	0
4	31	JOCK	CHOCK	1	1	294	6
4	32	NOTE	DOTE	1	2	189	0
4	33	THICK	TICK	2	3	197	0
4	34	CHAIR	CARE	2	4	186	0
4	35	BONG	DONG	1	5	232	3
4	36	YOU	RUE	1	6	136	- 3
4	37	WREATH	NEATH	3	7	088	- 6
4	38	GAFF	CALF	2	1	201	3
4	39	MOM	BOMB	1	2	217	3
4	40	THOUGH	DOUGH	2	3	201	3
4	41	JILT	GILT	2	4	179	0
4	42	PENT	TENT	2	5	141	0
4	43	YAWL	WALL	1	6	239	3
4	44	ROOSE	NOOSE	3	7	138	- 3
4	45	VEAL	FEEL	1	1	127	- 3
4	46	NAB	DAB	1	2	177	0
4	47	VON	BON	2	3	218	3
4	48	SOLE	THOLE	2	4	179	0
4	49	FIN	THIN	2	5	126	- 3
4	50	KEG	PEG	2	6	203	3
4	51	GNAW	RAW	3	7	257	3

TEST	WORD NUMBER	LEFT WORD	RIGHT WORD	ANSWER	ATTRIBUTE	PEAK STRENGTH (milivolts)	POWER (dB)
4	52	DUNE	TUNE	1	1	152	0
4	53	MEAT	BEAT	1	2	087	- 6
4	54	SHAD	CHAD	2	3	153	0
4	55	JOT	GOT	2	4	241	3
4	56	BOWL	DOLE	2	5	205	3
4	57	GILL	DILL	2	6	187	0
4	58	RED	NED	3	7	187	0

APPENDIX D

APPENDIX D

LIST OF WRONG ANSWERS

APPENDIX D

LIST OF WRONG ANSWERS

SUMMARY OF ERRORS FOR DIAGNOSTIC RHYME TESTS 28-Oct-1982

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
None	None	LYNNE	DAUNT	TAUNT	1	Left
None	None	LYNNE	VOX	BOX	3	Right
None	None	PETER	FAD	THAD	5	Right
None	None	ALVA	FIN	THIN	5	Right
None	None	WOLF	FORE	THOR	5	Right

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
None	Armoured Car	WOLF	BOND	POND	1	Left
None	Armoured Car	LYNNE	DAUNT	TAUNT	1	Left
None	Armoured Car	PETER	DAUNT	TAUNT	1	Left
None	Armoured Car	ALVA	SHAD	CHAD	3	Right
None	Armoured Car	ALVA	THOSE	DOZE	3	Right
None	Armoured Car	WOLF	VEE	BEE	3	Left
None	Armoured Car	LYNNE	VEE	BEE	3	Left
None	Armoured Car	ALVA	VEE	BEE	3	Right
None	Armoured Car	PETER	VEE	BEE	3	Left
None	Armoured Car	LYNNE	VILL	BILL	3	Right
None	Armoured Car	ALVA	VOX	BOX	3	Right
None	Armoured Car	PETER	FIN	THIN	5	Right
None	Armoured Car	WOLF	FIN	THIN	5	Left
None	Armoured Car	WOLF	FORE	THOR	5	Right
None	Armoured Car	LYNNE	MET	NET	5	Left
None	Armoured Car	LYNNE	MET	NET	5	Neither
None	Armoured Car	WOLF	MET	NET	5	Left
None	Armoured Car	LYNNE	CAUGHT	TAUGHT	6	Neither
None	Armoured Car	ALVA	KEG	PEG	6	Neither
None	Armoured Car	LYNNE	SHAG	SAG	6	Left

SUMMARY OF ERRORS FOR DIAGNOSTIC RHYME TESTS

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
None	Helicopter	ALVA	SHAD	CHAD	3	Right
None	Helicopter	LYNNE	SHOES	CHOOSE	3	Right
None	Helicopter	WOLF	VEE	BEE	3	Left
None	Helicopter	PETER	VEE	BEE	3	Left
None	Helicopter	WOLF	FAD	THAD	5	Right
None	Helicopter	ALVA	FAD	THAD	5	Left
None	Helicopter	LYNNE	FIN	THIN	5	Right
None	Helicopter	WOLF	FIN	THIN	5	Left
None	Helicopter	WOLF	FIN	THIN	5	Right
None	Helicopter	PETER	FIN	THIN	5	Left
None	Helicopter	ALVA	FIN	THIN	5	Right
None	Helicopter	PETER	MET	NET	5	Left

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 4.8	None	LYNNE	DAUNT	TAUNT	1	Left
RELP 4.8	None	ALVA	GIN	CHIN	1	Left
RELP 4.8	None	LYNNE	GOAT	COAT	1	Left
RELP 4.8	None	LYNNE	VOLE	FOAL	1	Left
RELP 4.8	None	PETER	VOLE	FOAL	1	Left
RELP 4.8	None	WOLF	VOLE	FOAL	1	Left
RELP 4.8	None	PETER	VOLE	FOAL	1	Left
RELP 4.8	None	PETER	ZOO	SUE	1	Left
RELP 4.8	None	LYNNE	FENCE	PENCE	3	Right
RELP 4.8	None	ALVA	SHAD	CHAD	3	Right
RELP 4.8	None	LYNNE	SHAD	CHAD	3	Right
RELP 4.8	None	PETER	SHAD	CHAD	3	Right
RELP 4.8	None	WOLF	SHAW	CHAW	3	Right
RELP 4.8	None	PETER	SHAW	CHAW	3	Right
RELP 4.8	None	ALVA	SHAW	CHAW	3	Right
RELP 4.8	None	ALVA	SHOES	CHOOSE	3	Right
RELP 4.8	None	LYNNE	SHOES	CHOOSE	3	Right
RELP 4.8	None	WOLF	SHOES	CHOOSE	3	Right
RELP 4.8	None	ALVA	THAN	DAN	3	Right
RELP 4.8	None	ALVA	THAN	DAN	3	Right
RELP 4.8	None	WOLF	THEN	DEN	3	Left
RELP 4.8	None	ALVA	THEN	DEN	3	Left
RELP 4.8	None	LYNNE	THONG	TONG	3	Left
RELP 4.8	None	LYNNE	THONG	TONG	3	Left
RELP 4.8	None	WOLF	VEE	BEE	3	Left
RELP 4.8	None	ALVA	VILL	BILL	3	Right
RELP 4.8	None	ALVA	VILL	BILL	3	Right
RELP 4.8	None	LYNNE	VOX	BOX	3	Right
RELP 4.8	None	ALVA	VOX	BOX	3	Right
RELP 4.8	None	LYNNE	VOX	BOX	3	Right
RELP 4.8	None	PETER	VOX	BOX	3	Right
RELP 4.8	None	WOLF	VOX	BOX	3	Right
RELP 4.8	None	WOLF	VOX	BOX	3	Right
RELP 4.8	None	PETER	VOX	BOX	3	Right
RELP 4.8	None	ALVA	VOX	BOX	3	Right
RELP 4.8	None	LYNNE	JAB	GAB	4	Left
RELP 4.8	None	WOLF	JAB	GAB	4	Left
RELP 4.8	None	PETER	JOE	GO	4	Right
RELP 4.8	None	PETER	JOE	GO	4	Right
RELP 4.8	None	LYNNE	SING	THING	4	Right
RELP 4.8	None	LYNNE	SING	THING	4	Right
RELP 4.8	None	PETER	SING	THING	4	Right
RELP 4.8	None	WOLF	SING	THING	4	Right
RELP 4.8	None	WOLF	SING	THING	4	Right
RELP 4.8	None	PETER	SING	THING	4	Right
RELP 4.8	None	ALVA	SING	THING	4	Right
RELP 4.8	None	PETER	ZEE	THEE	4	Right
RELP 4.8	None	ALVA	BANK	DANK	5	Left
RELP 4.8	None	WOLF	BANK	DANK	5	Left
RELP 4.8	None	LYNNE	BANK	DANK	5	Left
RELP 4.8	None	ALVA	BID	DID	5	Left

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 4.8	None	PETER	BID	DID	5	Left
RELP 4.8	None	LYNNE	BID	DID	5	Left
RELP 4.8	None	ALVA	FAD	THAD	5	Left
RELP 4.8	None	PETER	FAD	THAD	5	Right
RELP 4.8	None	ALVA	FIN	THIN	5	Right
RELP 4.8	None	PETER	FIN	THIN	5	Right
RELP 4.8	None	PETER	FORE	THOR	5	Left
RELP 4.8	None	ALVA	FORE	THOR	5	Right
RELP 4.8	None	WOLF	FORE	THOR	5	Right
RELP 4.8	None	ALVA	FOUGHT	THOUGHT	5	Right
RELP 4.8	None	PETER	FOUGHT	THOUGHT	5	Right
RELP 4.8	None	LYNNE	FOUGHT	THOUGHT	5	Right
RELP 4.8	None	ALVA	FOUGHT	THOUGHT	5	Left
RELP 4.8	None	WOLF	FOUGHT	THOUGHT	5	Right
RELP 4.8	None	LYNNE	MET	NET	5	Right
RELP 4.8	None	WOLF	MET	NET	5	Right
RELP 4.8	None	ALVA	MET	NET	5	Right
RELP 4.8	None	LYNNE	MOON	NOON	5	Right
RELP 4.8	None	LYNNE	PENT	TENT	5	Right
RELP 4.8	None	ALVA	YIELD	WIELD	6	Left

SUMMARY OF ERRORS FOR DIAGNOSTIC RHYME TESTS

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 4.8	Armoured Car	ALVA	DAUNT	TAUNT	1	Left
RELP 4.8	Armoured Car	PETER	DAUNT	TAUNT	1	Left
RELP 4.8	Armoured Car	PETER	VAST	FAST	1	Left
RELP 4.8	Armoured Car	LYNNE	VAST	FAST	1	Left
RELP 4.8	Armoured Car	ALVA	VAST	FAST	1	Left
RELP 4.8	Armoured Car	WOLF	VAULT	FAULT	1	Left
RELP 4.8	Armoured Car	WOLF	VOLE	FOAL	1	Left
RELP 4.8	Armoured Car	LYNNE	VOLE	FOAL	1	Left
RELP 4.8	Armoured Car	PETER	VOLE	FOAL	1	Left
RELP 4.8	Armoured Car	ALVA	MAD	BAD	2	Neither
RELP 4.8	Armoured Car	PETER	MEND	BEND	2	Right
RELP 4.8	Armoured Car	PETER	MOAN	BONE	2	Right
RELP 4.8	Armoured Car	ALVA	NIP	DIP	2	Neither
RELP 4.8	Armoured Car	LYNNE	SHOES	CHOOSE	3	Right
RELP 4.8	Armoured Car	ALVA	SHOES	CHOOSE	3	Right
RELP 4.8	Armoured Car	LYNNE	THEN	DEN	3	Right
RELP 4.8	Armoured Car	WOLF	THONG	TONG	3	Left
RELP 4.8	Armoured Car	PETER	THONG	TONG	3	Left
RELP 4.8	Armoured Car	ALVA	THOSE	DOZE	3	Right
RELP 4.8	Armoured Car	WOLF	THOSE	DOZE	3	Left
RELP 4.8	Armoured Car	LYNNE	THOUGH	DOUGH	3	Right
RELP 4.8	Armoured Car	PETER	VEE	BEE	3	Right
RELP 4.8	Armoured Car	WOLF	VEE	BEE	3	Right
RELP 4.8	Armoured Car	ALVA	VILL	BILL	3	Right
RELP 4.8	Armoured Car	LYNNE	VILL	BILL	3	Right
RELP 4.8	Armoured Car	LYNNE	VON	BON	3	Right
RELP 4.8	Armoured Car	PETER	VON	BON	3	Right
RELP 4.8	Armoured Car	ALVA	VOX	BOX	3	Right
RELP 4.8	Armoured Car	LYNNE	VOX	BOX	3	Right
RELP 4.8	Armoured Car	PETER	VOX	BOX	3	Right
RELP 4.8	Armoured Car	LYNNE	CHEEP	KEEP	4	Left
RELP 4.8	Armoured Car	LYNNE	JAB	GAB	4	Left
RELP 4.8	Armoured Car	ALVA	JAB	GAB	4	Right
RELP 4.8	Armoured Car	WOLF	JAB	GAB	4	Left
RELP 4.8	Armoured Car	LYNNE	JILT	GILT	4	Right
RELP 4.8	Armoured Car	PETER	JILT	GILT	4	Right
RELP 4.8	Armoured Car	LYNNE	SANK	THANK	4	Right
RELP 4.8	Armoured Car	WOLF	SANK	THANK	4	Right
RELP 4.8	Armoured Car	ALVA	SAW	THAW	4	Neither
RELP 4.8	Armoured Car	ALVA	SING	THING	4	Right
RELP 4.8	Armoured Car	LYNNE	SING	THING	4	Right
RELP 4.8	Armoured Car	PETER	SING	THING	4	Right
RELP 4.8	Armoured Car	ALVA	ZEE	THEE	4	Right
RELP 4.8	Armoured Car	LYNNE	ZEE	THEE	4	Right
RELP 4.8	Armoured Car	PETER	ZEE	THEE	4	Right
RELP 4.8	Armoured Car	WOLF	ZEE	THEE	4	Right
RELP 4.8	Armoured Car	LYNNE	BANK	DANK	5	Left
RELP 4.8	Armoured Car	PETER	BANK	DANK	5	Left
RELP 4.8	Armoured Car	ALVA	BOWL	DOLE	5	Right
RELP 4.8	Armoured Car	ALVA	FAD	THAD	5	Left
RELP 4.8	Armoured Car	LYNNE	FIN	THIN	5	Right

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 4.8	Armoured Car	PETER	FIN	THIN	5	Right
RELP 4.8	Armoured Car	LYNNE	FORE	THOR	5	Left
RELP 4.8	Armoured Car	PETER	FOUGHT	THOUGHT	5	Right
RELP 4.8	Armoured Car	ALVA	FOUGHT	THOUGHT	5	Left
RELP 4.8	Armoured Car	WOLF	FOUGHT	THOUGHT	5	Right
RELP 4.8	Armoured Car	PETER	FOUGHT	THOUGHT	5	Left
RELP 4.8	Armoured Car	ALVA	MOON	NOON	5	Right
RELP 4.8	Armoured Car	ALVA	PEAK	TEAK	5	Left
RELP 4.8	Armoured Car	LYNNE	PEAK	TEAK	5	Left
RELP 4.8	Armoured Car	LYNNE	PENT	TENT	5	Right
RELP 4.8	Armoured Car	PETER	PENT	TENT	5	Right
RELP 4.8	Armoured Car	ALVA	GILL	DILL	6	Right
RELP 4.8	Armoured Car	WOLF	HIT	FIT	6	Right
RELP 4.8	Armoured Car	PETER	HIT	FIT	6	Right
RELP 4.8	Armoured Car	ALVA	KEY	TEA	6	Left
RELP 4.8	Armoured Car	WOLF	KEY	TEA	6	Left

SUMMARY OF ERRORS FOR DIAGNOSTIC RHYME TESTS

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 4.8	Helicopter	LYNNE	DAUNT	TAUNT	1	Left
RELP 4.8	Helicopter	ALVA	DUNE	TUNE	1	Right
RELP 4.8	Helicopter	PETER	JOCK	CHOCK	1	Left
RELP 4.8	Helicopter	WOLF	VAST	FAST	1	Right
RELP 4.8	Helicopter	LYNNE	VAULT	FAULT	1	Left
RELP 4.8	Helicopter	WOLF	VAULT	FAULT	1	Left
RELP 4.8	Helicopter	LYNNE	VEAL	FEEL	1	Left
RELP 4.8	Helicopter	PETER	VEAL	FEEL	1	Left
RELP 4.8	Helicopter	ALVA	VEAL	FEEL	1	Right
RELP 4.8	Helicopter	LYNNE	VOLE	FOAL	1	Left
RELP 4.8	Helicopter	LYNNE	VOLE	FOAL	1	Right
RELP 4.8	Helicopter	PETER	VOLE	FOAL	1	Left
RELP 4.8	Helicopter	ALVA	MOM	BOMB	2	Neither
RELP 4.8	Helicopter	PETER	MOM	BOMB	2	Right
RELP 4.8	Helicopter	WOLF	MOOT	BOOT	2	Right
RELP 4.8	Helicopter	PETER	MOOT	BOOT	2	Right
RELP 4.8	Helicopter	LYNNE	MOSS	BOSS	2	Right
RELP 4.8	Helicopter	PETER	MOSS	BOSS	2	Right
RELP 4.8	Helicopter	ALVA	MOSS	BOSS	2	Right
RELP 4.8	Helicopter	WOLF	NEWS	DUES	2	Left
RELP 4.8	Helicopter	LYNNE	FENCE	PENCE	3	Right
RELP 4.8	Helicopter	PETER	FENCE	PENCE	3	Right
RELP 4.8	Helicopter	WOLF	FOO	POO	3	Left
RELP 4.8	Helicopter	ALVA	FOO	POO	3	Neither
RELP 4.8	Helicopter	PETER	FOO	POO	3	Right
RELP 4.8	Helicopter	WOLF	FOO	POO	3	Right
RELP 4.8	Helicopter	ALVA	FOO	POO	3	Right
RELP 4.8	Helicopter	ALVA	SHAD	CHAD	3	Right
RELP 4.8	Helicopter	LYNNE	SHAD	CHAD	3	Left
RELP 4.8	Helicopter	PETER	SHAW	CHAW	3	Right
RELP 4.8	Helicopter	LYNNE	SHEET	CHEAT	3	Right
RELP 4.8	Helicopter	LYNNE	SHOES	CHOOSE	3	Right
RELP 4.8	Helicopter	WOLF	SHOES	CHOOSE	3	Right
RELP 4.8	Helicopter	ALVA	SHOES	CHOOSE	3	Right
RELP 4.8	Helicopter	ALVA	THAN	DAN	3	Right
RELP 4.8	Helicopter	PETER	THAN	DAN	3	Right
RELP 4.8	Helicopter	ALVA	THEN	DEN	3	Left
RELP 4.8	Helicopter	LYNNE	THEN	DEN	3	Left
RELP 4.8	Helicopter	PETER	THEN	DEN	3	Left
RELP 4.8	Helicopter	WOLF	THEN	DEN	3	Left
RELP 4.8	Helicopter	LYNNE	THICK	TICK	3	Left
RELP 4.8	Helicopter	LYNNE	THONG	TONG	3	Left
RELP 4.8	Helicopter	LYNNE	THOSE	DOZE	3	Left
RELP 4.8	Helicopter	PETER	THOSE	DOZE	3	Right
RELP 4.8	Helicopter	ALVA	THOUGH	DOUGH	3	Neither
RELP 4.8	Helicopter	WOLF	THOUGH	DOUGH	3	Left
RELP 4.8	Helicopter	LYNNE	VEE	BEE	3	Right
RELP 4.8	Helicopter	LYNNE	VEE	BEE	3	Left
RELP 4.8	Helicopter	WOLF	VEE	BEE	3	Right
RELP 4.8	Helicopter	ALVA	VEE	BEE	3	Left
RELP 4.8	Helicopter	LYNNE	VEE	BEE	3	Left

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 4.8	Helicopter	PETER	VEE	BEE	3	Left
RELP 4.8	Helicopter	PETER	VEE	BEE	3	Left
RELP 4.8	Helicopter	WOLF	VEE	BEE	3	Left
RELP 4.8	Helicopter	WOLF	VEE	BEE	3	Left
RELP 4.8	Helicopter	ALVA	VEE	BEE	3	Neither
RELP 4.8	Helicopter	LYNNE	VILL	BILL	3	Right
RELP 4.8	Helicopter	ALVA	VILL	BILL	3	Right
RELP 4.8	Helicopter	WOLF	VILL	BILL	3	Right
RELP 4.8	Helicopter	PETER	VILL	BILL	3	Right
RELP 4.8	Helicopter	ALVA	VON	BON	3	Right
RELP 4.8	Helicopter	LYNNE	VON	BON	3	Right
RELP 4.8	Helicopter	PETER	VON	BON	3	Right
RELP 4.8	Helicopter	WOLF	VON	BON	3	Right
RELP 4.8	Helicopter	WOLF	VON	BON	3	Left
RELP 4.8	Helicopter	WOLF	VON	BON	3	Left
RELP 4.8	Helicopter	LYNNE	VOX	BOX	3	Right
RELP 4.8	Helicopter	ALVA	VOX	BOX	3	Right
RELP 4.8	Helicopter	WOLF	VOX	BOX	3	Right
RELP 4.8	Helicopter	ALVA	VOX	BOX	3	Left
RELP 4.8	Helicopter	PETER	VOX	BOX	3	Right
RELP 4.8	Helicopter	LYNNE	CHAIR	CARE	4	Left
RELP 4.8	Helicopter	LYNNE	CHEEP	KEEP	4	Left
RELP 4.8	Helicopter	PETER	CHEEP	KEEP	4	Left
RELP 4.8	Helicopter	PETER	JAB	GAB	4	Left
RELP 4.8	Helicopter	PETER	JILT	GILT	4	Right
RELP 4.8	Helicopter	ALVA	JILT	GILT	4	Left
RELP 4.8	Helicopter	LYNNE	JILT	GILT	4	Left
RELP 4.8	Helicopter	PETER	JOT	GOT	4	Right
RELP 4.8	Helicopter	ALVA	SANK	THANK	4	Right
RELP 4.8	Helicopter	ALVA	SANK	THANK	4	Left
RELP 4.8	Helicopter	LYNNE	SING	THING	4	Right
RELP 4.8	Helicopter	ALVA	SING	THING	4	Right
RELP 4.8	Helicopter	WOLF	SING	THING	4	Right
RELP 4.8	Helicopter	PETER	SING	THING	4	Right
RELP 4.8	Helicopter	LYNNE	ZEE	THEE	4	Right
RELP 4.8	Helicopter	PETER	ZEE	THEE	4	Right
RELP 4.8	Helicopter	ALVA	BANK	DANK	5	Left
RELP 4.8	Helicopter	LYNNE	BANK	DANK	5	Left
RELP 4.8	Helicopter	PETER	BANK	DANK	5	Left
RELP 4.8	Helicopter	ALVA	FAD	THAD	5	Left
RELP 4.8	Helicopter	LYNNE	FAD	THAD	5	Left
RELP 4.8	Helicopter	LYNNE	FAD	THAD	5	Right
RELP 4.8	Helicopter	ALVA	FAD	THAD	5	Neither
RELP 4.8	Helicopter	LYNNE	FAD	THAD	5	Right
RELP 4.8	Helicopter	PETER	FAD	THAD	5	Right
RELP 4.8	Helicopter	PETER	FAD	THAD	5	Right
RELP 4.8	Helicopter	WOLF	FAD	THAD	5	Right
RELP 4.8	Helicopter	ALVA	FIN	THIN	5	Right
RELP 4.8	Helicopter	LYNNE	FIN	THIN	5	Right
RELP 4.8	Helicopter	LYNNE	FIN	THIN	5	Left
RELP 4.8	Helicopter	PETER	FIN	THIN	5	Right
RELP 4.8	Helicopter	LYNNE	FIN	THIN	5	Left

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 4.8	Helicopter	PETER	FIN	THIN	5	Left
RELP 4.8	Helicopter	PETER	FIN	THIN	5	Left
RELP 4.8	Helicopter	WOLF	FIN	THIN	5	Left
RELP 4.8	Helicopter	WOLF	FIN	THIN	5	Left
RELP 4.8	Helicopter	LYNNE	FOUGHT	THOUGHT	5	Right
RELP 4.8	Helicopter	ALVA	MET	NET	5	Right
RELP 4.8	Helicopter	LYNNE	MET	NET	5	Left
RELP 4.8	Helicopter	WOLF	MET	NET	5	Left
RELP 4.8	Helicopter	PETER	MET	NET	5	Left
RELP 4.8	Helicopter	ALVA	PEAK	TEAK	5	Left
RELP 4.8	Helicopter	PETER	PEAK	TEAK	5	Right
RELP 4.8	Helicopter	ALVA	PENT	TENT	5	Left
RELP 4.8	Helicopter	WOLF	PENT	TENT	5	Left
RELP 4.8	Helicopter	LYNNE	POOL	TOOL	5	Right
RELP 4.8	Helicopter	LYNNE	GILL	DILL	6	Right
RELP 4.8	Helicopter	LYNNE	GILL	DILL	6	Left
RELP 4.8	Helicopter	PETER	GILL	DILL	6	Right
RELP 4.8	Helicopter	PETER	GILL	DILL	6	Left
RELP 4.8	Helicopter	WOLF	GILL	DILL	6	Left
RELP 4.8	Helicopter	ALVA	KEG	PEG	6	Neither
RELP 4.8	Helicopter	ALVA	KEY	TEA	6	Neither
RELP 4.8	Helicopter	LYNNE	KEY	TEA	6	Left
RELP 4.8	Helicopter	LYNNE	KEY	TEA	6	Right
RELP 4.8	Helicopter	ALVA	KEY	TEA	6	Right
RELP 4.8	Helicopter	PETER	KEY	TEA	6	Right
RELP 4.8	Helicopter	ALVA	KEY	TEA	6	Right

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 9.6	None	LYNNE	BOND	POND	1	Left
RELP 9.6	None	LYNNE	DAUNT	TAUNT	1	Left
RELP 9.6	None	PETER	VOLE	FOAL	1	Left
RELP 9.6	None	WOLF	VOLE	FOAL	1	Left
RELP 9.6	None	PETER	FOO	POO	3	Right
RELP 9.6	None	LYNNE	SHAD	CHAD	3	Right
RELP 9.6	None	WOLF	SHAD	CHAD	3	Right
RELP 9.6	None	ALVA	SHAD	CHAD	3	Right
RELP 9.6	None	WOLF	SHAW	CHAW	3	Right
RELP 9.6	None	ALVA	SHAW	CHAW	3	Right
RELP 9.6	None	PETER	SHAW	CHAW	3	Right
RELP 9.6	None	WOLF	SHOES	CHOOSE	3	Right
RELP 9.6	None	ALVA	VILL	BILL	3	Right
RELP 9.6	None	PETER	VOX	BOX	3	Left
RELP 9.6	None	PETER	VOX	BOX	3	Right
RELP 9.6	None	LYNNE	VOX	BOX	3	Right
RELP 9.6	None	ALVA	JEST	GUEST	4	Left
RELP 9.6	None	PETER	JILT	GILT	4	Right
RELP 9.6	None	LYNNE	SANK	THANK	4	Right
RELP 9.6	None	ALVA	SING	THING	4	Right
RELP 9.6	None	LYNNE	SING	THING	4	Right
RELP 9.6	None	WOLF	SING	THING	4	Right
RELP 9.6	None	ALVA	BID	DID	5	Left
RELP 9.6	None	LYNNE	BID	DID	5	Left
RELP 9.6	None	LYNNE	FAD	THAD	5	Right
RELP 9.6	None	ALVA	FAD	THAD	5	Left
RELP 9.6	None	PETER	FIN	THIN	5	Right
RELP 9.6	None	WOLF	FORE	THOR	5	Right
RELP 9.6	None	LYNNE	FORE	THOR	5	Left
RELP 9.6	None	LYNNE	PENT	TENT	5	Left
RELP 9.6	None	LYNNE	GILL	DILL	6	Left

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 9.6	Armoured Car	LYNNE	VAST	FAST	1	Left
RELP 9.6	Armoured Car	ALVA	VOLE	FOAL	1	Left
RELP 9.6	Armoured Car	PETER	FENCE	PENCE	3	Right
RELP 9.6	Armoured Car	LYNNE	FENCE	PENCE	3	Right
RELP 9.6	Armoured Car	WOLF	FOO	POO	3	Right
RELP 9.6	Armoured Car	PETER	FOO	POO	3	Right
RELP 9.6	Armoured Car	WOLF	SHAD	CHAD	3	Right
RELP 9.6	Armoured Car	ALVA	SHAD	CHAD	3	Right
RELP 9.6	Armoured Car	ALVA	SHAW	CHAW	3	Right
RELP 9.6	Armoured Car	LYNNE	SHAW	CHAW	3	Right
RELP 9.6	Armoured Car	PETER	SHAW	CHAW	3	Right
RELP 9.6	Armoured Car	ALVA	SHEET	CHEAT	3	Right
RELP 9.6	Armoured Car	LYNNE	SHEET	CHEAT	3	Right
RELP 9.6	Armoured Car	WOLF	SHEET	CHEAT	3	Right
RELP 9.6	Armoured Car	ALVA	THAN	DAN	3	Right
RELP 9.6	Armoured Car	WOLF	THAN	DAN	3	Right
RELP 9.6	Armoured Car	LYNNE	VEE	BEE	3	Left
RELP 9.6	Armoured Car	ALVA	VILL	BILL	3	Right
RELP 9.6	Armoured Car	LYNNE	VILL	BILL	3	Right
RELP 9.6	Armoured Car	WOLF	VON	BON	3	Right
RELP 9.6	Armoured Car	ALVA	VON	BON	3	Right
RELP 9.6	Armoured Car	ALVA	VOX	BOX	3	Right
RELP 9.6	Armoured Car	PETER	VOX	BOX	3	Right
RELP 9.6	Armoured Car	WOLF	VOX	BOX	3	Left
RELP 9.6	Armoured Car	PETER	JILT	GILT	4	Right
RELP 9.6	Armoured Car	LYNNE	SANK	THANK	4	Right
RELP 9.6	Armoured Car	ALVA	SING	THING	4	Right
RELP 9.6	Armoured Car	PETER	SING	THING	4	Right
RELP 9.6	Armoured Car	LYNNE	SING	THING	4	Right
RELP 9.6	Armoured Car	PETER	ZEE	THEE	4	Right
RELP 9.6	Armoured Car	LYNNE	ZEE	THEE	4	Right
RELP 9.6	Armoured Car	WOLF	ZEE	THEE	4	Right
RELP 9.6	Armoured Car	ALVA	ZEE	THEE	4	Right
RELP 9.6	Armoured Car	PETER	FAD	THAD	5	Left
RELP 9.6	Armoured Car	PETER	FIN	THIN	5	Right
RELP 9.6	Armoured Car	WOLF	FIN	THIN	5	Right
RELP 9.6	Armoured Car	WOLF	FIN	THIN	5	Left
RELP 9.6	Armoured Car	PETER	FIN	THIN	5	Left
RELP 9.6	Armoured Car	ALVA	FORE	THOR	5	Right
RELP 9.6	Armoured Car	ALVA	FOUGHT	THOUGHT	5	Left
RELP 9.6	Armoured Car	WOLF	FOUGHT	THOUGHT	5	Right
RELP 9.6	Armoured Car	LYNNE	FOUGHT	THOUGHT	5	Left
RELP 9.6	Armoured Car	ALVA	MET	NET	5	Right
RELP 9.6	Armoured Car	ALVA	GILL	DILL	6	Left
RELP 9.6	Armoured Car	PETER	HOP	FOP	6	Right
RELP 9.6	Armoured Car	ALVA	KEY	TEA	6	Left
RELP 9.6	Armoured Car	PETER	KEY	TEA	6	Right

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 9.6	Helicopter	LYNNE	VAST	FAST	1	Left
RELP 9.6	Helicopter	PETER	VAST	FAST	1	Left
RELP 9.6	Helicopter	PETER	VOLE	FOAL	1	Left
RELP 9.6	Helicopter	ALVA	VOLE	FOAL	1	Left
RELP 9.6	Helicopter	ALVA	VOLE	FOAL	1	Right
RELP 9.6	Helicopter	WOLF	VOLE	FOAL	1	Right
RELP 9.6	Helicopter	PETER	MOAN	BONE	2	Right
RELP 9.6	Helicopter	ALVA	MOSS	BOSS	2	Right
RELP 9.6	Helicopter	PETER	MOSS	BOSS	2	Right
RELP 9.6	Helicopter	LYNNE	NEWS	DUES	2	Left
RELP 9.6	Helicopter	PETER	FENCE	PENCE	3	Right
RELP 9.6	Helicopter	LYNNE	FENCE	PENCE	3	Right
RELP 9.6	Helicopter	WOLF	FENCE	PENCE	3	Right
RELP 9.6	Helicopter	ALVA	FENCE	PENCE	3	Neither
RELP 9.6	Helicopter	LYNNE	FOO	POO	3	Right
RELP 9.6	Helicopter	WOLF	FOO	POO	3	Right
RELP 9.6	Helicopter	LYNNE	SHAD	CHAD	3	Right
RELP 9.6	Helicopter	WOLF	SHAD	CHAD	3	Right
RELP 9.6	Helicopter	PETER	SHAD	CHAD	3	Right
RELP 9.6	Helicopter	ALVA	SHAD	CHAD	3	Right
RELP 9.6	Helicopter	WOLF	SHEET	CHEAT	3	Right
RELP 9.6	Helicopter	LYNNE	SHEET	CHEAT	3	Right
RELP 9.6	Helicopter	LYNNE	SHEET	CHEAT	3	Left
RELP 9.6	Helicopter	PETER	SHEET	CHEAT	3	Left
RELP 9.6	Helicopter	WOLF	SHEET	CHEAT	3	Left
RELP 9.6	Helicopter	LYNNE	THONG	TONG	3	Left
RELP 9.6	Helicopter	ALVA	THOSE	DOZE	3	Left
RELP 9.6	Helicopter	PETER	THOSE	DOZE	3	Left
RELP 9.6	Helicopter	WOLF	THOSE	DOZE	3	Left
RELP 9.6	Helicopter	LYNNE	THOUGH	DOUGH	3	Left
RELP 9.6	Helicopter	PETER	THOUGH	DOUGH	3	Right
RELP 9.6	Helicopter	WOLF	THOUGH	DOUGH	3	Left
RELP 9.6	Helicopter	ALVA	VEE	BEE	3	Left
RELP 9.6	Helicopter	PETER	VEE	BEE	3	Left
RELP 9.6	Helicopter	LYNNE	VEE	BEE	3	Left
RELP 9.6	Helicopter	WOLF	VEE	BEE	3	Left
RELP 9.6	Helicopter	WOLF	VILL	BILL	3	Right
RELP 9.6	Helicopter	ALVA	VILL	BILL	3	Right
RELP 9.6	Helicopter	LYNNE	VON	BON	3	Right
RELP 9.6	Helicopter	ALVA	VON	BON	3	Left
RELP 9.6	Helicopter	PETER	VON	BON	3	Left
RELP 9.6	Helicopter	PETER	VON	BON	3	Right
RELP 9.6	Helicopter	ALVA	VON	BON	3	Neither
RELP 9.6	Helicopter	WOLF	VON	BON	3	Left
RELP 9.6	Helicopter	PETER	VOX	BOX	3	Right
RELP 9.6	Helicopter	LYNNE	VOX	BOX	3	Right
RELP 9.6	Helicopter	ALVA	VOX	BOX	3	Right
RELP 9.6	Helicopter	ALVA	CHEEP	KEEP	4	Left
RELP 9.6	Helicopter	PETER	CHEEP	KEEP	4	Left
RELP 9.6	Helicopter	WOLF	CHEEP	KEEP	4	Left
RELP 9.6	Helicopter	ALVA	CHEW	COO	4	Neither

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
RELP 9.6	Helicopter	ALVA	JAB	GAB	4	Left
RELP 9.6	Helicopter	WOLF	JAB	GAB	4	Left
RELP 9.6	Helicopter	LYNNE	JILT	GILT	4	Right
RELP 9.6	Helicopter	PETER	JILT	GILT	4	Right
RELP 9.6	Helicopter	LYNNE	SANK	THANK	4	Right
RELP 9.6	Helicopter	ALVA	SAW	THAW	4	Right
RELP 9.6	Helicopter	PETER	SING	THING	4	Right
RELP 9.6	Helicopter	LYNNE	SING	THING	4	Right
RELP 9.6	Helicopter	WOLF	SING	THING	4	Right
RELP 9.6	Helicopter	ALVA	SING	THING	4	Right
RELP 9.6	Helicopter	LYNNE	ZEE	THEE	4	Right
RELP 9.6	Helicopter	WOLF	ZEE	THEE	4	Right
RELP 9.6	Helicopter	PETER	ZEE	THEE	4	Right
RELP 9.6	Helicopter	ALVA	ZEE	THEE	4	Right
RELP 9.6	Helicopter	LYNNE	BANK	DANK	5	Left
RELP 9.6	Helicopter	PETER	BANK	DANK	5	Left
RELP 9.6	Helicopter	LYNNE	BOWL	DOLE	5	Left
RELP 9.6	Helicopter	LYNNE	FAD	THAD	5	Left
RELP 9.6	Helicopter	LYNNE	FAD	THAD	5	Right
RELP 9.6	Helicopter	WOLF	FAD	THAD	5	Right
RELP 9.6	Helicopter	LYNNE	FIN	THIN	5	Left
RELP 9.6	Helicopter	WOLF	FIN	THIN	5	Right
RELP 9.6	Helicopter	PETER	FIN	THIN	5	Right
RELP 9.6	Helicopter	WOLF	FIN	THIN	5	Left
RELP 9.6	Helicopter	PETER	FORE	THOR	5	Right
RELP 9.6	Helicopter	LYNNE	FORE	THOR	5	Right
RELP 9.6	Helicopter	ALVA	FORE	THOR	5	Right
RELP 9.6	Helicopter	LYNNE	FORE	THOR	5	Left
RELP 9.6	Helicopter	WOLF	FORE	THOR	5	Left
RELP 9.6	Helicopter	WOLF	MET	NET	5	Right
RELP 9.6	Helicopter	PETER	MET	NET	5	Left
RELP 9.6	Helicopter	WOLF	MOON	NOON	5	Right
RELP 9.6	Helicopter	LYNNE	POT	TOT	5	Left
RELP 9.6	Helicopter	ALVA	POT	TOT	5	Left
RELP 9.6	Helicopter	ALVA	WAD	ROD	5	Left
RELP 9.6	Helicopter	LYNNE	GILL	DILL	6	Left
RELP 9.6	Helicopter	ALVA	GILL	DILL	6	Right
RELP 9.6	Helicopter	LYNNE	HOP	FOP	6	Right
RELP 9.6	Helicopter	ALVA	KEY	TEA	6	Right
RELP 9.6	Helicopter	LYNNE	KEY	TEA	6	Right
RELP 9.6	Helicopter	ALVA	KEY	TEA	6	Left

SUMMARY OF ERRORS FOR DIAGNOSTIC RHYME TESTS

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
PERT 4.8	None	WOLF	VAULT	FAULT	1	Left
PERT 4.8	None	WOLF	VAULT	FAULT	1	Right
PERT 4.8	None	WOLF	MEAT	BEAT	2	Left
PERT 4.8	None	PETER	SHAD	CHAD	3	Right
PERT 4.8	None	ALVA	SHAD	CHAD	3	Right
PERT 4.8	None	LYNNE	SHAW	CHAW	3	Right
PERT 4.8	None	LYNNE	THEN	DEN	3	Left
PERT 4.8	None	PETER	THEN	DEN	3	Left
PERT 4.8	None	WOLF	THEN	DEN	3	Left
PERT 4.8	None	LYNNE	VEE	BEE	3	Left
PERT 4.8	None	PETER	VEE	BEE	3	Left
PERT 4.8	None	PETER	JILT	GILT	4	Right
PERT 4.8	None	WOLF	SANK	THANK	4	Right
PERT 4.8	None	ALVA	SANK	THANK	4	Right
PERT 4.8	None	LYNNE	SANK	THANK	4	Right
PERT 4.8	None	PETER	SANK	THANK	4	Right
PERT 4.8	None	LYNNE	ZEE	THEE	4	Right
PERT 4.8	None	ALVA	BONG	DONG	5	Right
PERT 4.8	None	LYNNE	FAD	THAD	5	Right
PERT 4.8	None	ALVA	FAD	THAD	5	Left
PERT 4.8	None	LYNNE	FAD	THAD	5	Left
PERT 4.8	None	PETER	FIN	THIN	5	Right
PERT 4.8	None	ALVA	FIN	THIN	5	Left
PERT 4.8	None	LYNNE	FIN	THIN	5	Left
PERT 4.8	None	PETER	FIN	THIN	5	Left
PERT 4.8	None	WOLF	FIN	THIN	5	Left
PERT 4.8	None	ALVA	FIN	THIN	5	Right
PERT 4.8	None	WOLF	FIN	THIN	5	Right
PERT 4.8	None	LYNNE	FIN	THIN	5	Right
PERT 4.8	None	LYNNE	MOON	NOON	5	Right
PERT 4.8	None	PETER	PENT	TENT	5	Right
PERT 4.8	None	WOLF	PENT	TENT	5	Right
PERT 4.8	None	PETER	GILL	DILL	6	Right
PERT 4.8	None	ALVA	KEY	TEA	6	Right
PERT 4.8	None	LYNNE	KEY	TEA	6	Right
PERT 4.8	None	PETER	KEY	TEA	6	Right
PERT 9.6	None	LYNNE	DINT	TINT	1	Right
PERT 9.6	None	LYNNE	SHEET	CHEAT	3	Left
PERT 9.6	None	ALVA	SHEET	CHEAT	3	Right
PERT 9.6	None	LYNNE	SHEET	CHEAT	3	Right
PERT 9.6	None	LYNNE	SHOES	CHOOSE	3	Right
PERT 9.6	None	ALVA	SHOES	CHOOSE	3	Right
PERT 9.6	None	ALVA	VILL	BILL	3	Right
PERT 9.6	None	LYNNE	VOX	BOX	3	Right
PERT 9.6	None	PETER	VOX	BOX	3	Right
PERT 9.6	None	ALVA	VOX	BOX	3	Right
PERT 9.6	None	ALVA	CHEEP	KEEP	4	Right
PERT 9.6	None	PETER	JEST	GUEST	4	Right
PERT 9.6	None	LYNNE	SING	THING	4	Right
PERT 9.6	None	WOLF	SING	THING	4	Right
PERT 9.6	None	PETER	SING	THING	4	Right

CODING	BACKGROUND NOISE	PERSON	LEFT WORD	RIGHT WORD	ATTRIBUTE	REPLY
PERT 9.6	None	ALVA	SING	THING	4	Right
PERT 9.6	None	LYNNE	BANK	DANK	5	Left
PERT 9.6	None	LYNNE	BID	DID	5	Left
PERT 9.6	None	PETER	BID	DID	5	Left
PERT 9.6	None	WOLF	FORE	THOR	5	Right
PERT 9.6	None	LYNNE	FORE	THOR	5	Left
PERT 9.6	None	LYNNE	FOUGHT	THOUGHT	5	Right
PERT 9.6	None	WOLF	FOUGHT	THOUGHT	5	Right
PERT 9.6	None	ALVA	FOUGHT	THOUGHT	5	Right
PERT 9.6	None	PETER	FOUGHT	THOUGHT	5	Right
PERT 9.6	None	ALVA	POT	TOT	5	Right