

AD-A096 092

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST
DEC 80 J J WOLF, K D FIELD, W H RUSSELL

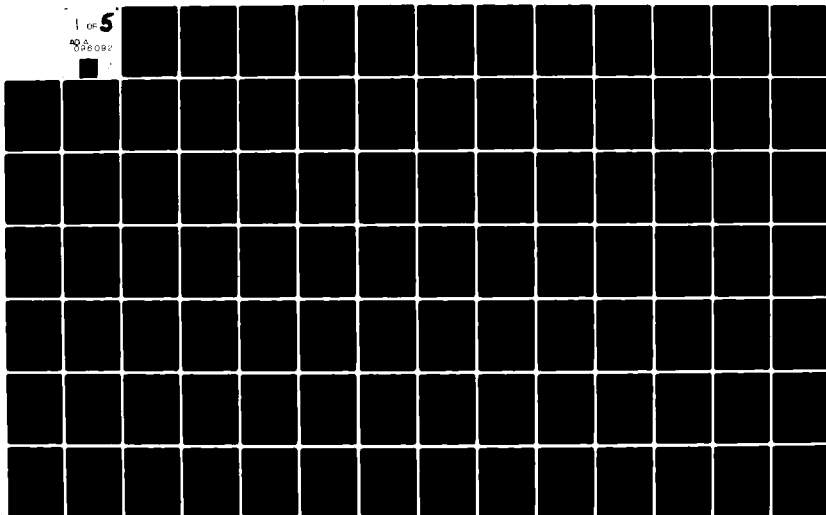
F/G 17/2.1
APC CODER FOR S--ETC(U)
DCA100-79-C-0037

UNCLASSIFIED

BBN-4565-VOL-2

NL

1 OF 5
DA
000000



Bolt Beranek and Newman Inc.



AD A096092

LEVEL

12

Report No. 4565

10/1/80
J/S/W/...

**Design and Real-Time Implementation of a Robust APC Coder
for Speech Transmission Over 16 Kb/s Noisy Channels.**

Volume II: Real-Time Implementation.

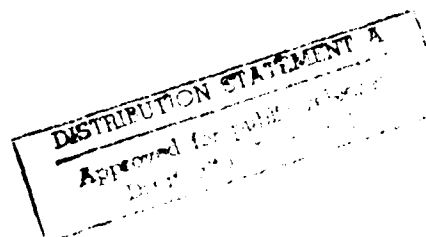
Final Report.

DTIC

MAR 9 1981

December 1980

Prepared for:
Defense Communications Agency



DDC FILE COPY

81 2 00 075

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 4565, Vol. II	2. GOVT ACCESSION NO. AD-A696 092	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR SPEECH TRANSMISSION OVER 16 Kb/s NOISY CHANNELS Vol. II: Real-Time Implementation		5. TYPE OF REPORT & PERIOD COVERED Final Report July 1979 - December 1980
7. AUTHOR(s) J.J. Wolf, K.D. Field, W.H. Russell		6. PERFORMING ORG. REPORT NUMBER BBN Report No. 4565
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02238		8. CONTRACT OR GRANT NUMBER(s) DCA100-79-C-0037 ✓
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE December 1980
		13. NUMBER OF PAGES 417
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Defense, for sale to the general public.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Speech coding, 16 kb/s speech transmission, adaptive predictive coding, digital voice terminal, real-time speech coder, medium-bandwidth speech transmission, array processor		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the design and development of a real-time robust APC speech coder that transmits high-quality speech over a 16 kb/s channel with bit-error rates of up to 1%. This volume (Vol. II) describes the implementation of the optimized adaptive predictive coding algorithm as a full-duplex real-time speech coder on the CSP Inc. MAP-300 array processing computer. The report documents in detail the MAP-300 program, and it includes complete listings of the real-time coder program.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4565

DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER
FOR SPEECH TRANSMISSION OVER 16 Kb/s NOISY CHANNELS

Final Report

Volume II: Real-Time Implementation

Authors: J. Wolf, K. Field, W. Russell

December 1980

Transmission For	
GR&I	☒
IAB	☐
Unannounced	☐
Notification	☐
by	
Distribution/	
Availability Codes	
Dist	Special
A	

Prepared for:
Defense Communications Agency

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
2. REAL-TIME IMPLEMENTATION	3
2.1 OVERVIEW	3
2.1.1 System Function	3
2.1.2 System Components	3
2.1.3 System Design	4
2.2 SYSTEM OPERATION	9
2.2.1 Transmitter	9
2.2.1.1 ADAM Scroll Program (ADPROG)	12
2.2.1.2 A/D Interrupt Service (ADAMINT)	15
2.2.1.3 ANALYZE Module	17
2.2.1.4 TMODEM Interrupt Service (TMODEMINT)	27
2.2.1.5 TMODEM and RMODEM I/O Scroll Program (RTPROG)	29
2.2.1.6 Transmitter Input/Output Buffer Initial Sequence	32
2.2.2 Receiver	33
2.2.2.1 RMODEM Scroll Program	36
2.2.2.2 RMODEM Interrupt Service (RMODEMINT)	36
2.2.2.3 SYNTHESIZE Module	45
2.2.2.4 D/A Interrupt Service Routine (AOMINT)	50
2.2.2.5 D/A Scroll Program (DAPROG)	50
2.2.2.6 Receiver Input/Output Buffer Initial Sequence	52
2.3 SYSTEM HARDWARE	54
2.3.1 MAP-300 Hardware	55
2.3.2 Audio and Modem Interface Hardware	55
2.4 SYSTEM SOFTWARE	56
2.4.1 MAP-300 Software Components	57
2.4.1.1 CSPI-Supplied MAP-300 Software	57
2.4.1.2 BBN-written MAP-300 Software	59
2.4.2 PDP-11 Software Components	60
2.4.2.1 CSPI-Supplied PDP-11 Software	61
2.4.2.2 BBN-written PDP-11 Software	61

2.5	SYSTEM TIMING PERFORMANCE	67
3.	SOFTWARE OPERATING PROCEDURES	71
3.1	SOFTWARE EXECUTION PROCEDURE	71
3.1.1	Optional Software Execution Procedure	73
3.1.2	Analog Input Level Setting	74
3.2	SOFTWARE INSTALLATION PROCEDURE	75
4.	REFERENCES	79
APPENDIX A.	BBN-WRITTEN FUNCTION DESCRIPTIONS	81
APPENDIX B.	MAP-300 BUFFERS	110
APPENDIX C.	MAP-300 SCALARS	126
APPENDIX D.	PROGRAM LISTINGS	138

LIST OF FIGURES

FIG. 1.	SYSTEM COMPONENTS	5
FIG. 2.	SYSTEM STRUCTURE	6
FIG. 3.	BACKGROUND PROCESS LOOP	8
FIG. 4.	TRANSMITTER PROCESS	10
FIG. 5.	ADPROG: ADAM SCROLL PROGRAM	14
FIG. 6.	A/D INTERRUPT SERVICE ROUTINE	16
FIG. 7.	TMODEM INTERRUPT SERVICE ROUTINE (TMODEMINT)	30
FIG. 8.	TMODEM AND RMODEM SCROLL PROGRAM FLOW CHART	31
FIG. 9.	RECEIVER PROCESS	34
FIG. 10.	RMODEM INTERRUPT SERVICE MODULE (RMODEMINT)	38
FIG. 11.	FRAME SYNCHRONIZATION INITIALIZATION MODULE (SYNCINIT)	41
FIG. 12.	FRAME SYNCHRONIZATION SEARCH MODULE (SYNCSRCH)	42
FIG. 13.	FRAME SYNCHRONIZATION UPDATE MODULE (SYNCUPDATE)	44
FIG. 14.	D/A INTERRUPT SERVICE ROUTINE (AOMINT)	51
FIG. 15.	DAPROG: AOM SCROLL PROGRAM	53
FIG. 16.	MAP-300 MEMORY ORGANIZATION	58
FIG. 17.	MAP-300 TIMING	69

Bolt Beranek and Newman Inc.

Report No. 4565

LIST OF TABLES

TABLE 1.	TSINK BUFFER FORMAT	27
TABLE 2.	TBITS BUFFER FORMAT	28

1. INTRODUCTION

The purpose of this project was the design and development of a real-time speech coding system that transmits high-quality speech at a data rate of 16 kb/s (kilobits/second). The final report of this project is organized in two volumes. Volume I describes our work in the development and optimization of the speech coding algorithm. Volume II (this volume) describes the implementation of the final optimized speech coding algorithm as a real-time full duplex system on a CSP Inc. (CSPI) MAP-300 signal processing computer and associated hardware.

In the body of this volume we present detailed documentation of the MAP-300 real-time implementation of the speech coding algorithm (Section 2) and instructions on the installation and use of the speech coder software (Section 3). Contained in appendices are function descriptions of BBN-written MAP-300 modules (Appendix A), documentation of the MAP-300 buffers and scalars (Appendices B and C), and listings of the MAP-300 and PDP-11 (FORTRAN) programs that constitute the speech coder system (Appendix D).

Bolt Beranek and Newman Inc.

Report No. 4565

2. REAL-TIME IMPLEMENTATION

2.1 OVERVIEW

This section describes the function, components, and design of the BBN16 real-time speech coder implementation. The later sections in this chapter describe in more detail the operation of the speech coder system, the hardware and software used to construct it, and the real-time performance of the completed system.

2.1.1 System Function

The speech coder system is a full duplex terminal of a complete communication system. It is intended to be connected via a 16 kb/s digital I/O link through a communication channel to an identical system. The speech coder system functions simultaneously as both a transmitter and a receiver.

2.1.2 System Components

The speech coder system contains both hardware and software elements. The hardware elements include a CSP Inc. MAP-300 array processor attached to a PDP-11, a handset including microphone and earphone, a hookswitch, amplifiers and low-pass filters, and digital line interfaces. The software elements include MAP-300 programs, which comprise the real-time software, and the program that runs on the PDP-11, which is used only to load and

initialize the MAP software. Fig. 1 is a block diagram of the system.

2.1.3 System Design

The real-time speech coder consists of six separate foreground processes. The Transmitter requires an analog-in process, an analysis process, and a digital-out process. The Receiver requires a digital-in process, a synthesis process, and an analog-out process. Since all of these processes make use of the MAP-300 CSPU to some extent, a mechanism for scheduling them is necessary. Part of this mechanism is contained in the MAP hardware interrupt structures and the SNAP-II executive program. The rest is implemented using flags in conjunction with a background process running in the CSPU. Fig. 2 is a diagram of these processes. The processes share data buffers and communicate the status of these buffers through flags. Since the four I/O processes are continuous, pairs of double buffers are used to enable the necessary sharing.

Each I/O process includes an interrupt service routine, which handles the flags and transfers data to or from the shared buffers. Two levels of double buffering are provided within each I/O process to maintain acceptable system performance under real-time exception conditions.

The background process is shown in Fig. 3. The ANALYZE and

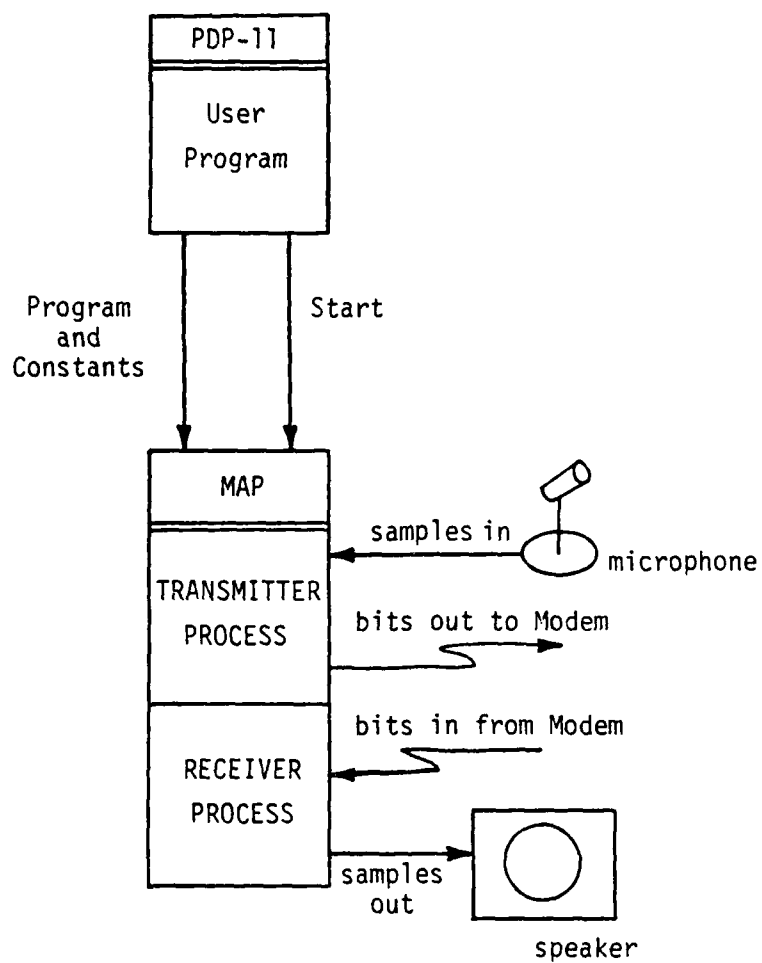


FIG. 1. SYSTEM COMPONENTS

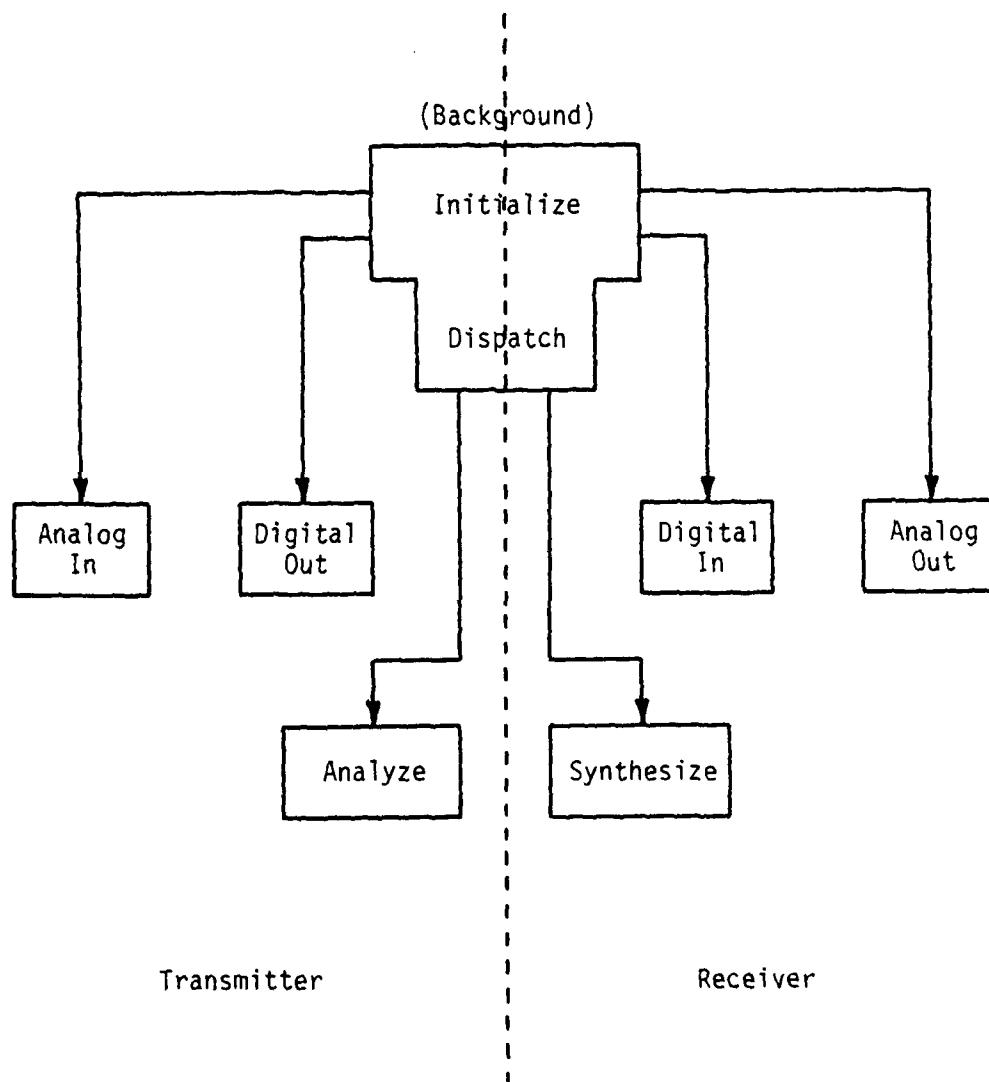


FIG. 2. SYSTEM STRUCTURE

SYNTHESIZE modules are logically asynchronous; they respectively belong to the independent Transmitter and Receiver. However, the modules must be controlled by a sequential machine, the CSPU. The control strategy for allowing each module as much flexibility as possible results in a structure that executes a module only if that module's input and output buffers are available.

The Initialization module is the first module executed, and it is executed only once. It sets all buffer flags to indicate empty, loads and starts the programs in the various I/O Scroll processors, and enables interrupts from these processors.

Control then passes to the basic loop of the background process. This loop executes the ANALYZE and SYNTHESIZE modules when the required I/O buffers are available. For example, the SYNTHESIZEA module will be executed only when the RBITSB buffer is full and the RSINKA buffer is empty.

The ANALYZEA and ANALYZEB modules are functionally equivalent, differing only in their input and output buffers. The structure of the SNAP-II programming environment (specifically, the use of prebound functions for run-time efficiency) does not allow changing the buffers used in a particular function. Therefore, two separate modules have been created to deal with the two sets of TSOURCE and TBITS buffers, A and B. Similarly, two separate SYNTHESIZE modules are required.

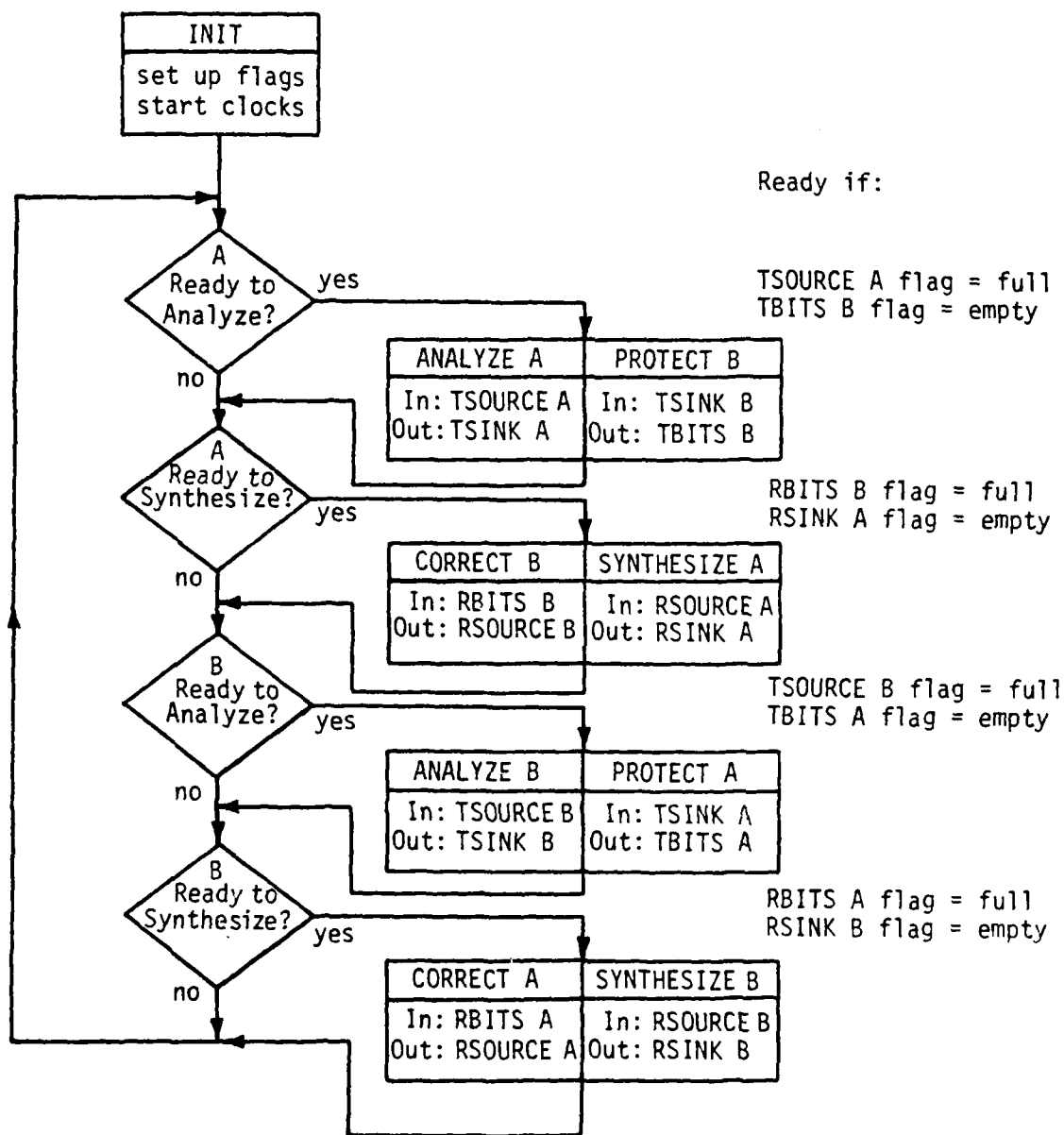


FIG. 3. BACKGROUND PROCESS LOOP

All of the foreground processes are loosely coupled. Execution of neither the ANALYZE nor the SYNTHESIZE module is connected rigidly to any I/O process.

2.2 SYSTEM OPERATION

The speech coder system functions as one terminal of a full duplex digital voice connection. It accepts voice input, digitizes it, and processes it. The processed speech is transmitted as a sequence of bits to a similar terminal. The system also accepts a sequence of bits representing speech transmitted from a remote terminal and processes this sequence to obtain synthetic voice output.

2.2.1 Transmitter

The Transmitter is shown in Fig. 4. The low-pass filtered input speech is fed into an A/D converter contained in an I/O Scroll processor (the ADAM, or Analog Data Acquisition Module), a component processor of the MAP. The program running in this scroll controls the sampling of the speech data, transferring 216 samples (one frame) into each of two buffers, alternately. The sampling rate is 6.621 kHz, so each frame is about 32.625 ms long.

When the ADAM fills one of these buffers, it generates an interrupt to the main MAP processor, the CSPU. The A/D Interrupt

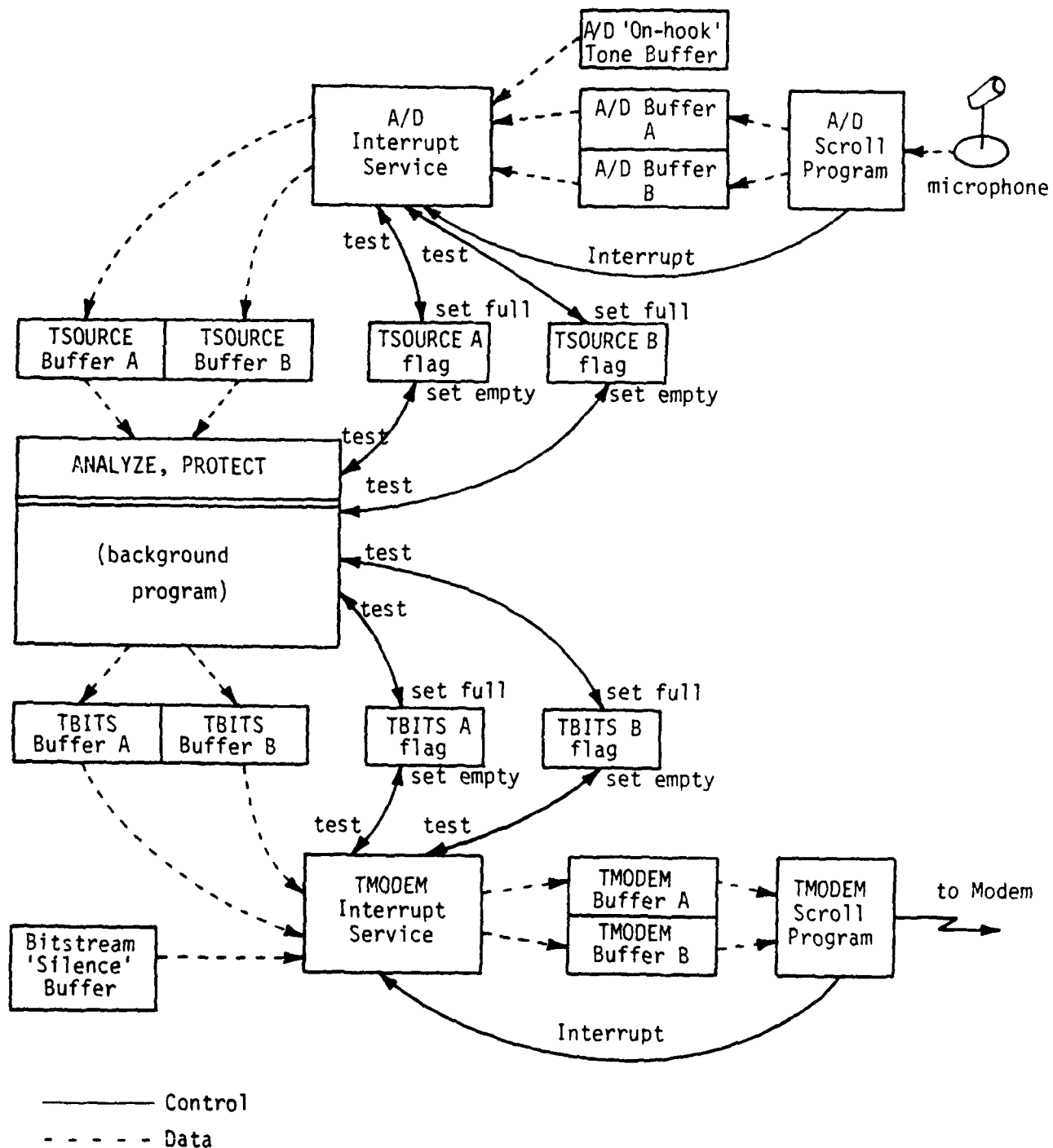


FIG. 4. TRANSMITTER PROCESS

Service routine, which is activated by this interrupt, transfers the data from the just-filled A/D buffer to the current one of the two TSOURCE buffers (if the current one is empty) and sets the corresponding TSOURCEA or TSOURCEB flag to indicate that the buffer is now full. If the current TSOURCE Buffer is not empty, the A/D buffer contents are discarded.

The data in the TSOURCE buffers is used as input by the ANALYZE program module. The Background Process (Section 2.1.3) tests the TSOURCE flags to determine when data is available to the ANALYZE module. After processing a TSOURCE buffer, ANALYZE clears the corresponding flag to indicate that the buffer is empty. The ANALYZE module processes the data, producing coded speech, which is then written into one of the (empty) TSINK buffers. The ANALYZE module runs in the CSPU and AP at background level and is therefore asynchronous with the interrupt service routines.

After ANALYZE fills a TSINK buffer, the PROPAR and PRORES modules transform the data from this buffer into an error-protected bitstream. This bitstream is stored in one of the (empty) TBITS buffers, and the corresponding TBITS flag is set to indicate full. The PROPAR/PRORES operation logically follows ANALYZE, in that data must be processed by ANALYZE before it is available to PROPAR/PRORES. However, the PROPAR and PRORES modules are actually executed concurrently with the next

execution of the ANALYZE module. In fact, they appear as part of the ANALYZE function list. The first time the ANALYZE and PROPAR/PRORES modules are executed, PROPAR/PRORES will operate on a TSINK buffer before it has been filled by ANALYZE; therefore, the TSINK buffers are initialized to coded silence.

The data processed by the PROPAR/PRORES modules will eventually be output to the modem by the TMODEM program, running in another I/O Scroll processor. This program takes data, in the form of a bitstream, from the TMODEM buffers and puts it out to the modem. When the TMODEM buffer is emptied, the TMODEM scroll program generates an interrupt to the CSPU and begins taking data from the other TMODEM buffer.

This interrupt causes execution of the TMODEM Interrupt Service routine. This routine transfers data from a full TBITS buffer to the just-emptied TMODEM buffer and clears the corresponding TBITS flag to indicate empty. If the TBITS buffer is not full, a buffer corresponding to coded silence is transferred to the TMODEM buffer.

2.2.1.1 ADAM Scroll Program (ADPROG)

Speech input is performed by the ADAM, an IOS-2 scroll processor that contains an analog multiplexer, sample/hold, and A/D converter. The ADAM receives both the input speech signal and the input sampling clock from the Speech Processor Interface (SPI). The signal is in the range -5 to +5 volts, and the sampling rate is 6.621 kHz.

The ADAM program (ADPROG) is shown in Fig. 5. The speech samples are double-buffered in A/D input buffers named TADBA and TADBB. (All MAP buffers and most scalars are prefixed with "T" or "R" to indicate that they are used in the speech coder Transmitter or Receiver respectively.) The A/D input buffers are 216 samples long, which corresponds to a frame length of about 32.625 ms. The A/D samples are written into memory in short (16-bit) floating point format. (The ADAM and AOM are capable of working only in 16-bit floating or 16-bit fixed point formats.)

ADPROG sets the ADAM multiplexer to Channel 1 and sets the F1 flag to initiate sampling. ADPROG maintains a pointer into the A/D input buffer, which is initialized to TADBA-1. When a sample is converted, this address pointer is incremented and used to transfer the A/D sample to MAP memory. The address pointer is compared to the buffer end address; when the end has been reached, a line 1 interrupt is passed to the CSPU to signal the filling of the buffer, and the SYNCSTOP register is checked. SYNCSTOP is a register that is set by the CSPU to signal the ADAM to stop sampling. In normal operation, it remains clear, so ADPROG resets the address pointer to the other A/D input buffer (TADBB), and speech sample input proceeds without interruption. Interrupt 1 from the ADAM is used to signal the filling of each successive A/D buffer and the switch to the other A/D buffer.

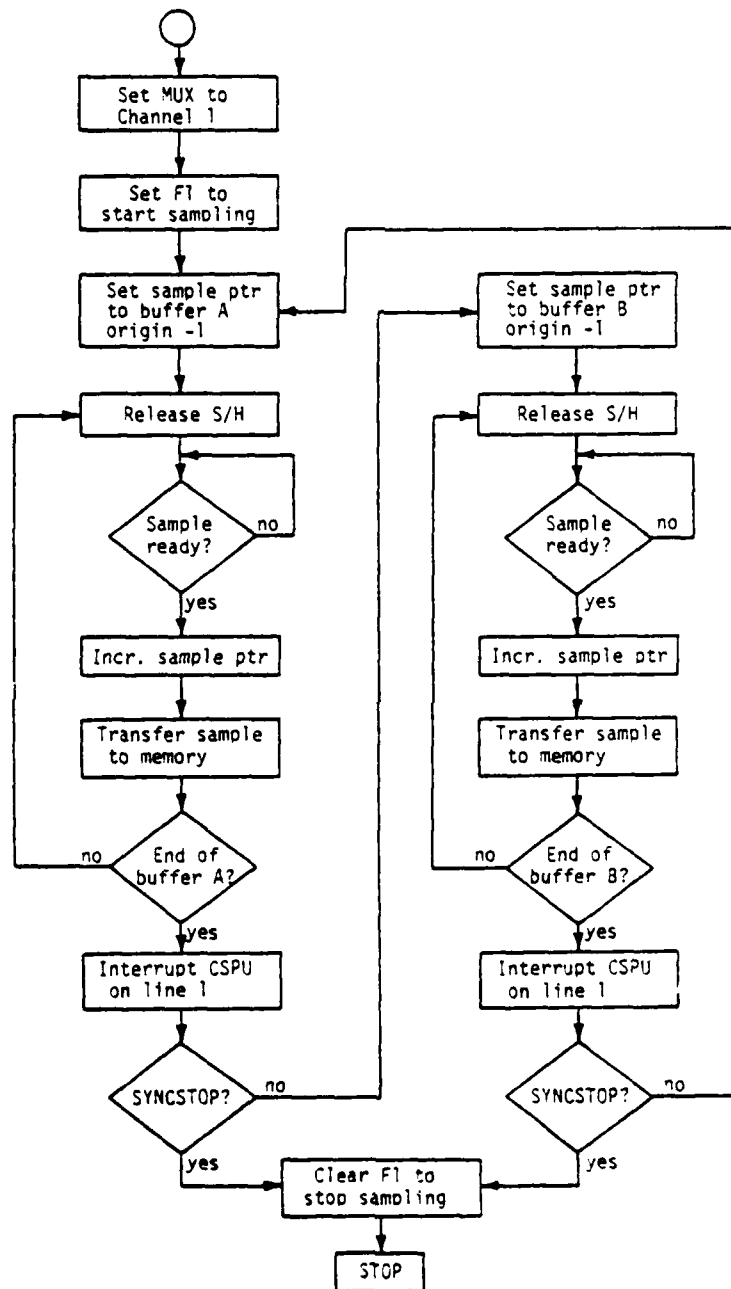


FIG. 5. ADPROG: ADAM SCROLL PROGRAM

2.2.1.2 A/D Interrupt Service (ADAMINT)

ADAMINT is activated by each ADAM line 1 interrupt, signifying the filling of another A/D input buffer. ADAMINT, like the other three speech coder input/output interrupt service routines, maintains a pair of integer scalar "pointer offsets" to keep track of its input and output buffer relationships. These pointer offsets, which take on the values -2 and 0, are used to reference small tables of address pointers. Thus, for example, ADPO ("A/D Pointer Offset") references ADBPTR, a table of pointers to the two A/D input buffers, and TSRPO ("TSOURCE Pointer Offset") references the tables TSRBPTR and TSRFPTR, which point to TSOURCE buffer-copying subroutines and buffer-status flags.

The operation of ADAMINT is shown in Fig. 6. When it receives a buffer-filled (line 1) interrupt, ADAMINT switches ADPO to the buffer just filled by the ADAM. Then ADAMINT checks the status of the next TSOURCE buffer to be filled; if it is available (empty) and the handset hookswitch is up (indicating that the handset is in service), ADAMINT copies the new A/D buffer to the current TSOURCE buffer. If the TSOURCE buffer is empty but the hookswitch is down, a preconstructed "tone" buffer is copied instead to the TSOURCE buffer. (This simulated input is intended to indicate to the person using the remote speech coder that this coder is in operation, but the handset is resting in the holder.) In either case, the TSOURCE buffer flag is set

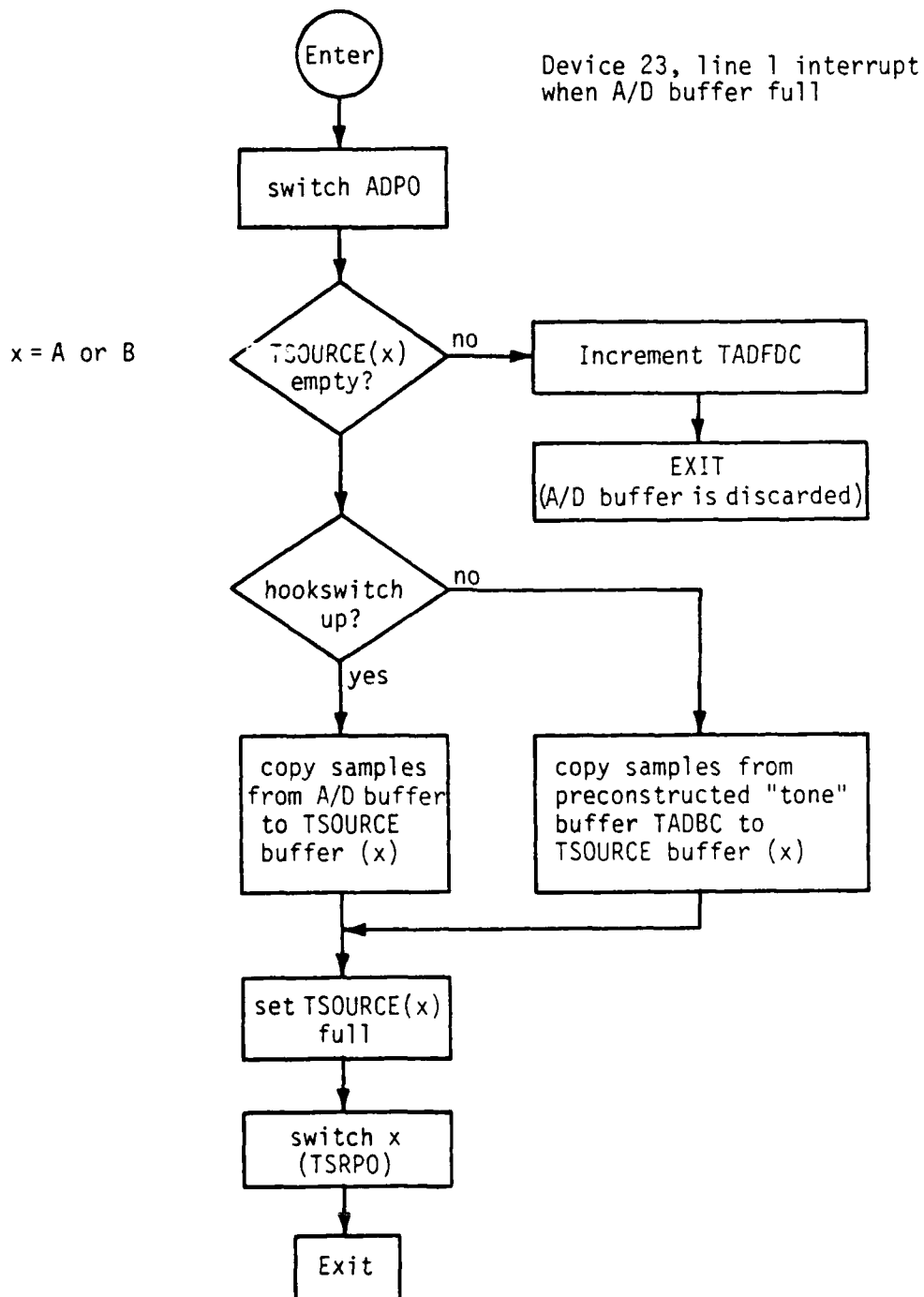


FIG. 6. A/D INTERRUPT SERVICE ROUTINE

to nonzero to indicate full, TSRPO is switched, and the routine exits. If, on the other hand, the TSOURCE flag shows that the current TSOURCE buffer is not empty and therefore unavailable, ADAMINT simply exits, effectively discarding the new A/D data. An "A/D Frame Discard Counter" (TADFDC) is incremented by one to keep track of the fact that an A/D buffer was discarded.

2.2.1.3 ANALYZE Module

The ANALYZE Module implements the APC analysis algorithm described in Section 2 of the "Specification of the Optimized 16 kb/s APC Algorithm" (henceforth known as the "Algorithm Specification") (Volume I, Appendix A of this report). As described in Section 2.1.3, two versions of the ANALYZE Module exist, one for each of the two sets of input/output buffers required for double buffering. Each version consists of a function list of MAP-300 function calls. The two versions (ANLZA and ANLZB) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions.

This section describes, on a function by function basis, the ANLZA/ANLZB function lists, which appear in subroutine BBN16F of the speech coder software (Appendix D). The specification of alternative parameters for the two function lists is indicated in the descriptions below by a '/' (e.g., 'TSRA/B' indicates TSRA for the ANLZA function list and TSRB for the ANLZB function

list). Each function is either supplied by CSPI as part of the SNAP-II Release 3.5 software system or written by BBN. CSPI-supplied functions are described in detail in CSPI documents [2,3,4]. BBN-written functions are described in detail in Appendix A of this report. The buffers and scalars passed as parameters to the functions are defined in subroutine BBN16C of the speech coder software (Appendix D) and are described in detail in Appendices B and C.

2.2.1.3.1 Begin Function List

MPBFL(ANLZA/B)

This CSPI function specifies the start of the indicated function list.

2.2.1.3.2 Set G-flag

MPGSC(IG3,ISET)

This BBN function causes general-purpose flag G3 to be set, indicating the start of ANLZ processing. This use of flag G3 is intended for system debugging, timing, and internal measurement and is not required for proper operation of the speech coding software.

2.2.1.3.3 Monitor Peak Input Level

ADPEAK(TSRA/B)

This BBN function serves to maintain a running maximum (in integer scalar TADPK) of the digitized speech input samples, allowing the user to properly adjust the analog input signal level for maximum A/D range without

clipping. The maximum sample in the current input buffer is compared to the previous maximum, and TADPK is updated if necessary. TADPK is displayed (and reset to zero) during speech coder operation in response to a "T" command.

2.2.1.3.4 Transmitter Frame History Updates

THIST(TSP1,TSP0,TE1,TE11,TRH,TRH0)

This BBN function performs frame history updates on three buffers. The last-frame data from the end of each buffer is moved to the beginning of the buffer for use during current frame processing. (The buffers are referenced by TSP1, TE1, and TRH. The end portions of these buffers are referenced by TSP0, TE11, and TRH0.)

2.2.1.3.5 Preemphasis

DPRE(TSP0,TALPH,TSRA/B,TPHST,TONES)

This CSPI function (Discrete Weighted Pre-emphasis Filter) preemphasizes the input speech (TSRA/TSRB) (as indicated in Section 2.1 of the Algorithm Specification), with filter coefficient TALPH, filter history TPHST, unity weighting vector TONES, and output preemphasized speech TSP0.

2.2.1.3.6 Done with Input

MPWT(PCSR,AP)
MPIST(TSRFA/B,0)

These CSPI functions clear the appropriate buffer status flag (TSRFA or TSRFB) once the previous function has completed, indicating that the associated buffer can now be considered empty (i.e., ready to be filled with new input speech data).

2.2.1.3.7 Pitch Filter Analysis

The functions in this section perform pitch filter analysis on the preemphasized input speech, as indicated in Section 2.2 of the Algorithm Specification.

SSUM(TDCN,TSP,TFSZI)

This CSPI function sums the extended frame of preemphasized input speech samples (TSP), and multiplies the sum by the negative reciprocal of the extended frame size (TFSZI), producing the negative of the extended frame DC value (TDCN).

VMUL(TWSP,1,TSP,TDCN,THAM0,0)

This CSPI function removes the DC term (TDCN) from the input speech (TSP) and multiplies the result by a Hamming window (THAM0), producing windowed speech in TWSP.

FFTLR(TXSP,1,TWSPZ,TCOST,TXSP)

This BBN function is a slightly modified version of the CSPI function FFTNR (Real to complex FFT, not in place) (described in [4]). The modification involves a reformatting of the complex frequency domain output (TXSP) such that the last real output point appears at the end of the array, rather than in the first imaginary output location (which is known to be zero). The real input (TWSPZ) is a zero-padded copy of the windowed speech in TWSP. TCOST is a pregenerated table of cosine values.

VMSQ(TPSPR,TXSPR,TXSPI)

This CSPI function computes the power spectrum (TPSPR) of the windowed speech by computing the magnitude squared from the DFT real (TXSPR) and imaginary (TXSPI) parts.

VCLR(TPSPI)

This CSPI function clears the imaginary part of the complex buffer containing the real power spectrum. This clear operation must be performed every frame because the following inverse FFT function writes to TPSPI.

FFILR(TRP,1,TPSP,TCOST,TRP)

This BBN function is a slightly modified version of the CSPI function FFINR (Complex to real inverse FFT, not in place) (described in [4]). The modification involves reformatting of the complex frequency domain input (TPSP) corresponding to the modification described above for FFTLR. TPSP is the complex input consisting of real part TPSPR and imaginary part TPSPI. The real output TRP contains the autocorrelation coefficients of the windowed speech.

PITCH(TRPP,TMH,TRP,TIM,T14,TC2P)

This BBN function computes and codes the pitch lag (in terms of number of speech samples) from the autocorrelation coefficients (TRP). The pitch lag is output in TMH, and the coded pitch is output as an integer half-word in the left half of real scalar TIM. -TRP(pitch lag -1), -TRP(pitch lag), and -TRP(pitch lag +1) are output (for the three-tap pitch predictor coefficient computation) in TRPP. The single-tap pitch predictor coefficient is output in real scalar TC2P.

MEWL(TC,TINC,TRPP,TKTHR,TRP,TWORK)

This CSPI function (described in [4]) computes the three-tap pitch predictor coefficients (TC) from the pitch-centered autocorrelation coefficients (TRPP).

STAB(TCH,TC2P,TIC,TC)

This BBN function performs a stability check on the three-tap pitch predictor coefficients (TC) and uses the

single-tap predictor (with zeros for the first and third coefficients, TC2P as the second coefficient) if the three-tap coefficients are found to be unstable. The pitch predictor coefficients are then coded and quantized into buffers TIC and TCH, respectively.

2.2.1.3.8 Inverse Pitch Filter

INVPF (TE10, TMH, TSPl, TCH)

This BBN function inverse filters the preemphasized speech (TSPl) (as indicated in Section 2.3 of the Algorithm Specification), using quantized pitch predictor coefficients (TCH) and pitch lag (TMH) and produces a first residual signal (TE10).

2.2.1.3.9 Spectral Filter Analysis

The functions in this section perform spectral filter analysis on the first residual signal, as indicated in Section 2.4 of the Algorithm Specification.

VMUL (TWE1, 1, TE10, 0, THAM1, 0)

This CSPI function multiplies the first residual signal (TE10) by a Hamming window (THAM1), producing a windowed first residual (TWE1).

DCORZ (TRS, 0, TWE1, TWE1)

This CSPI function (described in [4]) computes seven autocorrelation coefficients (TRS) from the windowed first residual (TWE1).

HFC (TRSP, TRS, TLMU)

This BBN function modifies the autocorrelation coefficients (TRS), producing high-frequency-corrected autocorrelation coefficients (TRSP).

MWLQ6 (TKE,TKTHR,TKH,TRSP,TIK)

This BBN function is a modified version of the CSPI function MWLD. It uses the Levinson recursion to compute the spectral reflection coefficients from the corrected autocorrelation coefficients (TRSP), and then codes and quantizes them into buffers TIK and TKH.

2.2.1.3.10 Inverse Spectral Filter

The functions in this section perform inverse filtering of the first residual signal, as indicated in Sections 2.4.5 and 2.5 of the Algorithm Specification.

VKTOA (TAH,TKH)

This BBN function converts reflection coefficients (TKH) to spectral predictor coefficients (TAH).

DCORZ (TE2,0,TE1,TAHR)

This CSPI function (described in [4]) inverse filters the first residual (TE1) by performing a correlation operation with the pitch predictor coefficients in reverse order (TAHR) (hence a convolution), producing a second residual signal (TE2).

2.2.1.3.11 Error Protection on Previous Frame Parameter Data

PROPAR (B/A)

This BBN function performs error protection on previous frame parameter data (in buffer TSNKB/TSNKA), depositing error protected and bitstreamed transmission data in buffer TBTA/TBTA. This function executes in the CSPU while the previous function (DCORZ) is executing in the AP. (See Section 2.2.1.3.18 for a more detailed discussion of this module.)

2.2.1.3.12 Noise Shaping Filter Calculation

VMUL(TANS,1,TAH,0,TFAC,0)

This CSPI function computes noise shaping filter coefficients (TANS) as indicated in Section 2.4.6 of the Algorithm Specification.

2.2.1.3.13 Gain Factor Calculation

GAIN(TGFAC,TIG,TIDG,T72R,TE2,T3R)

This BBN function computes and quantizes energy and segment delta-gains as indicated in Section 2.6 of the Algorithm Specification. TIG is the coded energy of the second residual (TE2). TIDG contains the three coded segment delta-gains, and TGFAC contains the three residual quantizer segment scale factors.

The quantization table for energy differs from that in the Algorithm Specification by a scale factor of $2.511 \times (10^{-7})$ (-66 dB), which accounts for the difference in input speech signal magnitude between the simulation (+1024) and the real-time implementation (+0.5). (The ADAM and AOM have a range of ± 1 , but that amplitude is rare.) The energy and delta-gain decoding tables differ from their quantization tables by a square root. Quantization is performed on the computed energies, but the APC and synthesis operations use gain (or square root of energy) factors. The gain decoding table also differs from that in the Algorithm Specification by -66 dB (scale factor of $5.01 \times (10^{-4})$).

2.2.1.3.14 APC Residual Calculation

APC(TIUA/B,TMH,TSP0,TFILT,TGFAC,TRH,TVH,TQ1)

This BBN function computes and codes the APC residual as indicated in Section 2.7 of the Algorithm Specification. The input preemphasized speech is in TSP0. The spectral, pitch, and noise shaping filter coefficients are contiguous in buffer TFILT. TMH

contains the pitch lag. TRH, TVH, and TQ1 retain frame-to-frame memories of the filters. The coded APC residual is stored in buffer TIUA/TIUB (equivalent to the beginning of TSNKA/TSNKB).

2.2.1.3.15 Error Protection on Previous Frame Residual Data

PRORES(B/A)
MPIST(TBTFB/A,1)

These functions (PRORES is a BBN function; MPIST is a CSPI function) perform bitstreaming on the previous frame's residual data (in buffer TSNKB/TSNKA), depositing bitstreamed transmission data in buffer TBTB/TBTA. The appropriate buffer status flag (TBTFB or TBTFA) is set, indicating that the associated buffer is now full (i.e., ready to be transmitted). These functions execute in the CSPI while the previous function (APC) is executing in the AP. (See Section 2.2.1.3.18 for a more detailed discussion of this module.)

2.2.1.3.16 Transmitter Data Collection

GTHR(TSNKA/B,TIG,TIDG,TIM,TIC,TIK)

This BBN function collects all coded parameter transmission data into a single buffer (TSNKA/TSNKB). (The coded residual data has been placed directly into TSNKA/TSNKB by the APC function).

2.2.1.3.17 End of Function List

MPEFL(ANLZA/B)

This CSPI function specifies the end of the indicated function list.

2.2.1.3.18 Error Protection and Bitstreaming

The PROPAR and PRORES modules take as input a TSINK buffer

(either TSNKA or TSNKB) containing quantized and coded analysis parameters and residual samples and produce as output a TBITS buffer (again, TBTA or TBTB according to the TSINK buffer designation) in which:

1. certain high-order data bits have been grouped together and protected with (7,4) Hamming codewords;
2. the data to be transmitted has been "bitstreamed", one bit per half-word, in the form used by the TMODEM scroll program;
3. histogram information of the coded analysis parameters has been recorded.

These operations are performed in the CSPU, and therefore they can be executed concurrently with an AP operation. These operations are implemented as two separate CSPU modules so that each module's execution time is (mostly) hidden by that of an AP module. The PROPAR module performs the error-protection and bitstreaming of the analysis parameters (gain, delta-gains, pitch, pitch filter coefficients, and reflection coefficients), and it also records the histogram information for those parameters (except pitch). PRORES bitstreams the residual values only.

The format of the TSINK buffer is shown in Table 1, along with the number of bits per parameter and the number of high-order bits protected by Hamming codes. (C1-C3 denote the three pitch filter coefficients, and K1-K6 denote the six reflection coefficients.) A total of 44 high-order bits of the transmitted parameters are protected using 11 Hamming (7,4) codewords.

<u>Word</u>	<u>Parameter</u>	<u>No. of bits</u>	<u>Bits protected</u>
0-215	Residual(216 samples)	2	0
216	Gain	6	6
217-219	Delta-gain (3)	2	2
220	Pitch	7	7
221	C1	3	2
222	C2	4	3
223	C3	3	2
224	K1	6	5
225	K2	5	4
226	K3	4	3
227-229	K4,K5,K6	4	2

TABLE 1. TSINK BUFFER FORMAT

The TBITS buffer contains one data bit in the rightmost bit of each 16-bit half-word. The format of the TBITS buffer (also the frame of data transmitted) is shown in Table 2. (Bits of coded parameters are numbered starting with bit 0 on the right. B5432 thus denotes a 4-bit field of bits 5 through 2.)

A histogram-gathering function was included in the PROPAR module for gathering statistics on the effectiveness of the quantization tables for the analysis parameters and for verifying correct transmitter operation. A PDP-11 program (HIST16) was implemented (not delivered, but shown in Appendix D) to read the histogram buffers defined on Bus 1 and list them in a text file.

2.2.1.4 TMODEM Interrupt Service (TMODEMINT)

The TMODEM Interrupt Service routine is shown in Fig. 7. When activated by a Line 1 interrupt from the modem IOS-2 scroll, this routine updates the TMODEM pointer offset to point to the

<u>Bit No.</u>	<u>Function</u>
0	Sync bit
1-2	1st residual sample (LSB first)
.	.
.	.
431-432	216th residual sample
433-439	B5432 gain (Hamming (7,4) codeword)
440-446	B10 gain, B10 delta-gain-1 (7,4)
447-453	B10 delta-gain-2, B10 delta-gain-3 (7,4)
454-460	B6543 pitch (7,4)
461	B0 C1
462-468	B2 C1, B210 pitch (7,4)
469	B0 C2
470-476	B321 C2, B1 C1 (7,4)
477	B0 C3
478	B0 K1
479-485	B54 K1, B21 C3 (7,4)
486	B0 K2
487-493	B4321 K2 (7,4)
494	B0 K3
495-501	B3 K3, B321 K1 (7,4)
502-508	B32 K4, B21 K3 (7,4)
509-510	B10 K4
511-512	B10 K5
513-514	B10 K6
515-521	B32 K6, B32 K5 (7,4)

TABLE 2. TBITS BUFFER FORMAT

current (just emptied) TMODEM buffer, then checks the current TBITS buffer flag to see if there is new bitstream data ready to be transmitted. If the flag is set to full, the data is copied from the TBITS buffer to the current TMODEM buffer, the TBITS flag is set to empty, and the TBITS pointer offset is updated to point to the other TBITS buffer/flag. If, on the other hand, the current TBITS buffer flag indicates not-full, bitstream data corresponding to a frame of silence is copied (from buffer TBTC) to the current TMODEM buffer, the "fake frame" is counted by

incrementing TMFFC, and the routine exits without having changed the TBITS pointer offset.

In either case, only 521 words (bits) of data are copied to the TMODEM buffer. The data bit of the first word of each TMODEM buffer is not copied into, as it is the synchronization bit: a 0 in TMODEMA and a 1 in TMODEMB. Since data is transmitted alternately from TMODEMA and TMODEMB, the data frames at the receiver have the sync bit alternating between 0 and 1 in successive frames.

2.2.1.5 TMODEM and RMODEM I/O Scroll Program (RTPROG)

The digital data input and output is performed by an IOS-2SM Scroll processor that has been augmented by a modem interface. This interface also contains the dual-rate clock for timing both the modem data and the speech sample input and output.

The TMODEM and RMODEM Scroll program (RTPROG) is shown in Fig. 8. Although the TMODEM output and RMODEM input processes are logically distinct, they are performed by this single program. After setting the rates of the modem-data and speech-sample clocks and initializing buffer switches and address pointers, the program enters a basic loop that checks the two peripheral flags for data ready from the modem receiver interface (P1 set) or for the modem transmitter interface ready to accept data (P2 set).

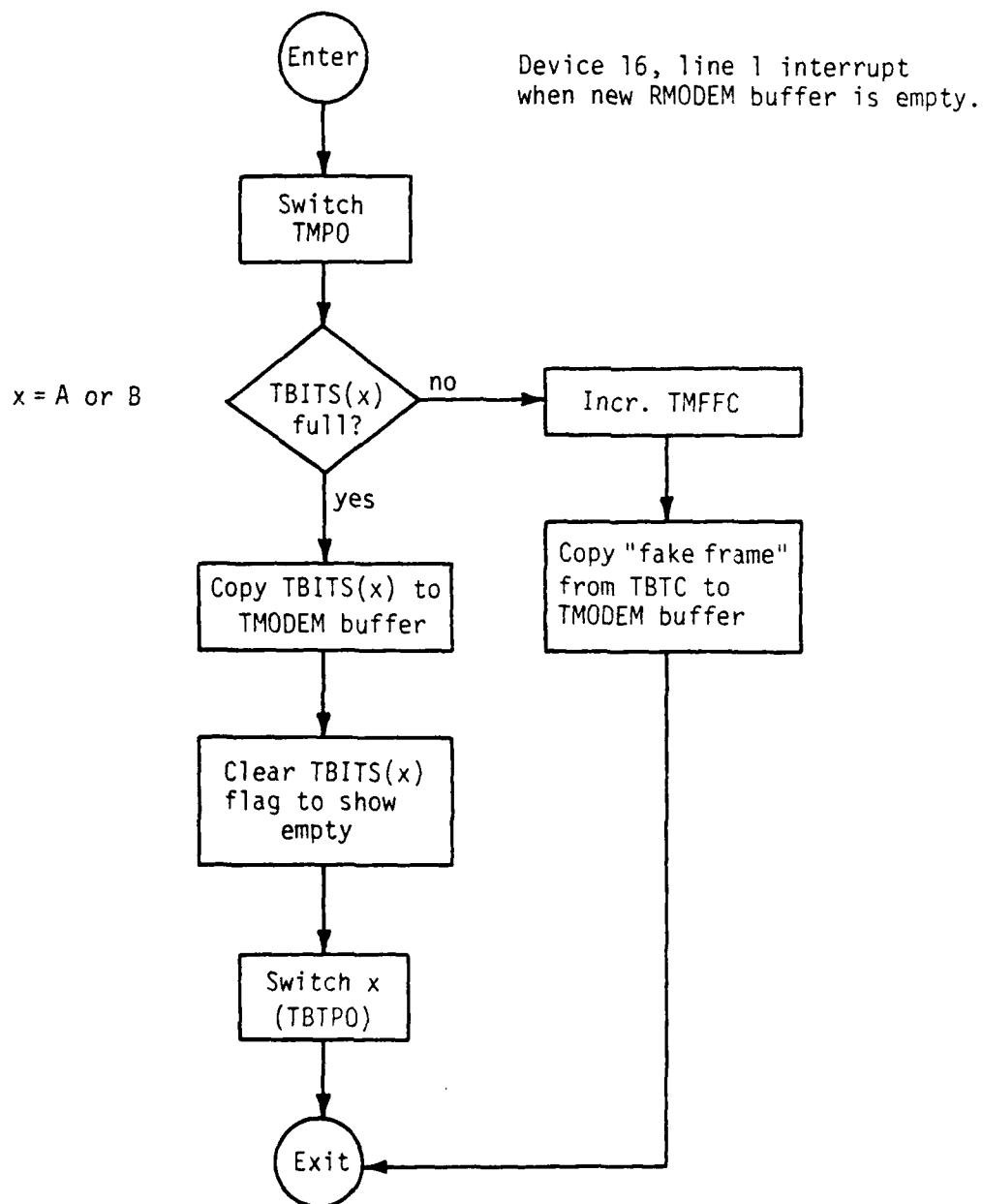


FIG. 7. TMODEM INTERRUPT SERVICE ROUTINE (TMODEMINT)

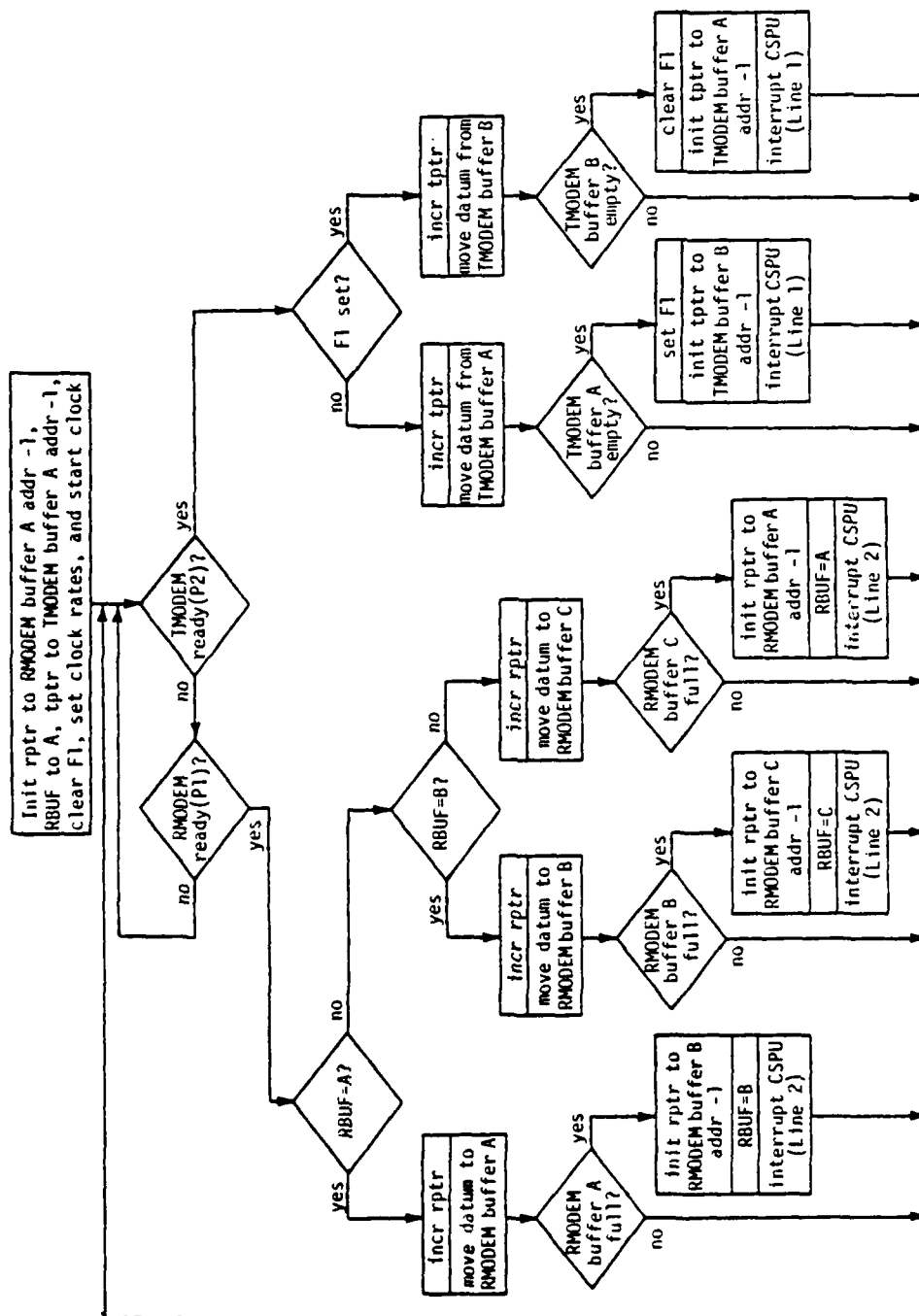


FIG. 8. TMODEM AND RMODEM SCROLL PROGRAM FLOW CHART

When a new datum is available from the interface, it is transferred to RMODEM buffer A, B, or C, depending on the value of a buffer switch in register R0 and an address pointer in register R1. The transfer clears flag P1. If the input buffer is now full, the buffer switch and address pointer are changed to point to the next buffer and the CSPU is interrupted on line 2.

When the interface becomes ready to accept a datum (to be transmitted), a word is transferred from TMODEM buffer A or B, depending on the value of flag F1, used as a buffer switch, and an address pointer in register R2. The transfer clears flag P2. If the output buffer is now empty, the buffer switch and address pointer are changed to indicate the other TMODEM buffer and the CSPU is interrupted on line 1.

2.2.1.6 Transmitter Input/Output Buffer Initial Sequence

The several layers of shared input/output buffers in the speech coder transmitter and receiver require that the modules that reference them be initialized to do so in the proper order. The start-up sequence for the transmitter is described below.

The ADAM scroll program starts by writing speech samples into buffer TADBA (and then proceeds to write into TADBB, etc.). At the first ADAM interrupt, ADAMINT copies from TADBA to TSRA (TSOURCE A buffer) and sets the flag TSRFA. The background process executes ANLZA first, since TSRFA=1 and TBTFB is initially 0. The execution of ANLZA includes PROPAR(B) and

PRORES(B), which write the first bitstream data into buffer TBTB; then the flag TBTFB is set to 1. The TMODEMINT interrupt routine is initialized to expect its first bitstream data in buffer TBTB, so that is the first data copied into a TMODEM buffer for output by the TMODEM scroll program.

Meanwhile, the TMODEM scroll program has started by transmitting (initialized) data from buffer TMDMA. It then sends an interrupt and proceeds to read from TMDMB, and so forth. The initial TMODEM interrupt activates TMODEMINT, whose first task is to provide new data for the just-emptied TMDMA buffer. TMODEMINT is initialized to check the flag TBTFB, as described above. At the end of the first frame, however, ANLZA has not yet executed and there is as yet no data in TBTB, so TMODEMINT copies "dummy" bitstream data instead from TBTC into TMDMA and exits. The next time TMODEMINT is activated, it again checks TBTFB and (since ANLZA has run by now) finds it set, so it copies the new bitstream data from TBTB to the (just-emptied) TMDMB buffer. Thereafter all data buffers are alternately filled and emptied without conflict.

2.2.2 Receiver

The Receiver is similar in structure to the Transmitter. The Receiver is shown in Fig. 9.

Data is accepted from the modem and put into one of three

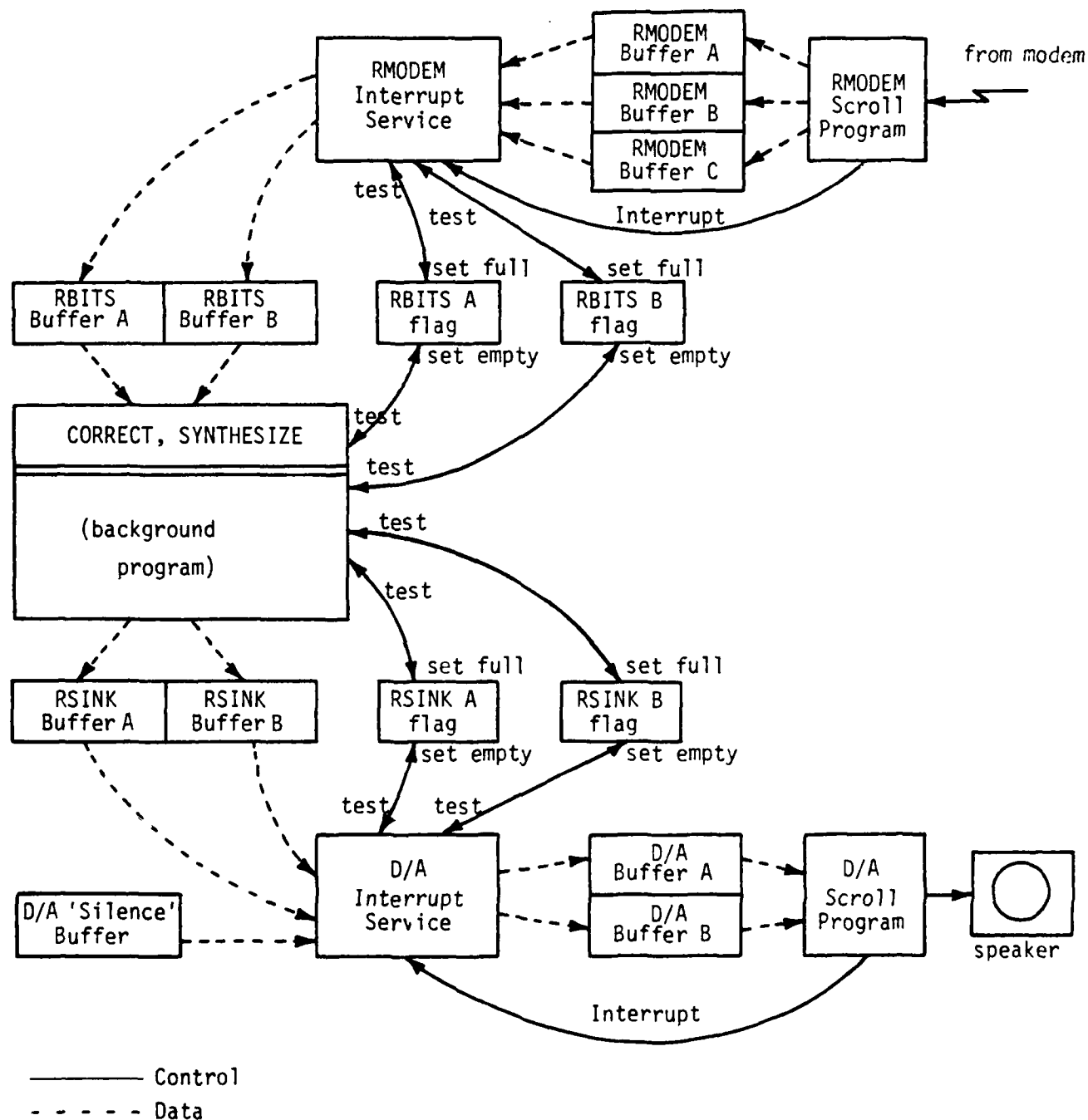


FIG. 9. RECEIVER PROCESS

RMODEM buffers by the RMODEM program running in an I/O scroll. When this program fills a buffer, it generates an interrupt to the CSPU and begins filling the next buffer (A,B,C,... in rotation).

This interrupt activates the RMODEM Interrupt Service routine, which checks frame synchronization and, if correct, transfers the new data to an empty RBITS buffer and sets the corresponding RBITS flag to indicate full. The data that is actually transferred does not correspond exactly to the just-filled RMODEM buffer. Rather, it is a "frame" of data completely contained in the new RMODEM buffer and the RMODEM buffer previously filled. A frame of data begins with a sync bit. Since this bit may occur anywhere within a single RMODEM buffer, two full buffers (and therefore three buffers altogether) are required to guarantee that a complete frame is available.

The CORPAR and DECODA/B modules, running in the CSPU at background level, empty the full RBITS buffer, perform error correction and decoding of the incoming bitstream, copy these decoded parameters into an (empty) RSOURCE buffer, and clear the RBITS flag to signify empty.

The SYNTHESIZE module, running in the CSPU at background level, empties the (full) RSOURCE buffer, performs the processing necessary to synthesize speech, and puts the output speech samples into an (empty) RSINK buffer, setting the corresponding

RSINK flag to indicate full. The SYNTHESIZE module, although logically following the CORPAR and DECODA/B modules, in fact contains them and is executed concurrently with them. That is, SYNTHESIZE operates on the data supplied by the previous execution of CORPAR and DECODA/B. The first time that SYNTHESIZE is executed, it operates on a RSOURCE buffer that has not been filled by CORPAR and DECODA/B; therefore the RSOURCE buffers are initialized to contain data that will synthesize silence.

The data from the RSINK buffer is transferred to an empty D/A buffer by the D/A Interrupt Service routine. This routine also sets the corresponding RSINK flag to indicate empty.

This Interrupt Service routine is activated by an interrupt from the D/A Scroll program, running in another I/O scroll (the AOM, or Analog Output Module). This scroll program transfers data from a D/A buffer to the D/A converter, interrupts the CSPU when the buffer is empty, and begins transferring data from the other D/A buffer.

2.2.2.1 RMODEM Scroll Program

The RMODEM scroll program is described in Section 2.2.1.5.

2.2.2.2 RMODEM Interrupt Service (RMODEMINT)

The RMODEM Interrupt Service routine is shown in Fig. 10. It is activated by a line 2 interrupt from the modem IOS-2 scroll, which indicates that the next RMODEM buffer is full. The

routine updates the RMODEM pointer offset to point to the just-filled buffer. Then it copies the on-hook bit from the first word in the current RMODEM buffer to the integer scalar RONHK, where it is saved for use by the ADAM interrupt service routine (Section 2.2.1.2).

Then the integer scalar RERRS is tested. This is normally zero, but if it is set to nonzero (by a command from the host program), a channel error simulation routine (RMI\$SIM) is called. RMI\$SIM uses the value in RERRS as the number of errors to be introduced (quasi-randomly) per frame. Since there are 522 bits per frame, each error contributes about 0.2% to the error rate. This error simulation is intended only for demonstration; it is not an accurate simulation of an errorful channel.

The next set of operations deals with frame synchronization. Since the receiver has no prior knowledge about the position of the frame boundaries in the input data stream, it must infer the boundary position from the received data pattern. The first bit of each transmitted frame is the sync bit, which alternates between 0 and 1 in successive frames; this pattern allows the receiver to detect and maintain frame synchronization. The synchronization routines are described in Section 2.2.2.2.1 below.

RMODEMINT tests the value of RSYNC, an integer scalar that maintains the receiver's sync-state. If RSYNC=0 (as it is when

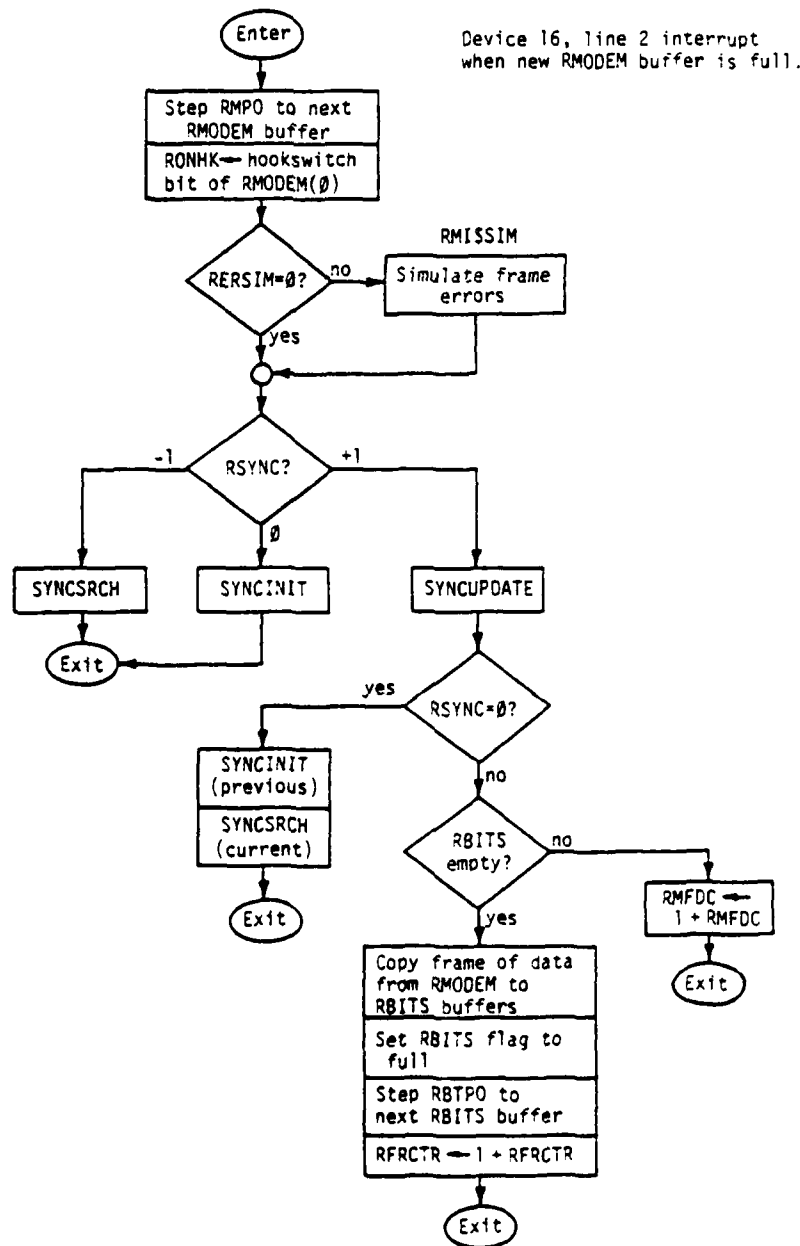


FIG. 10. RMODEM INTERRUPT SERVICE MODULE (RMODEMINT)

the speech coder is initialized), nothing is known about sync, so the SYNCINIT routine is called to initialize the sync-search process with the current RMODEM buffer. If $RSYNC < 0$, then the receiver is still searching for sync, so the SYNCSEARCH routine is called to continue the search using the new data in the current RMODEM buffer. In both of these cases, RMODEMINT exits immediately. If $RSYNC > 0$, then the receiver has gained frame-sync, so the SYNCUPDATE routine is called to check that the current RMODEM buffer continues to show the expected sync-bit value. Upon the return from SYNCUPDATE, RSYNC is checked again; if it has been set to zero, then the receiver has lost sync and must again search the incoming bitstream for the sync-bit pattern; therefore SYNCINIT and SYNCSEARCH are called to restart sync-searching with the previous and current RMODEM buffers.

If RSYNC is still greater than zero after SYNCUPDATE, then synchronization is confirmed, and the receiver makes use of the data for speech output. RMODEMINT checks the current RBITS buffer flag; if empty, the most recent frame of data is copied from the RMODEM buffers to the current RBITS flag, the RBITS flag is set to full, and the RBITS pointer offset (RBTP0) is switched to the next RBITS buffer. Note that in general, the current frame will straddle the previous and current RMODEM buffers. It is for this reason that there are three RMODEM buffers, two to hold the current frame while the third is being filled by the RMODEM scroll program.

If, for some reason, the RBITS buffer is not empty (ready to accept new data), the new data frame is effectively discarded. A frame discard counter (RMFDC) is incremented to take note of this, and the routine exits without updating RBTPO.

2.2.2.2.1 Synchronization Routines

The SYNCINIT subroutine (Fig. 11) initializes the state of the frame synchronization section of the speech coder.

Its primary function is to copy the data bit of each word of the current RMODEM buffer into RSSPF, the "previous frame" memory used in SYNCsrch, and to clear the counts in the RSSSS buffer. SYNCINIT also zeros the gain word of both RSOURCE buffers and the entire previous RBITS buffer, so that when the speech coder receiver resumes operation after regaining sync, the incorrect information in these buffers will not produce unwanted transients in the speech output.

The SYNCsrch subroutine (Fig. 12) scans the incoming RMODEM buffers (one per call), building up statistics on the frame-to-frame data patterns in each bit position of the frame until it detects a large enough number (ACQTHR) of consecutive bit alternations in a single bit position to allow it to identify that position as carrying the sync bit.

When that occurs, it sets the value of RBOFO ("beginning-of-frame-offset") to the distance from the start of the buffer to

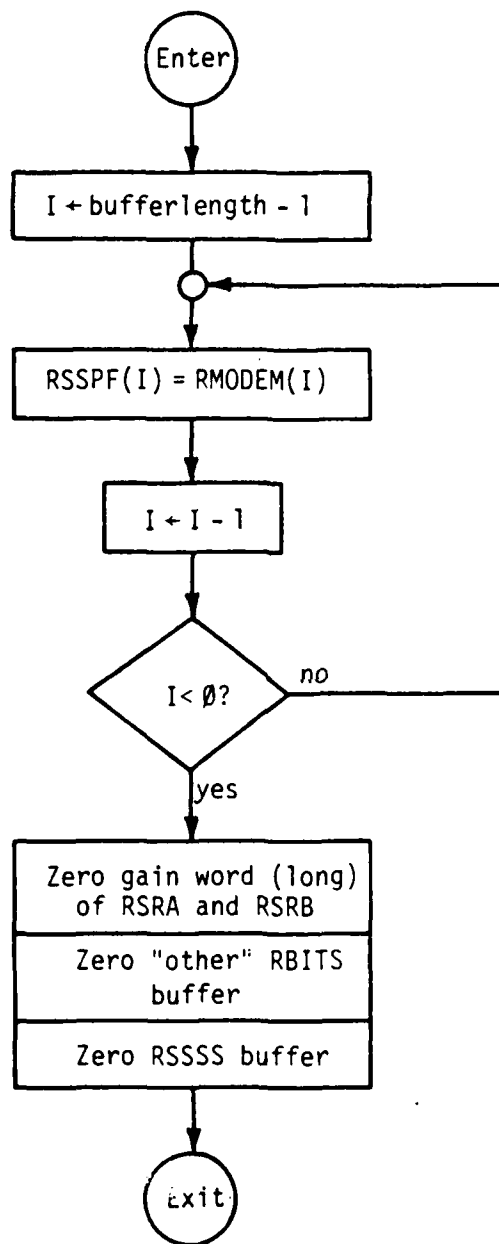


FIG. 11. FRAME SYNCHRONIZATION INITIALIZATION MODULE (SYNCINIT)

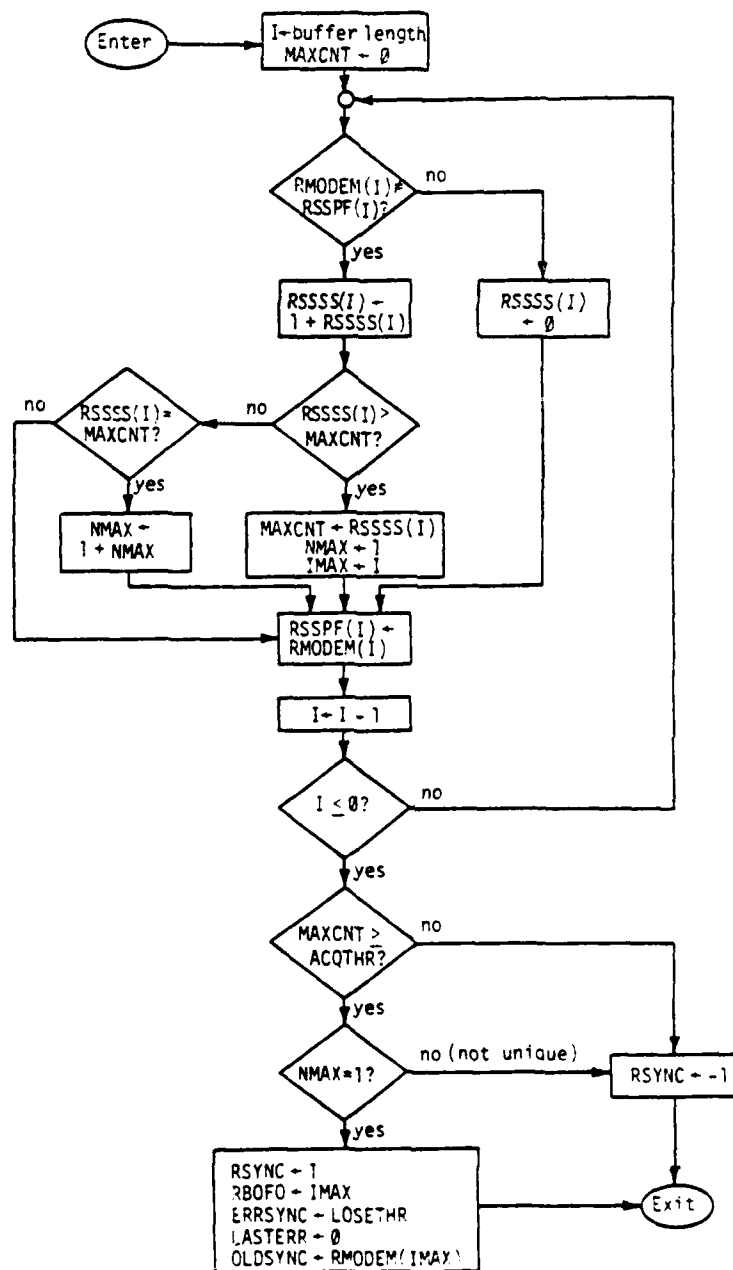


FIG. 12. FRAME SYNCHRONIZATION SEARCH MODULE (SYNCSRCH)

the sync-bit position and RSYNC to +1 to cause RMODEMINT to start passing input frames to the speech coder synthesizer on the next RMODEM interrupt. SYNCSEARCH also initializes three variables that are used in subsequent calls to SYNCUPDATE.

The SYNCUPDATE subroutine (Fig. 13) is used once sync has been acquired, to check the sync-bit position of each incoming RMODEM buffer to see if it has the expected value.

It is the function of SYNCUPDATE to declare sync lost if the declared sync bit does not show the expected frame-to-frame alternation. Since channel errors may cause the sync bit to be incorrect, the loss-of-sync criterion must be more stringent than a single incorrect sync bit. The criterion is a sufficient number (LOSETHR) of incorrect sync bits without two consecutive correct sync bits. A simpler criterion, such as LOSETHR consecutive sync bit errors, would be unsatisfactory, since the high-order bit of many speech coder parameters (e.g., energy) may have the same value for many consecutive frames, and a constant bit would compare "correctly" with an alternating bit sequence every other frame.

The performance of the synchronization routines is controlled by two thresholds, ACQTHR and LOSETHR. It is necessary to set ACQTHR high enough so that it is very unlikely that SYNCSEARCH will detect sync on the wrong bit position. The value of LOSETHR must be high enough that SYNCUPDATE is unlikely

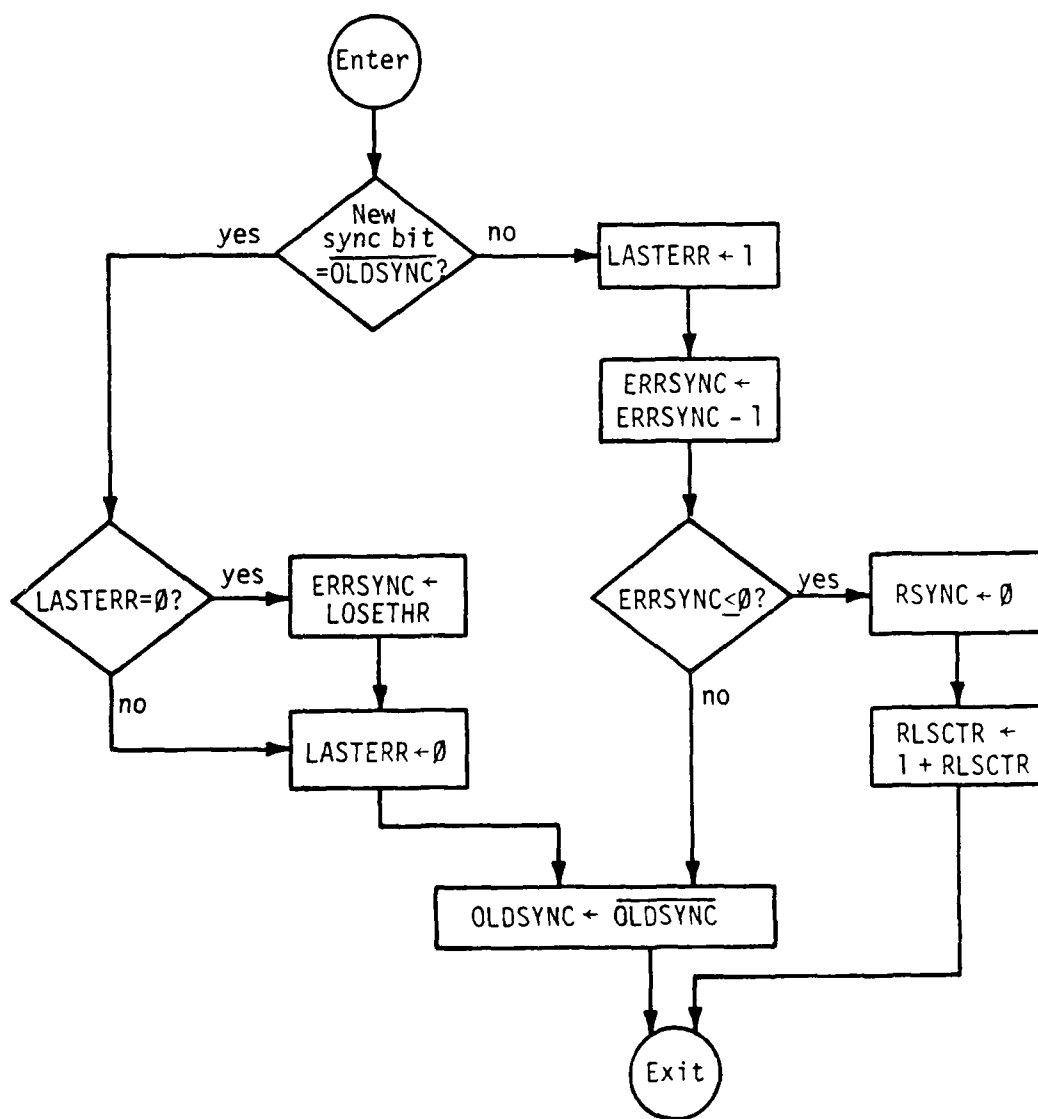


FIG. 13. FRAME SYNCHRONIZATION UPDATE MODULE (SYNCUPDATE)

to declare loss-of-sync due to channel errors falling on the sync bit, but small enough that if sync should be lost for some reason, the receiver does not synthesize too much "garbage" before detecting the loss-of-sync. Of course, while the receiver is searching for sync, RMODEMINT does not pass data to the synthesizer, so silence is output. For a 1% channel bit error rate, the values of ACQTHR=10 and LOSETHR=4 give a probability of acquiring false sync of about 0.0002 and an expected time to spontaneous loss-of-sync of about 10 hours, respectively.

2.2.2.3 SYNTHESIZE Module

The SYNTHESIZE Module implements the APC synthesis algorithm described in Section 3 of the Algorithm Specification. As described in Section 2.1.3, two versions of the SYNTHESIZE Module exist, one for each of the two sets of input/output buffers required for double buffering. Each version consists of a function list of MAP-300 function calls. The two versions (SYNZA and SYNZB) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions.

As in the case of Section 2.2.1.3, this section describes, on a function by function basis, the SYNZA/SYNZB function lists, which also appear in subroutine BBN16F.

2.2.2.3.1 Unbitstreaming and Error Correction

The CORPAR and DECODA/B modules take as input an RBITS

buffer containing a frame of bitstream data input from the modem and produce as output an RSOURCE buffer containing error-corrected and decoded floating point values of analysis/synthesis parameters and residual samples, ready for immediate use by the speech coder synthesizer.

The format of the RBITS buffer is the same as the TBITS buffer, shown in Table 2 above. The data bits are regrouped, and the 7-bit codewords are error-corrected by lookup in an inverse Hamming code table. The bits are then grouped into coded parameter values and decoded by lookup in parameter-specific decoding tables of floating-point values. (As noted in Section 2.2.1.3.13, the gain and delta-gain decoding tables differ from the ones in the Algorithm Specification by a scale factor.) The format of the RSOURCE buffer is similar to that of the TSINK buffer (Table 1), except that the half-word coded TSINK values are replaced by full-word (32-bit) floating point values.

As in the case of the error-protection and bitstreaming operation, the unbitstreaming, error-correction, and decoding operation is performed by two separate modules, which are part of the SYNTHESIZE function list. CORPAR does the operations on the analysis parameters, while DECODA/B performs unbitstreaming and decoding on the residual samples. Unlike the other "bit-pushing" modules, DECODA/B is an AP module. (There are two distinct DECOD modules. DECODA decodes from buffer RBTA to buffer RSRA; DECODB

decodes using the other pair of buffers. The SNAP-II Executive permits the definition of only 63 distinct data buffers; this proved to be too few to allow the definition (to SNAP-II) of the RBITS buffers. Therefore, two DECOD modules for the APS were created, each with the parameters of its data buffers "hard bound" in the source code.)

2.2.2.3.2 Begin Function List

MPBFL(SYNZA/B)

This CSPI function specifies the start of the indicated function list.

2.2.2.3.3 Clear G-flag

MPGSC(IG3,ICLR)

This BBN function causes general purpose flag G3 to be cleared, indicating the start of SYNZ processing. This use of flag G3 is intended for system debugging, timing, and internal measurement and is not required for proper operation of the speech coding software.

2.2.2.3.4 Receiver Data Distribution

DEAL(RSRA/B,RGH,RDGH,RMH,RCH,RKH)

This BBN function distributes all decoded parameter data from a single buffer (RSRA/RSRB) to separate buffers and scalars. (The decoded residual data will be read directly from RSRA/RSRB by the SCLRES function.)

2.2.2.3.5 Receiver Frame History Buffer Update

VMOV(RRH,RRH0)

This CSPI function performs a frame history update on a single buffer. The last-frame data from the end of the buffer (RRH0) is moved to the beginning of the buffer (RRH) for use during current frame processing.

2.2.2.3.6 Residual Scaling

SCLRES(RWH,RGH,RDGH,RUHA/B)

This BBN function computes quantizer scale factors and scales the received APC residual (RUHA/RUHB are equivalent to the beginning of the RSRA/RSRB) into buffer RWH, as indicated in Sections 3.2 and 3.3 of the Algorithm Specification.

2.2.2.3.7 Spectral Synthesis

VLTSY(RVH,RVHM,RKH,RWH)

This BBN function performs a lattice-form spectral synthesis filtering operation on the scaled APC residual (RWH) as indicated in Section 3.5 of the Algorithm Specification, using reflection coefficients (RKH) directly. The output is in RVH. (This function implements an 8-pole filter; we are using it with the last two coefficients zeroed to implement a 6-pole filter.)

2.2.2.3.8 Decoding of Previous Frame Receiver Residual Data and Error Correction on Previous Frame Receiver Parameter Data

DECODEB/A
CORPAR(B/A)
MPWT(PRCR,AP)
MPIST(RBTFB/A,0)

These functions (DECOB/DECODA and CORPAR are BBN-written; MPWT and MPIST are CSPI-supplied) decode the previous frame received residual samples and error-correct the previous frame received parameter data (from RBTB/RBTA), depositing decoded data in buffer RSRB/RSRA. The appropriate buffer status flag (RBTFB or RBTFA) is cleared, indicating that the associated buffer can now be considered empty (i.e., ready to be filled with new received coded data). The last three function execute in the CSPU while the first executes in the AP. (See Section 2.2.2.3.1 for a more detailed discussion of DECODA/DECOB and CORPAR.)

2.2.2.3.9 Pitch Synthesis

PITSYN(RRH,RMH,RVH,RCH)

This BBN function performs a pitch synthesis filtering operation as indicated in Section 3.6 of the Algorithm Specification. The input is in RVH, and the output is in RRH.

2.2.2.3.10 Deemphasis

DFL22(RRHDA/B,RDCF0,RRH0,RDMY0)

This CSPI function does signal deemphasis as indicated in Section 3.7 of the Algorithm Specification. The filter coefficients are in four consecutive scalars starting with RDCF0. The filter memory is in four consecutive scalars starting with RDMY0. The synthesized speech is stored in RRHDA/RRHDB (equivalent to RSNKA/RSNKB).

2.2.2.3.11 Output Ready

MPWT(PCRSR,AP)
MPIST(RSNFA/B,1)

These CSPI functions set the appropriate buffer status flag (RSNFA or RSNFB) once the previous function has

completed, indicating that the associated buffer is full (i.e., ready to be emptied).

2.2.2.3.12 End of Function List

MPEFL(SYNZA/B)

This CSPI function specifies the end of the indicated function list.

2.2.2.4 D/A Interrupt Service Routine (AOMINT)

AOMINT is activated by each AOM line 1 interrupt, signifying the emptying of a D/A output buffer by the AOM scroll program. Its operation, as illustrated by Fig. 14, is analogous to that of the TMODEMINT routine, the only difference being the copying of synthesis output data from RSINK buffers to D/A buffers for output to the AOM. If an RSINK buffer is not available (full), a buffer of silence is output in its place. This happens whenever the speech coder is not receiving data from the remote speech coder and during frame synchronization or resynchronization.

2.2.2.5 D/A Scroll Program (DAPROG)

Speech output is performed by the AOM, an IOS-2 scroll processor that contains two D/A converters. The AOM gets its sample output clock from the SPI, and it sends the D/A output signal to the SPI for subsequent low-pass filtering and output. The signal is in the range -5 to +5 volts, and the sample rate is 6.621 kHz, the same as for the A/D input.

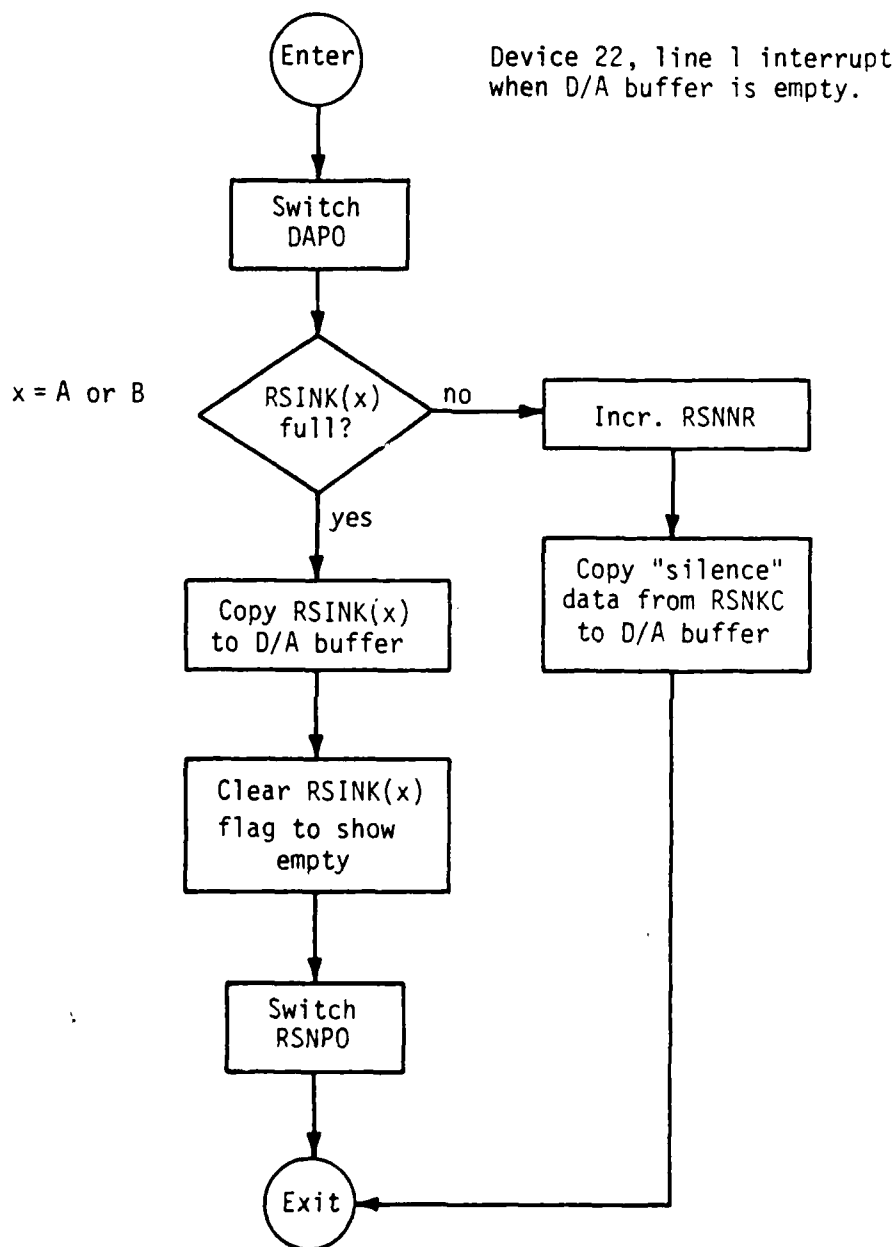


FIG. 14. D/A INTERRUPT SERVICE ROUTINE (AOMINT)

The AOM program (DAPROG) is illustrated in Fig. 15. In "single-channel" mode, the AOM outputs an internally-generated value on D/A Channel 0 while it converts speech samples from MAP memory on D/A Channel 1. The Channel 0 signal is a ramp that is reset each frame time; it is irrelevant to speech coder operation, but it may be used for synchronization or horizontal sweep on an external oscilloscope.

The operation of DAPROG is analogous to that of ADPROG. Output begins from buffer RDABA, and when the end of that buffer is reached, a line 1 interrupt is sent to the CSPU, and buffer RDABB is selected. Reaching the end of RDABB produces another line 1 interrupt and a switch to RDABA, and so forth.

2.2.2.6 Receiver Input/Output Buffer Initial Sequence

The buffer start-up sequence for the speech coder receiver is described below.

The RMODEM scroll program starts by writing received data words into buffer RMDMA, sending an interrupt when it is filled. Upon the initial interrupt, RMODEMINT begins by using RMDMA for frame-sync initialization. Since frame-sync has not yet been acquired, RMODEMINT does not output any data. Successive RMODEM interrupts cause RMODEMINT to use buffers RMDMB, RMDMC, RMDMA, etc. for further frame-sync searching until, after at least 12 input frames, sync has been found. Then RMODEMINT will copy a frame of bitstream data from the RMODEM buffers to buffer RBTA

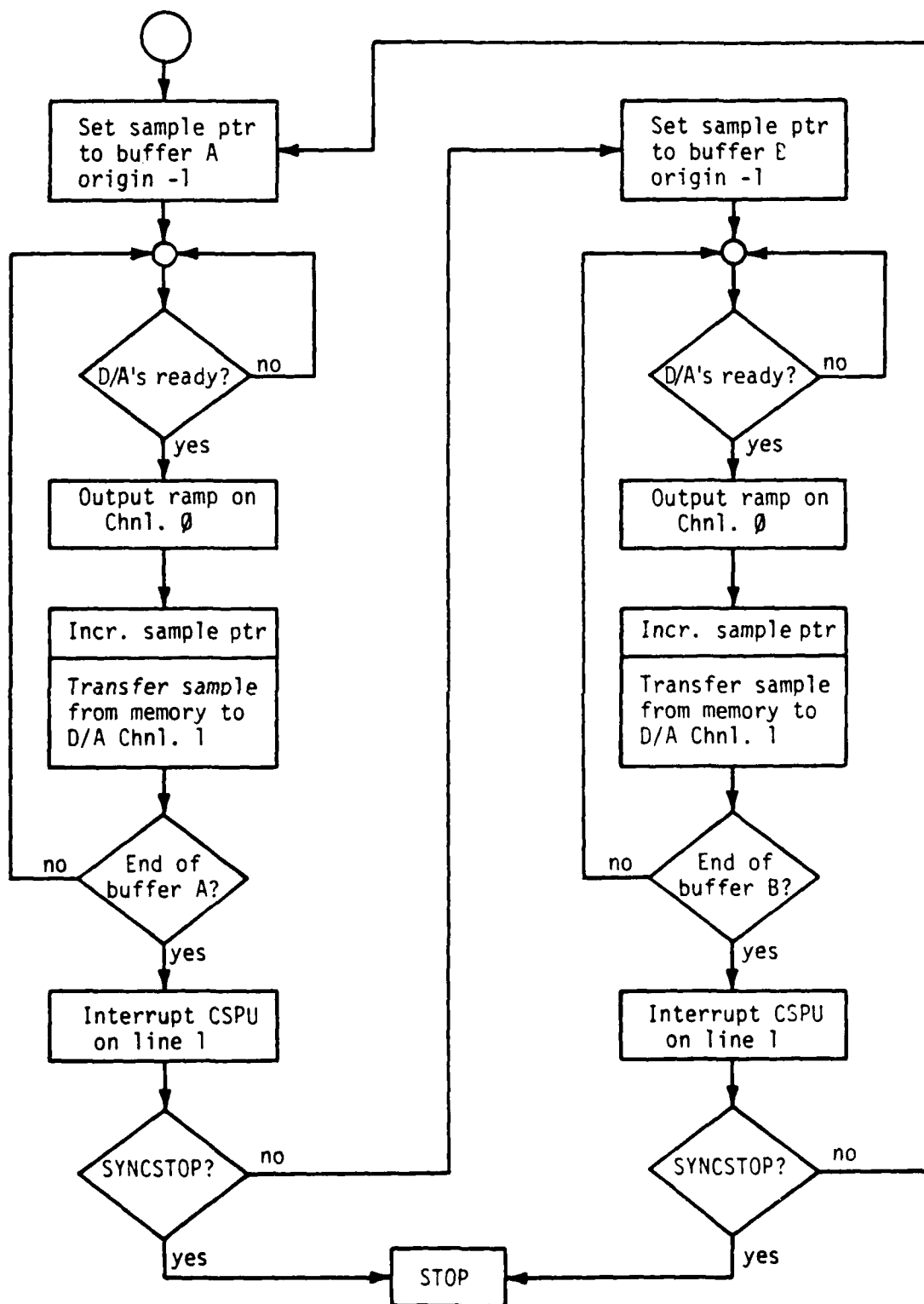


FIG. 15. DAPROG: AOM SCROLL PROGRAM

and set the flag RBTFA=1. The background process will execute SYNZB first, since RBTFA=1 and RSNFB is initialized to 0. SYNZB writes the first synthesized speech data in buffer RSNKB and sets the flag RSNFB=1. The AOMINT interrupt routine is initialized to expect the first synthetic speech data in buffer RSNKB, so when it is next activated, that buffer is copied into a D/A buffer for output by the AOM scroll program.

Meanwhile, the AOM scroll program has started by reading data (initialized to silence) from buffer RDABA, sending an interrupt, going on to RDABB, etc. These AOM interrupts activate AOMINT, whose task is to copy new synthesized speech data from RSINK buffers (initially RSNKB) to the just-emptied D/A buffers (initially RDABA). On its first several (usually 13) activations, AOMINT finds RSNKB empty, because the receiver has not yet gained frame-sync, so AOMINT copies a buffer of silence from buffer RSNKC to the D/A buffers instead. On the next activation after SYNZB has run, AOMINT finds RSNKB full, so it copies it to the D/A buffer, and thereafter all data buffers are alternately filled and emptied without conflict.

2.3 SYSTEM HARDWARE

The 16 kb/s speech coder system is implemented on a MAP-300 array processing computer, which is manufactured by CSP Inc., of Billerica, Mass. Section 2.3.1 describes the configuration of

MAP-300 equipment that is necessary for the speech coder system. Section 2.3.2 describes the additional audio signal interface and IOS-2SM scroll modifications for connection to a modem that complete the system.

2.3.1 MAP-300 Hardware

The MAP-300 configuration that was specified by Defense Communications Agency for the implementation of the speech coder system is listed below.

- 1 1030 MAP-300 Processor
- 1 2030 8K x 32 MOS Master Memory, 500 nsec, Bus 1
- 1 2050 16K x 32 MOS Slave Memory, 500 nsec, Bus 1
- 1 2203 8K x 32 MOS Master Memory, 300 nsec, Bus 2
- 1 2410 4K x 32 MOS Master Memory, 170 nsec, Bus 3
- 1 3110 PDP-11 Interface
- 1 4020 Model 2SM I/O Scroll
- 2 4040 Bus Switch (for Model 2SM I/O Scroll)
- 1 5120 Analog Data Acquisition Module
- 1 5130 Analog Output Module
- 1 6100 Expansion Chassis
- 1 6200 Auxiliary Power Supply

2.3.2 Audio and Modem Interface Hardware

In addition to the MAP-300 equipment listed above, two other pieces of equipment enable the MAP-300 to function as a complete, stand-alone speech coder system: an audio signal interface and a digital data interface to a modem. These two items were designed and built by the GTE Sylvania Electronic Systems Group in Needham, Mass. (see [1], Appendix A). Identical interfaces were provided for all three DCA MAP-300 systems so that they would be interchangeable at the hardware level.

The audio signal interface consists of a handset, tape input and output jacks, and circuitry for the amplification, equalization, and filtering necessary for interfacing speech input and output signals to the MAP-300 A/D and D/A converters.

The modem interface consists of modifications to the MAP-300 IOS-2SM scroll processor for the purpose of transferring data from MAP memory to the modem and also data from the modem into MAP memory. Two real-time clocks derived from a single master oscillator are also provided for controlling the modem data rate and the speech sampling rates.

2.4 SYSTEM SOFTWARE

The speech coder software consists of two distinct sets of modules. The first set is made up of modules that run in the MAP-300, including CSPI-supplied as well as BBN-written programs. Section 2.4.1 describes the MAP-300 modules that must be loaded into the MAP-300 processor before the speech coder can be operated.

The second set of software modules consists of FORTRAN routines that run in the PDP-11. These routines make use of the CSPI-supplied SNAP-II software system to initialize and start the MAP-300 speech coder programs. Section 2.4.2 describes these routines.

2.4.1 MAP-300 Software Components

All MAP-300 processing is done in conjunction with Release 3.5 of the CSPI-supplied SNAP-II software system. For the most part, BBN-written MAP-300 programs take the form of new AP or CSPU functions, callable via the standard SNAP-II calling procedure. In addition, BBN-written interrupt service routines and input/output scroll programs have been added to the executive.

CSPI-supplied MAP-300 software necessary for speech coder operation is described in Section 2.4.1.1. MAP-300 speech coder software written by BBN is described in Section 2.4.1.2.

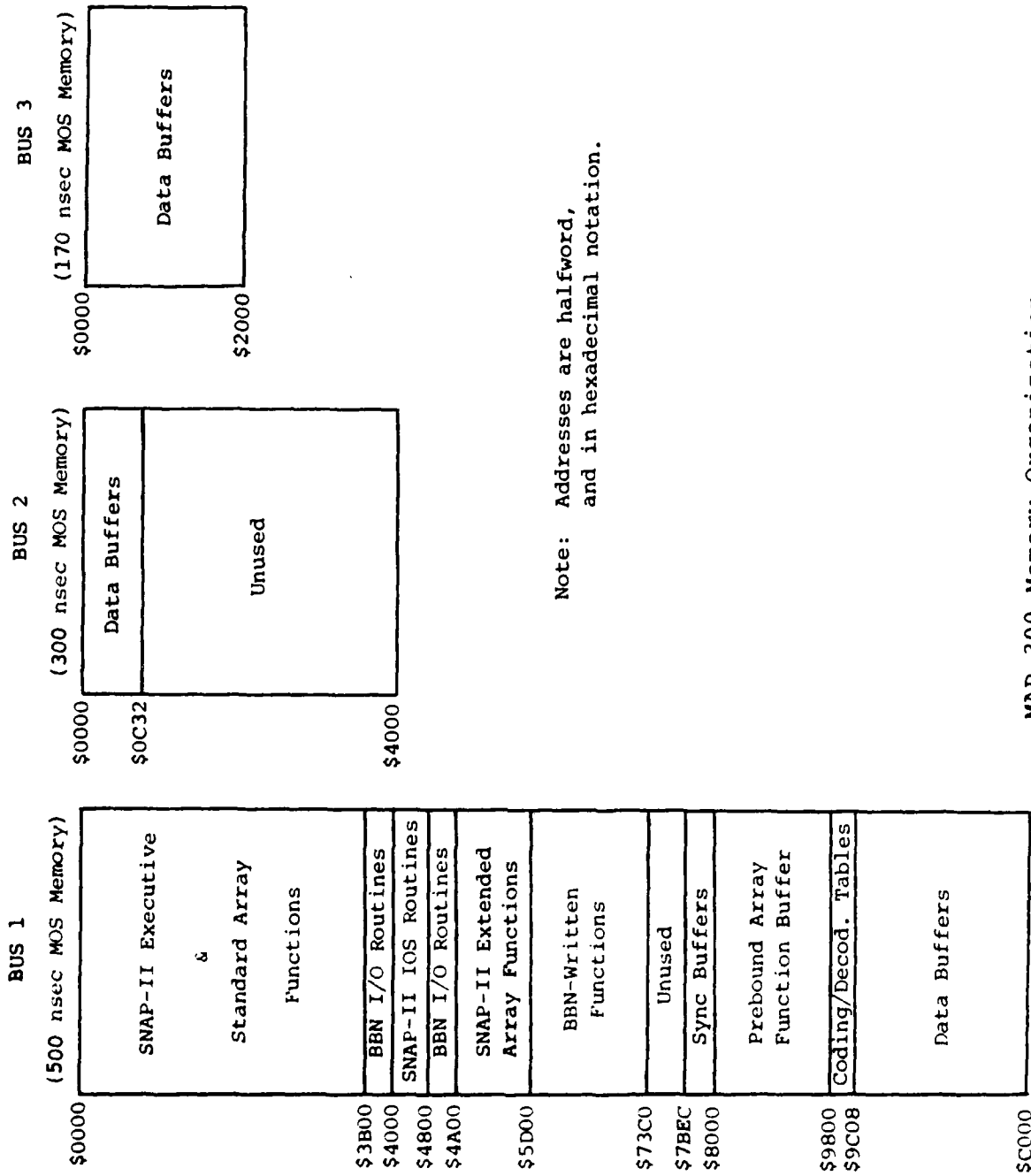
Fig. 16 shows the MAP-300 memory organization, including the location of CSPI and BBN software, as well as the location of buffers defined in the speech coder system.

2.4.1.1 CSPI-Supplied MAP-300 Software

The following CSPI-supplied MAP-300 software modules are required for speech coder operation:

SNAP-II Software System Release 3.5 Model 8300-RSX11M.
(This package includes the SNAP-II Executive and the
Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package Release 0.1
Model 8400-RSX11M.
(This package includes the SNAP-II IOS Modules.)



MAP-300 Memory Organization
Fig. 16

These software modules are described in CSPI documentation [2,3,5].

2.4.1.2 BBN-written MAP-300 Software

BBN-written additions and modifications to the SNAP-II software system are contained in four separate files. (For all BBN-written MAP-300 modules, a file extension of .MSO designates "MAP source", .MOB designates "MAP object", and .MLI designates "MAP listing".) The files with first name "BBN16M" contain added SNAP-II functions (array and non-array) for algorithmic and system support processing. "BBN16T" contains tables for quantization and decoding. "BBN16U" contains the error protection and correction modules, ADAM, AOM, and IOS-2 programs, and CSPU routines that respond to interrupts from these devices. These programs and routines are described in Section 2.2 of this report. "BBN16P" contains patches to Release 3.5 of the SNAP-II executive. Assembly-listings of these four files appear in Appendix D of this report.

BBN-written SNAP-II functions are functionally described in Appendix A of this report. Each function has been assigned a Function Control Block (FCB) number. An entry has been made for each new FCB in the Function Dispatch Table (FDT), indicating the location of the APU, APS and/or CSPU module(s) that implement the associated function.

Several unused CSPI-supplied functions, normally available

with Release 3.5 of the SNAP-II software system, have been disabled to provide room in the FDT for BBN-written functions. The following SNAP-II functions are unavailable once the BBN-written MAP-300 software has been loaded.

CXMUL	(FCB# 182)	CMML	(FCB# 220)
CXMUL	(FCB# 183)	CMINV	(FCB# 224)
CSMAI	(FCB# 188)	MWLD	(FCB# 225)
CSMA2	(FCB# 189)	ADMRB	(FCB# 229)
FF2D	(FCB# 212)	VHIST	(FCB# 230)
		VRAN1	(FCB# 235)

These functions can be made available by reloading the SNAP-II software system without loading BBN-written MAP-300 software.

In addition, the standard SNAP-II interrupt routines for interrupts from Device 16 (Lines 1 and 2), Device 22 (Line 1), and Device 23 (Line 1) have been modified to transfer control to BBN-written interrupt routines. The standard interrupt routines for these devices can be similarly accessed by reloading the SNAP-II Executive without loading BBN-written MAP-300 software.

2.4.2 PDP-11 Software Components

The proper operation of the speech coder is defined and controlled by a group of programs running in the PDP-11. These programs include a set of SNAP-II support subroutines and a MAP-300 driver, supplied by CSPI, as well as a number of FORTRAN programs written by BBN that make use of the CSPI-supplied routines to perform the specific task of defining and controlling the operation of the speech coder in the MAP.

CSPI-supplied PDP-11 programs are described in Section 2.4.2.1. PDP-11 programs written by BBN are described in Section 2.4.2.2.

2.4.2.1 CSPI-Supplied PDP-11 Software

The following CSPI-supplied PDP-11 software modules are required for speech coder operation:

SNAP-II Software System Release 3.5 Model 8300-RSX11M.
(This package includes the SNAP-II Host Support Packages for Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package Release 0.1 Model 8400-RSX11M
(This package includes the SNAP-II IOS Host Support Package.)

DEC RSX-11M I/O Driver Model 8901

These software modules are described in CSPI documentation [2,3,5,6,7].

2.4.2.2 BBN-written PDP-11 Software

A set of BBN-written FORTRAN programs performs the multiple tasks of defining and initializing MAP-300 buffers and scalars and of defining and executing the sequence of functions in the MAP that perform the actual speech coding operations. These FORTRAN programs are organized around a single mainline program (BBN16R), which calls, in turn, subroutines to configure MAP buffers and scalars (BBN16C), initialize these buffers and scalars (BBN16I), define function lists for various speech coder

tasks (BBN16F), and interact with the user in controlling the execution of these function lists (DCA96E). These subroutines are described in Sections 2.4.2.2.1 through 2.4.2.2.4 below.

2.4.2.2.1 Buffer and Scalar Configuration (BBN16C)

The task of defining MAP-300 buffers and scalars to the SNAP-II executive is performed by subroutine BBN16C. Upon entry, this routine first initializes the SNAP-II executive via the MPOPN function. All SNAP-II buffers are then configured using the MPCLB function, with buffer ID numbers, sizes, and addresses defined symbolically within the BBN16C routine. In general, transmitter buffer names begin with "T", while receiver buffer names begin with "R". Buffer sizes are stored in variables named by prefixing an "S" to the the buffer name. Similarly, buffer addresses are stored in variables named by prefixing "A" to the buffer name. The buffers used in the speech coder system are listed in Appendix B. All real scalars and integer scalars are then defined by assigning scalar ID numbers to the associated symbolic scalar names. Transmitter scalar names generally begin with "T", while receiver scalar names begin with "R". Real and integer scalars used in the speech coder system are listed in Appendix C.

All buffer and scalar names (and certain buffer sizes and addresses) are included in FORTRAN labeled COMMON blocks, which permit all other FORTRAN subroutines in the speech coder system

to reference these symbolic buffer and scalar names in their calls to SNAP-II functions.

2.4.2.2.2 Buffer and Scalar Initialization (BBN16I)

MAP-300 buffers and scalars are set to their initial values by subroutine BBN16I. Only certain buffers and scalars require such initialization. Appendices B and C indicate the proper initial contents of MAP buffers, real scalars, and integer scalars.

File "BBN16I" also contains several FORTRAN subroutines called by DCA96I and used to calculate the initial contents of certain buffers. Included are subroutines that generate a square-wave of a given frequency and amplitude (SQWAV) and a Hamming window of a given size (HAMMNG).

2.4.2.2.3 Control Structure Definition (BBN16F)

Subroutine BBN16F defines several function lists that specify the operation of the speech coder system. Function list ID numbers are defined symbolically. The FORTRAN variables containing the function list ID numbers are included in a labeled COMMON block so that they can be referenced by other subroutines.

As part of the function list definition task, subroutine BBN16F produces pre-bound versions of all of the SNAP-II array functions used in the speech coder system. Pre-binding is a SNAP-II operation that causes function parameter information,

normally communicated to the function at execution time, to be "bound" to the function at some earlier time (in this case at system initialization time). Approximately 6100 half-words are required on Bus 1 to store the pre-bound versions of the speech coder array functions.

Function lists ANLZA, ANLZB, SYNZA, and SYNZB are defined for the analysis and synthesis sections of the speech coder. (Since the system is double-buffered, the analysis and synthesis function lists are each defined twice to allow for the different sets of input and output buffers.) These function lists specify the sequence of MAP-300 array and non-array functions, operating on the system buffers and scalars previously defined, that implement the analysis and synthesis processes described in Sections 2.2.1.3 and 2.2.2.3 of this report.

The analysis and synthesis function lists described above are used in the specifications of several other function lists that define the outer structure of the speech coder system. Function list BBNLP defines the real-time speech coder loop, executing function lists ANLZA, SYNZA, ANLZB, and SYNZB when buffer status flags indicate that the input buffers are full and the output buffers are empty for each of these processes. Function list BBNRTS defines the start-up sequence for the real-time speech coder system, starting the Modem, ADAM, and AOM Scroll processors, and repeatedly executing the BBNLP function

list described above. This repeated execution is conditioned on the contents of MAP integer scalar RUN being non-zero; hence, the speech coder can be stopped by setting RUN to zero. Function list BBNRST restarts the real-time speech coder by setting RUN to non-zero and reinitiating the repeated execution of the real-time speech coder loop.

Two other function list definitions are included in this subroutine to support non-real-time file-to-file speech coder operation, speech coder timing operation, and speech coder oscilloscope display operation (for detailed internal timing). These modes of speech coder operation are not supported, and can not be invoked, in the delivered speech coder system. They were used for system development and debugging use under the RT-11 operating system and are included here to aid in future additions to the speech coder system. Function list BBNFFT specifies the speech coder loop with explicit calls to functions that execute the A/D, RMODEM, TMODEM, and D/A interrupt routines. Function list BBNTIM repeatedly invokes a sequence of six BBNFFT function list executions (one for each possible combination of input and output buffers), with software simulation of the transmitter-to-receiver communication path occurring after each BBNFFT execution.

All SNAP-II functions are called via PDP-11 support subroutines that perform the actual communication with the MAP

driver. These subroutines are contained in library SNPLIB for CSPI-supplied functions and in file BBN16H for BBN-written functions.

2.4.2.2.4 System Software Execution (BBN16E)

The execution of the speech coder system is controlled by subroutine BBN16E. This subroutine starts the real-time speech coder system by first loading the Modem, ADAM, and AOM Scrolls with their respective programs, and then executing the real-time speech coder start-up function list (BBNRTS) described in the preceding section. It then interacts with the user, responding to single-character commands to halt the speech coder ('Q'), enable/disable error simulation ('E'), enable/disable error correction ('C'), cause the speech coder to lose sync artificially ('L'), type out a group of speech coder state scalars ('T'), and suspend the controlling RSX-11M task, allowing the speech coder to continue executing in the MAP-300 ('S').

Provision is included in this subroutine for the user to select other speech coder modes of operation, specifically file-to-file, timing, and oscilloscope display modes. However, this mode selection process is bypassed in the delivered speech coder system, and real-time speech coder execution mode is forced. As described in the previous section, these other operating modes are not supported, and cannot be invoked, in the delivered speech coder system. They were used for system development and

debugging and are included here to aid in future additions to the speech coder system. File BBN16D contains dummy versions of various system-dependent subroutines, which constitute the unsupported software portions of these unsupported modes of speech coder operation.

2.5 SYSTEM TIMING PERFORMANCE

The speech coder system introduces a total delay of 9 frames, corresponding to 293.6 milliseconds, between voice input and synthesized voice output (not including any delays in transmission). This consists of 4 frames of delay introduced by the transmitter portion of the system and 5 frames of delay contributed by the receiver portion. In the transmitter, a frame of delay each is produced by the buffering in the ADAM scroll program, the ANALYZE processing module, and the TMODEM scroll program. In addition, the ANALYZE module introduces a frame of "folding" delay due to the concurrent execution of the ANALYZE module and the PROPAR/PRORES modules (see Section 2.2.1).

In the receiver, two frames of delay are contributed by the buffering in the RMODEM scroll program, and a frame each is produced by buffering in the SYNTHESIZE processing module and the AOM scroll program. In addition, the SYNTHESIZE module introduces a frame of "folding" delay due to the concurrent execution of the SYNTHESIZE module and the CORPAR and DECODA/B modules (see Section 2.2.2).

The observed internal timing of the speech coder modules is shown in Fig. 17. This figure reflects data obtained by running the speech coder in a loop without the ADAM, AOM, and MODEM scroll processors and observing the states of the MAP-300 RA flag (APU run) and G-flags with an oscilloscope. The speech coder was provided with simulated A/D input data. ADAM, AOM, and MODEM scroll interrupt routines were omitted from the timing loop. These interrupts were timed separately, and their execution times are indicated at the end of the figure. Including these interrupt routines, Fig. 17 shows a total processing time of 30.0 milliseconds for a 32.625 millisecond frame of data, or about 0.92 times real time. (This is the worst case total processing time. If the ADAM, AOM, or MODEM scroll interrupts were to occur while the AP were running and the CSPU were idle, some or all of the interrupt routine execution time would be hidden behind the concurrent AP routine.)

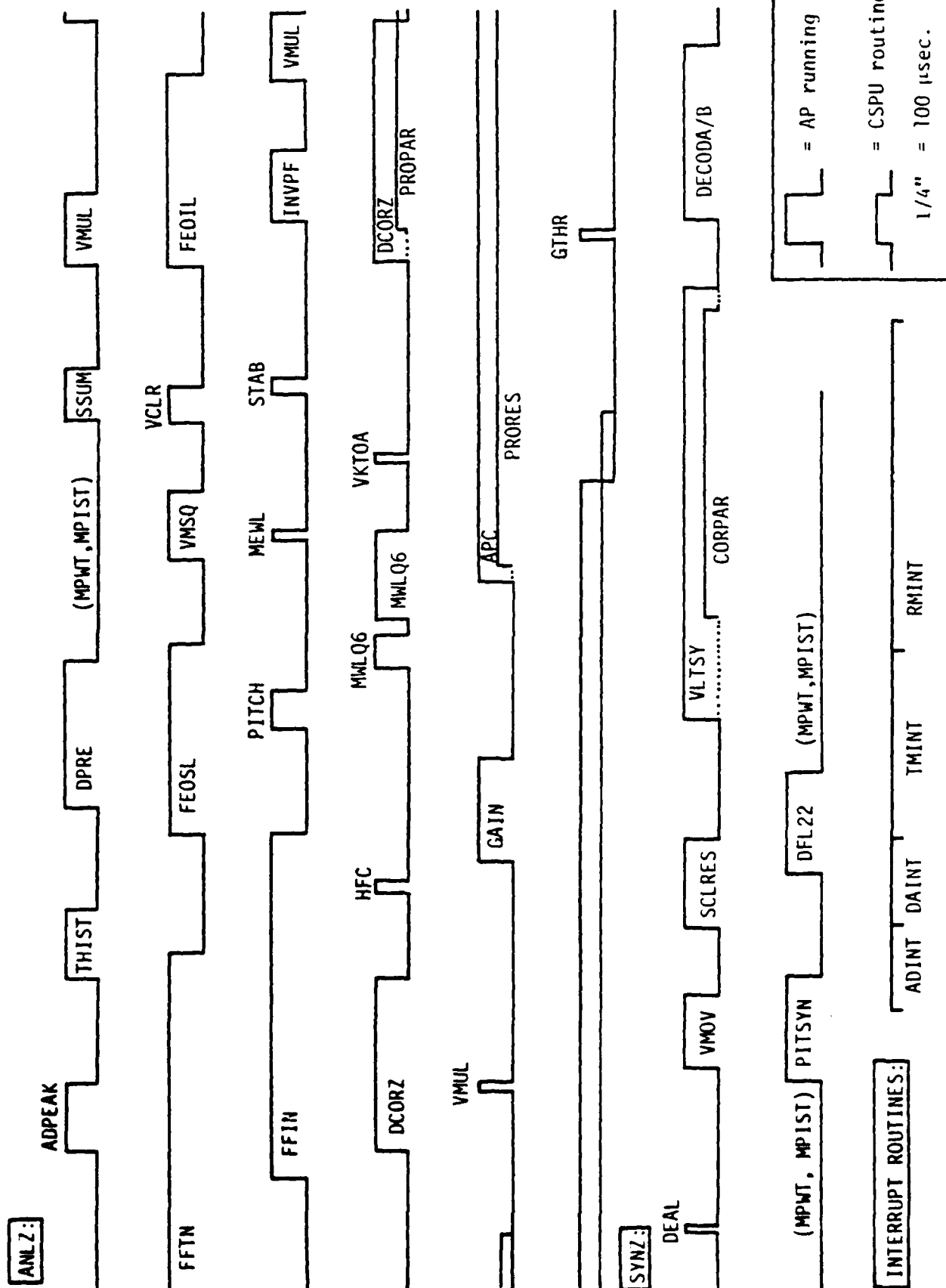


Figure 17 MAP-300 Timing

Bolt Beranek and Newman Inc.

Report No. 4565

3. SOFTWARE OPERATING PROCEDURES

The installation and execution of the speech coder system software are accomplished using the RSX-11M operating system as well as MAP-300 support software supplied by CSPI. It is assumed that the RSX-11M operating system, the MAP-300 device drivers, the MAP-300 loaders, and the SNAP-II software system have all previously been installed prior to the speech coder system installation.

After the speech coder system software has been installed, the speech coder can be executed by using the procedure given in Section 3.1. The software installation procedure is given in Section 3.2.

3.1 SOFTWARE EXECUTION PROCEDURE

The speech coder system execution procedure consists of two major steps. First, both MAP-300 processors in the full-duplex speech coder system must be loaded from the host PDP-11 with identical copies of the speech coder MAP-300 software. Second, two RSX-11M tasks must be run to initialize and start the two MAP-300 speech coders.

The following files must be disk-resident before speech coder execution can be attempted:

BBN16X.MBN

(MAP Binary file)

BBA16.TSK (Host Task for MAP #A)
BBB16.TSK (Host Task for MAP #B)

If these files do not exist, they must be created according to the speech coder system software installation procedure given in Section 3.2.

The following dialog indicates the proper command sequence for full-duplex speech coder system execution. User commands are underscored to differentiate them from computer responses.

```
>SET /UIC=[6,260]
>ALLOCATE MA:                    **allocate MAP #A**
>ALLOCATE MB:                    **allocate MAP #B**
>INSTALL BBA16
>INSTALL BBB16
>RUN [6,100]QLA                  **load MAP #A**
*BBN16X.MBN
TT1 -- STOP

>RUN [6,100]QLB                  **load MAP #B**
*BBN16X.MBN
TT1 -- STOP

>RUN BBA16                        **start MAP #A**
BBN 16K BPS SPEECH CODING SYSTEM

CONFIGURING MAP BUFFERS AND SCALARS...
INITIALIZING MAP BUFFERS AND SCALARS...
PREBINDING MAP FUNCTIONS AND DEFINING FUNCTION LISTS...
EXECUTING BBN16 SPEECH CODER...
SPEECH CODER IS IN OPERATION.

COMMANDS ARE:
S:    SUSPEND TASK (LEAVING SPEECH CODER RUNNING)
Q:    QUIT (HALT SPEECH CODER)
E N:  SIMULATE N ERRORS PER FRAME
C N:  ERROR-CORRECT IF N=0
L:    CAUSE RCVR TO LOSE SYNC
T:    TYPE OUT SPEECH CODER STATE

ENTER COMMAND: S
BBA16 -- PAUSE (SPEECH CODER CONTINUING)
```

```
>RUN BBB16          **start MAP #B**  
BBN 16K BPS SPEECH CODING SYSTEM
```

```
CONFIGURING MAP BUFFERS AND SCALARS...  
INITIALIZING MAP BUFFERS AND SCALARS...  
PREBINDING MAP FUNCTIONS AND DEFINING FUNCTION LISTS...  
EXECUTING BBN16 SPEECH CODER.....  
SPEECH CODER IS IN OPERATION.
```

COMMANDS ARE:

```
S:  SUSPEND TASK (LEAVING SPEECH CODER RUNNING)  
Q:  QUIT (HALT SPEECH CODER)  
E N: SIMULATE N ERRORS PER FRAME  
C N: ERROR-CORRECT IF N=0  
L:  CAUSE RCVR TO LOSE SYNC  
T:  TYPE OUT SPEECH CODER STATE
```

```
ENTER COMMAND: S  
BBB16 -- PAUSE (SPEECH CODER CONTINUING)
```

>

At this point in the command sequence, the full-duplex speech coder system should be in operation. Host tasks BBA16 and BBB16 can be resumed (for interaction with the speech coder software) by invoking the RSX-11M "RESUME" command. The speech coder system can be halted either by responding with "Q" to the "ENTER COMMAND:" typeout from BBA16 and BBB16 or by aborting the BBA16 and BBB16 tasks through the use of the RSX-11M "ABORT" command.

3.1.1 Optional Software Execution Procedure

Throughout the BBN16 Speech Coder MAP software, we have included code to set and clear various general-purpose flags (G-flags). These flags are used mostly for debugging, timing, and system measurement purposes; the manipulations are not required,

in general, for speech coder operation. Although these G-flag manipulations are strictly legal and proper according to CSPI documentation, they seem to induce "Bus 1 Memory Timeout" errors after apparently random amounts of time (ranging from a few minutes to several hours). For this reason, we have delivered with the system a patch file (NOGF16.MOB) which, when loaded into the MAP over the BBN16 speech coder software, no-op's all G-flag manipulations not needed for speech coder operation. This file can be loaded into the MAP processors (following the load of BBN16X) using the following command sequence:

```
>RUN [6,100]MALD
OBJECT INPUT? NOGF16.MOB
BINARY OUTPUT? (carriage return)
TT1 -- STOP 2
>RUN [6,100]MBLD
OBJECT INPUT?NOGF16.MOB
BINARY OUTPUT?(carriage return)
TT1 -- STOP 2
>
```

3.1.2 Analog Input Level Setting

The BBN16 speech coder software includes a facility for monitoring the peak input speech level. This facility allows the level to be set properly when the speech coder is being fed from an adjustable level signal source (such as a tape deck). The peak input level is maintained in integer scalar TADPK. TADPK can be displayed (along with a number of other scalars) during speech coder operation by entering a 'T' in response to the "COMMAND:" prompt. The peak input level range is 0-2047,

so the desired peak input level is approximately 1500-2000. A peak level above about 2000 introduces a potential input signal clipping condition, while a peak level below about 1500 does not take full advantage of the system's available dynamic range. The 'T' command resets TADPK to 0, so the displayed value pertains only to the time interval since the previous 'T' was typed.

3.2 SOFTWARE INSTALLATION PROCEDURE

The speech coder system software installation procedure consists of two major steps. First, the speech coder MAP-300 software must be converted from MAP object format to MAP binary format. Second, two RSX-11M tasks (for initializing and starting the two MAP-300 processors) must be task-built.

The following files must be disk-resident before speech coder software installation can be attempted:

BBN16X.MOB	(MAP object file)
	(This file consists of the following
	concatenated files, with intervening
	"FFFF" lines deleted:
	S300EX.MOB
	EAF300.MOB
	IOS300.MOB
	BBN16M.MOB
	BBN16U.MOB
	BBN16T.MOB
	BBN16P.MOB)
BBN16C.FOR	(HOST FORTRAN module)
BBN16D.FOR	(")
BBN16E.FOR	(")
BBN16F.FOR	(")
BBN16H.FOR	(")
BBN16I.FOR	(")

BBN16R.FOR (")

If these files do not exist, they must be recopied to disk from the RSX PIP-format "BBN 16 kb/s speech coder s/w" magnetic tape delivered with the speech coder system.

The MAP-300 speech coder software should be converted from MAP object format to MAP binary format using the following command sequence:

```
>RUN [6,100]MALD
OBJECT INPUT?BBN16X.MOB
BINARY OUTPUT?BBN16X.MBN
LOAD MAP?(Y OR N) N
TT1 -- STOP 2
>
```

The two RSX-11M tasks (for initializing and starting the two MAP-300 processors) should be generated next. First, all FORTRAN speech coder modules listed above must be compiled. Then the two tasks must be task-built, with each task including the compiled FORTRAN object modules, the CSPI-supplied SNAP host support library modules, and the appropriate CSPI-supplied MAP device driver for MAP #A or MAP #B. The following command sequence will accomplish this procedure:

```
>F4P BBN16C=BBN16C.FOR/CO:35
>F4P BBN16D=BBN16D.FOR/CO:35
>F4P BBN16E=BBN16E.FOR/CO:35
>F4P BBN16F=BBN16F.FOR/CO:35
>F4P BBN16H=BBN16H.FOR/CO:35
>F4P BBN16I=BBN16I.FOR/CO:35
>F4P BBN16R=BBN16R.FOR/CO:35
>TKB **build host task for MAP #A**
```

```
TKB>BBA16/FP/CP=BBN16R,BBN16C,BBN16D
TKB>BBN16E,BBN16F,BBN16H,BBN16I
TKB>[6,100]SNALIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=11
TKB>ASG=TI:7
TKB>ASG=MA:10
TKB>//
>TKB                                **build host task for MAP #B**
TKB>BBB16/FP/CP=BBN16R,BBN16C,BBN16D
TKB>BBN16E,BBN16F,BBN16H,BBN16I
TKB>[6,100]SNBLIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=11
TKB>ASG=TI:7
TKB>ASG=MB:10
TKB>//
>
```

The speech coder system software is now installed. The speech coder can be executed by using the procedure given in Section 3.1.

Bolt Beranek and Newman Inc.

Report No. 4565

4. REFERENCES

1. R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins and W. Russell, "Design and Real-time Implementation of a Baseband LPC Coder for Speech Transmission over 9600 BPS Noisy Channels," Final Report, BBN Report No. 4327, Vols. I and II, Bolt Beranek and Newman Inc., Cambridge, MA, February 1980.
2. "Simple Notation for Array Processing, Version II (SNAP-II) Reference Manual," CSPI Document No. JB6000-017-01, C.S.P. Inc., Billerica, MA, November 1978.
3. "Release Note: SNAP-II Release 3 Executive Features and Expanded Array Function Set," CSPI Document No. DW6000-021-04, C.S.P. Inc., Billerica, MA, July 1979.
4. "Release Note: SNAP-II Extended Array Function Library, II," CSPI Document No. RB6000-24-00, C.S.P. Inc., Billerica, MA, March 1980.
5. "SNAP-II Input/Output Scroll Package, User's Manual, CSPI Document No. JW8400-001-01, C.S.P. Inc., Billerica, MA, July 1979.
6. "Installation Procedure for Release 3 SNAP-II Software System on the DEC RSX-11M System," CSPI Document No. LL8901-004-03, C.S.P. Inc., Billerica, MA, July 1979.
7. "MAP Software Interface Description for the DEC RSX-11M System," CSPI Document No. ST8901-000-04, C.S.P. Inc., Billerica, MA, July 1979.

Bolt Beranek and Newman Inc.

Report No. 4565

APPENDIX A
BBN-WRITTEN FUNCTION DESCRIPTIONS

This Appendix contains a complete user-level description for each MAP-300 SNAP-II function written by BBN and used in the speech coder implementation. The descriptions are ordered alphabetically by function name.

Functions that were used in the speech coder implementation and are not described here were supplied by CSPI as part of the SNAP-II Software System (see Sections 2.4.1.1 and 2.4.2.1).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ADINT
NAME EXPANSION: Simulate A/D interrupt

FCB #: 124
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: ADAMINT

PARAMETER DEFINITIONS:			
<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

This SNAP callable function calls the ADAMINT A/D Interrupt Service module in exactly the same manner as it is used to respond to A/D interrupts, so it may be used to simulate such an interrupt. See Section 2.2.1.2.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ADPEAK (U)
NAME EXPANSION: Monitor peak signal from the A/D

FCB #: 182
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: ADPKU
APS MODULE NAME: ADPKA
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Buffer U	1-63	Real	Input

NUMBER OF OUTPUT SAMPLES: 1, in integer scalar TADPK
NOTES: --

FUNCTION DESCRIPTION:

Maintains a running peak absolute value of the data in the input buffers given to it on successive calls. The running peak is maintained as an integer value (to facilitate monitoring the peak level) in the integer scalar TADPK. Since SNAP-II standard binding does not support binding an integer scalar to an array function, the address of TADPK is "hard bound" in the APS module source code.

The input buffer is assumed to contain signals in the range -1 to +1 from a 12-bit A/D converter. The integer value maintained in TADPK falls in the range 0-2047.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: APC(Y,A,U,V,W,R,S,T)
NAME EXPANSION: APC loop

FCB #: 199
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: APCUS
APS MODULE NAME: APCSS
CSPU MODULE NAME: SBMSAPC

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Fixed	Output: Coded residual samples
Real Scalar A	0-191	Real	Input: Pitch lag
Buffer U	1-63	Real (YBS)	Input: Preemphasized speech
Buffer V	1-63	Real (15)	Input: Filter coeffs: 6 spectral, 3 pitch, 6 noiseshaping.
Buffer W	1-63	Real (6)	Input: Quantizer segment scale factors, alternating with their reciprocals.
Buffer R	1-63	Real (432)	Input/Output: Pitch Filter History: last frame, current frame.
Buffer S	1-63	Real (6)	Input/Output: Spectral Filter History
Buffer T	1-63	Real (6)	Input/Output: Noise Shaping Filter History

NUMBER OF OUTPUT SAMPLES: $YBS + RBS/2 + SBS + TBS$

NOTES: Uses SA to modify APS code.
Requires special CSPU binding.

FUNCTION DESCRIPTION:

Compute and code APC residual. See Section 2.7 of the Algorithm Specification (Volume I, Appendix A of this report) for complete description.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: CORPAR(I)
NAME EXPANSION: Correct parameters

FCB #: 123
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: CORPAR

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Literal I	-2,0	Integer	Specifies buffer pair

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Unbitstreams, error-corrects, and decodes parameters from one of the RBITS buffers to one of the RSOURCE buffers. The tables used for decoding each parameter are given in the file BBN16T.MSO.

The argument I specifies which pair of RBITS/RSOURCE buffers is used. I=-2 means RBTB/RSRA, and I=0 means RBTB/RSRB.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DAINI
NAME EXPANSION: Simulate D/A Interrupt

FCB #: 127
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: AOMINT

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

The SNAP-callable function calls the AOMINT D/A Interrupt Service module, so it may be used to simulate such an interrupt. See Section 2.2.2.5.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DEAL(Y,A,U,B,V,W)

NAME EXPANSION: Deal received decoded data from single buffer

FCB #: 212

ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DEALUS

APS MODULE NAME: DEALSS

CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (230)	Input: Received decoded data buffer
Real Scalar A	0-191	Real	Output: Received $\sqrt{\text{Energy}}$
Buffer U	1-63	Real (3)	Output: Received $\sqrt{\text{Delta Gains}}$
Real Scalar B	0-191	Real	Output: Received Pitch Lag
Buffer V	1-63	Real (3)	Output: Received Pitch Predictor Coeffs
Buffer W	1-63	Real (6)	Output: Received Spectral Predictor Coeffs

NUMBER OF OUTPUT SAMPLES: 14

NOTES:

FUNCTION DESCRIPTION:

On function exit, Y(0 - 215) is expected to contain 216 received residual samples.

SA <= Y(216)
U(0-2) <= Y(217-219)
SB <= Y(220)
V(0-2) <= Y(221-223)
W(0-5) <= Y(224-229)

AD-A096 092

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA

F/B 17/2.1

DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)

DEC 80 J J WOLF, K D FIELD, W H RUSSELL

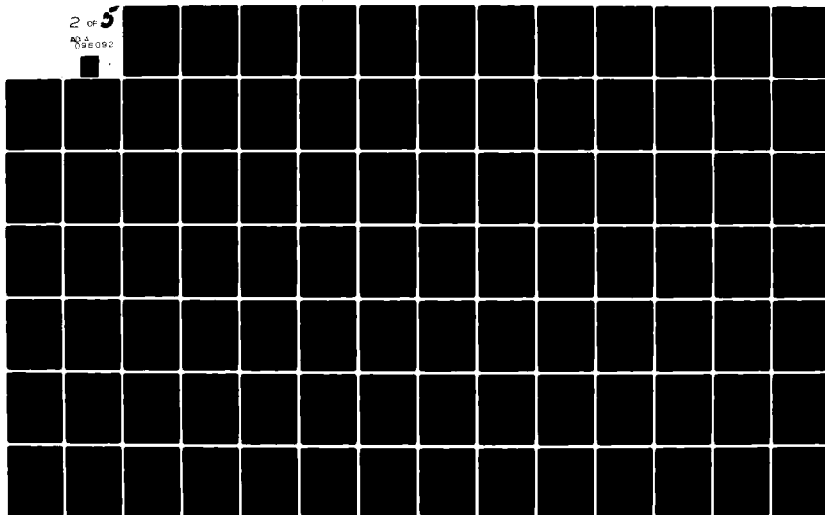
DCA100-79-C-0037

UNCLASSIFIED

BBN-4565-VOL-2

NL

2 of 5
NO. 096092



BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DECODA
NAME EXPANSION: Decode residual, buffer-pair "A"

FCB #: 188
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DCDU
APS MODULE NAME: DCDSA
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Unbitstreams and decodes the residual samples from buffer RBTA to buffer RSRA. The table used for decoding the residual is given in the file BBN16T.MSO.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DECODB
NAME EXPANSION: Decode residual, buffer-pair "B".

FCB #: 189
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DCDU
APS MODULE NAME: DCDSB
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Unbitstreams and decodes the residual samples from buffer RBTB to buffer RSRB. The table used for decoding the residual is given in the file 88N16T.M50.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: FFILR(Y,M,U,V,W)
NAME EXPANSION: Fast Fourier Transform, Not-in-Place, To Real Data, Long

PCB #: 198, 206

ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: --

APS MODULE NAME: -- (see below)

CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>

(see below)

NUMBER OF OUTPUT SAMPLES: USB*2

NOTES: One complex sample is input beyond the end of the configured U buffer (see below).

FUNCTION DESCRIPTION:

This is a slightly modified version of the CSPI function FFINR (from SNAP-II, Extended Array Function Library II [4]). The modification involves an assumed reformatting of the complex frequency domain input (U) such that the last real input point is assumed to appear as such, rather than in the first imaginary input location (which is known to be zero).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: FFTLR(Y,M,U,V,W)
NAME EXPANSION: Fast Fourier Transform, Not-in-Place, On Real Data, Long

FCB #: 204, 197

ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: --
APS MODULE NAME: -- (see below)
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>

(see below)

NUMBER OF OUTPUT SAMPLES: $UBS/2 + 1$

NOTES: One complex sample is output beyond the end of the configured Y buffer (see below).

FUNCTION DESCRIPTION:

This is a slightly modified version of the CSPI function FFTNR (from SNAP-II, Extended Array Function Library II [4]). The modification involves a reformatting of the complex frequency domain output (Y) such that the last real output point appears as such, rather than in the first imaginary output location (which is known to be zero).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: GAIN(Y,A,U,B,V,C)
NAME EXPANSION: Compute Quantizer Segment Scale Factors

FCB #: 196
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: GAINS
APS MODULE NAME: GAINSS
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (3)	Output: Quantizer segment scale factors, alternating with their reciprocals.
Real Scalar A	0-191	Real	Output: Coded frame energy
Buffer U	1-63	Real (3)	Output: Coded segment delta-gains
Real Scalar B	0-191	Real	Input: 1/72
Buffer V	1-63	Real (216)	Input: Second Residual Samples
Real Scalar C	0-191	Real	Input: 1/3

NUMBER OF OUTPUT SAMPLES: 7 total

NOTES: Fixed to process 216 input buffer V elements, in 3 segments of 72 elements each.

FUNCTION DESCRIPTION:

$$SG(n) = \frac{1}{72} \sum_{i=1}^{72} (V(i+(n*72)))^2 \quad n = 0,1,2$$

$$G = \frac{1}{3} \sum_{i=1}^{72} (SG(i)) ; GH = \text{quantized } \sqrt{G}; SA = \text{coded } G$$

$$DG(n) = SG(n)/(GH*GH) \quad n = 0,1,2$$

$$U(n) = \text{coded } DG(n) ; DGH(n) = \text{quantized } \sqrt{DG(n)} \quad n=0,1,2$$

$$\left. \begin{aligned} Y(2n) &= (GH*DGH(n)) \\ Y(2n+1) &= 1./Y(2n) \end{aligned} \right\} \quad n=0,1,2$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: GTHR(Y,A,U,B,V,W)

NAME EXPANSION: Gather Transmission data into single buffer

FCB #: 200

ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: GTHRUS

APS MODULE NAME: GTHRSS

CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Buffer Y	1-63	Fixed (230)	Output: Transmission buffer
Real Scalar A	0-191	Fixed (in lefthword)	Input: Coded Energy
Buffer U	1-63	Fixed (3)	Input: Coded Delta Gains
Real Scalar B	0-191	Fixed (in lefthword)	Input: Coded Pitch Lag
Buffer V	1-63	Fixed (3)	Input: Coded Pitch Predictor Coeffs.
Buffer W	1-63	Fixed (6)	Input: Coded Spectral Predictor Coeffs.

NUMBER OF OUTPUT SAMPLES: YBS-216

NOTES:

FUNCTION DESCRIPTION:

On function call, Y(0-215) is expected to contain 216 coded residual samples.

Y(216) <= SA
Y(217-219) <= U(0-2)
Y(220) <= SB
Y(221-223) <= V(0-2)
Y(224-229) <= W(0-5)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: HFC(Y,U,V)
NAME EXPANSION: High Frequency Correction

FCB #: 190
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: HFCUS
APS MODULE NAME: HFCSS
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: HFC'd autocorrelation coeffs
Buffer U	1-63	Real	Input: Autocorrelation coeffs
Buffer V	1-63	Real (3)	Input: LAMBDA* $\mu(i)$ LAMBDA = .035 $\mu = \begin{pmatrix} .375 \\ -.25 \\ .0625 \end{pmatrix}$

NUMBER OF OUTPUT SAMPLES: YBS
NOTES:

FUNCTION DESCRIPTION:

For $i = 0, 2$:

$$Y(i) = U(i) + E2*V(i)$$

For $i = 2, (YBS-1)$:

$$Y(i) = U(i)$$

$$K1 = U(1)/U(0)$$

$$E1 = (1 - K1^2)*U(0)$$

$$K2 = (U(2) - K1*U(1))/E1$$

$$E2 = E1*(1 - K2^2)$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: INVPF(Y,A,U,V)
NAME EXPANSION: 3-tap Inverse Pitch Filter

FCB #: 167
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: INVPU5
APS MODULE NAME: INVPSI
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: filter output
Real Scalar A	50-191	Real	Input: pitch in sample-times
Buffer U	1-63	Real (YBS+216)	Input: filter input, including previous 216 samples.
Buffer V	1-63	Real (3)	Input: Filter coefficients

NUMBER OF OUTPUT SAMPLES: YBS

NOTES: This routine uses SA to modify its APS code. Buffer U must be compact 32-bit floating point.

FUNCTION DESCRIPTION:

Implements a 3-tap FIR pitch filter. Let M = SA, the pitch in terms of sample times:

$$y(n) = u(n) + \sum_{j=0}^2 v(j) * u(n - M - j + 1), \quad n=0,1,\dots,YBS-1$$

The pitch can take on values $14 \leq M \leq 133$, so the buffer U contains the "previous" frame of 216 samples in addition to the "current" frame. That is, $U(0) = u(-216)$, $U(216) = u(0)$, etc.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPGSC(GFLAG,SETCLR)
NAME EXPANSION: G-Flag Set/Clear

FCB #: 106
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPGSC

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Literal flag	0-3	Integer	Selects G-flag
Literal SETCLR	0-1	Integer	SETCLR=0=>Set flag SETCLR=1=>Clear flag

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

MPGSC sets or clears one of the four G-flags. This is useful for program timing using an external oscilloscope.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPIFF(IA,IB, FLID)

NAME EXPANSION: If (IA.NE.0) & (IB.EQ.0) Conditional Function List Execution

FCB #: 105

ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -

APS MODULE NAME: -

CSPU MODULE NAME: MPIFFS

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Integer Scalar IA	0-127	Integer	Input
Integer Scalar IB	0-127	Integer	Input
Literal FLID	0-63	Integer	Input: Function List ID

NUMBER OF OUTPUT SAMPLES: -

NOTES: Function list 'FLID' must be previously defined.

FUNCTION DESCRIPTION:

Function list 'FLID' is executed if and only if Integer Scalar IA is not equal to zero, and Integer Scalar IB is equal to zero.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MWLQ6(Y,A,U,V,W)
NAME EXPANSION: Matrix (Weinter-Levinson-Durbin) Solution, with
 Quantized and coded outputs, 6th order
FCB #: 135
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: MWLFSAPU
APS MODULE NAME: MWLFSAPS
CSPU MODULE NAME: MWLQSSSM

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Buffer Y	1-63	Real (12)	Output: Reflection coefficients and error terms (as in MWLD)
Real Scalar A	0-191	Real	Input: Threshold value (as in MWLD)
Buffer U	1-63	Real (6)	Output: Quantized reflection coefficients
Buffer V	1-63	Real (7)	Input: Autocorrelation coefficients (as in MWLD).
Buffer W	1-63	Long Fixed (6)	Output: Coded reflection coefficients.

NUMBER OF OUTPUT SAMPLES: 6

NOTES: Special intermediate CSPU support is used.

FUNCTION DESCRIPTION:

This module consists of the SNAP-II function MWLD followed by a special quantization/coding module. MWLD outputs LPC coefficients into the buffer U, but the Q/C module overwrites them with the quantized reflection coefficients. The threshold and quantized-value tables are not specified as parameters, but are referenced symbolically by the APS module.

MWLD requires that buffers Y,U,V be compact (SI=1). Buffer W must be compact also.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PITCH(Y,A,U,B,C,D)
NAME EXPANSION: Compute and code pitch

FCB #: 134
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PITUS
APS MODULE NAME: PITSS
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (3)	Output: Autocorrelation coefficients -RP(Pitch lag -1) -RP(Pitch lag) -RP(Pitch lag +1)
Real Scalar A	0-191	Real	Output: Pitch lag (in samples)
Buffer U	1-63	Real (512)	Input: Autocorrelation coefficients
Real Scalar B	0-191	Fixed (in lefthword)	Output: Coded pitch
Real Scalar C	0-191	Real	Input: Constant 14.
Real Scalar D	0-191	Real	Output: Single tap coefficient

NUMBER OF OUTPUT SAMPLES: 3 Output Scalars, 3 Output Buffer samples

NOTES: Uses modified SMAX.
Only uses part of input buffer U.
Uses SA to modify APS code

FUNCTION DESCRIPTION:

A = index of $\max_{i=14}^{133} [U(i)]$; where first U element is U(0).

Left half-word of B = A - 14. = A - C

(This value is generated by multiplying by 2**-15, and ALIGN'ing.)

$D = \frac{-U(\text{pitch lag})}{U(0)}$ (single tap coefficient (for case of 3-taps unstable))

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PITSYN(Y,A,U,V)
NAME EXPANSION: 3-tap direct form pitch-synthesis recursive filter

FCB #: 203
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PITSUS
APS MODULE NAME: PITSSI
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u> (length)	<u>COMMENTS</u>
Buffer Y	1-63	Real (UBS+216)	Output: filter output plus 216 preceding output samples.
Real Scalar A	50-191	Real	Input: Pitch in samples
Buffer U	1-63	Real	Input: Filter input
Buffer V	1-63	Real (3)	Input: Filter Coefficients

NUMBER OF OUTPUT SAMPLES: UBS

NOTES: Uses value of SA to modify the APS code. Buffer Y must be compact 32-bit floating point.

FUNCTION DESCRIPTION:

Implements a 3-pole recursive filter for pitch synthesis. Let M=SA, the pitch expressed in samples

$$y(n) = u(n) - \sum_{j=0}^2 v(j) * y(n-M-j+1), \quad n=0,1,\dots,UBS-1$$

The pitch can take on values $14 \leq M \leq 133$, so the buffer Y must contain "previous" samples; hence, it functions as both input (previous output) and output (output computed by the function). For buffer-configuration reasons, Y must contain the previous 216 samples of output.

Buffer Y is defined so that the first value in it corresponds to y(-216).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PROPAR(I)
NAME EXPANSION: Protect analysis parameters

FCB #: 122
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: PROPAR

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Literal I	-2,0	Integer	Specifies buffer pair

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Error protects and bitstreams analysis parameters from one of the TSINK buffers into one of the TBITS buffers; also accumulates histogram statistics for all the analysis parameters except pitch.

The argument I specifies which pair of TSINK/TBITS buffers is used. I=-2 means TSINKA/TBITA, and I=0 means TSINKB/TBITB.

See Section 2.2.1.3.18.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PRORES(I)
NAME EXPANSION: "Protect" residual samples

FCB #: 121

ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --

APS MODULE NAME: --

CSPU MODULE NAME: PRORES

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Literal I	-2,0	Integer	Specifies buffer pair

NUMBER OF OUTPUT SAMPLES: --

NOTES:

FUNCTION DESCRIPTION:

Bitstreams coded residual samples from one of the TSINK buffers into one of the TBITS buffers. No "protection" is done.

The argument I specifies which pair of TSINK/TBITS buffers is used. I=-2 means TSNKA/TBTA, and I=0 means TSNKB/TBTB.

See Section 2.2.1.3.18.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RMINT
NAME EXPANSION: Simulate Receiver Modem Interrupt

FCB #: 126
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: RMODEMINT

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the RMODEMINT Receiver Modem Interrupt Service module, so it may be used to simulate such an interrupt. See Section 2.2.2.2.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SCLRES(Y,A,U,V)
NAME EXPANSION: Compute Quantizer scale factors and scale residual

FCB #: 202
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: SCLS
APS MODULE NAME: SCLSS
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (216)	Output: Scaled residual samples
Real Scalar A	0-191	Real	Input: Received Frame Energy
Buffer U	1-63	Real (3)	Input: Received Delta Gains
Buffer V	1-63	Real	Input: Received Residual samples

NUMBER OF OUTPUT SAMPLES: 216
NOTES:

FUNCTION DESCRIPTION:

$GFAC(j) = [SA * U(j)]$ $j = 1, 2, 3$
 $Y(i) = V(i) * GFAC(j)$ $j = 1$ for $i = 1, 72$
 $j = 2$ for $i = 73, 144$
 $j = 3$ for $i = 145, 216$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: STAB(Y,A,U,V)
NAME EXPANSION: Stability check on pitch predictor coefficients; code and quantize.

FCB #: 150

ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: STABS
APS MODULE NAME: STABSS
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Quantized stable pitch predictor coeffs.
Real Scalar A	0-191	Real	Input: Single Tap coeff.
Buffer U	1-63	Fixed	Output: Coded stable pitch predictor coeffs.
Buffer V	1-63	Real	Input: Pitch Predictor coefficients.

NUMBER OF OUTPUT SAMPLES: 3 Quantized, 3 coded values

NOTES: Module fixed to process 3 pitch predictor coeffs.

FUNCTION DESCRIPTION:

Do stability check on $V(0)$, $V(1)$, $V(2)$:

$$T1 = V(0) + V(1) + V(2)$$

$$T2 = V(0) - 2V(1) + V(2)$$

$$T3 = V(0) - V(2)$$

If $[(T1 < -1) .OR. (T2 < -1) .OR. (T3 < -1) .OR. (T2 > +2) .OR. (T3 > +1)]$

then unstable: set $C1 = 0$
 $C2 = SA$
 $C3 = 0$

Else, stable: set $C1 = V(0)$
 $C2 = V(1)$
 $C3 = V(2)$

Code and quantize $C1$, $C2$, $C3$, set Y buffer to quantized values,
set U buffer to coded values.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: THIST(Y,U,V,W,R,S)
NAME EXPANSION: Set up transmitter frame histories.

FCB #: 201
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: THUS
APS MODULE NAME: THSS
CSPU MODULE NAME: SBMSTH

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Preemphasized speech: history,current frame
Buffer U	1-63	Real (216)	Input: Preemphasized speech: current
Buffer V	1-63	Real	Output: First residual: history,current frame
Buffer W	1-63	Real (6)	Input: First residual: last 6 elements of current frame
Buffer R	1-63	Real	Output: APC pitch filter history last,current frame
Buffer S	1-63	Real (216)	Input: APC pitch filter history: current frame

NUMBER OF OUTPUT SAMPLES: UBS + WBS + SBS

NOTES: Requires special CSPU binding.

FUNCTION DESCRIPTION:

$Y(0) - Y(UBS - 1) \leq U(0) - U(UBS - 1)$
 $V(0) - V(WBS - 1) \leq W(0) - W(WBS - 1)$
 $R(0) - R(SBS - 1) \leq S(0) - S(SBS - 1)$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: TMINT
NAME EXPANSION: Simulate Transmitter Modem Interrupt

FCB #: 125
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: TMODEMINT

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the TMODEMINT Transmitter Modem Interrupt Service module, so it may be used to simulate such an interrupt. See Section 2.2.1.5.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VKTOA(Y,U)
NAME EXPANSION: Convert Reflection Coefficients (K) to Linear Prediction Coefficients (A)

FCB #: 133

ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VKAUS

APS MODULE NAME: VKASI

CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Buffer Y	1-63	Real	Output: LP coefficients A(1), A(2), ..., A(UBS)
Buffer U	1-63	Real	Input: Reflection Coefficients K(1), K(2), ..., K(UBS)

NUMBER OF OUTPUT SAMPLES: UBS

NOTES: This function is different from VKTOA in the 9.6 kb/s MAP-300 system; that was specific to an 8-pole system; this version is general. Also, this version does not output A(0).

FUNCTION DESCRIPTION:

The LPC coefficients $A(i)$, $i = 0, 1, \dots, p$ are computed from the reflection coefficients $K(j)$, $j = 1, 2, \dots, p$, by the recursion:

$$A_m(m) = K(m)$$

$$A_m(L) = A_{m-1}(L) + K(m)A_{m-1}(m-L) \quad \begin{matrix} L = 1, \dots, m-1 \\ m = 1, \dots, p \end{matrix}$$

where $A_m(L)$ indicates the value of the L-th coefficient on the m-th iteration.

The output buffer does not contain A(0), which is 1.0. The first output value is A(1).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VLTSY(Y,U,V,W)
NAME EXPANSION: Lattice Synthesis Filter

FCB #: 132
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VLTSY\$
APS MODULE NAME: V3200\$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

<u>PARAMETER</u>	<u>RANGE</u>	<u>SAMPLE TYPE</u>	<u>COMMENTS</u>
Buffer Y	1-63	Real	Output synthetic speech samples
Buffer U	1-63	Real(8)	Filter memory (G's)
Buffer V	1-63	Real(8)	Reflection coefficients (K's)
Buffer W	1-63	Real	Input samples

NUMBER OF OUTPUT SAMPLES: WBS

NOTES: Exactly 8 elements from each U and V are used. Contents of U are changed.

FUNCTION DESCRIPTION:

Performs lattice form all-pole filter, using reflection coefficients as filter coefficients.

Implements the following Fortran code:

```
DO 20 I=1, WBS
  F(7)=W(I)-G(7)*K(8)
  DO 10 J=6,0, -1
    F(J)=F(J+1)-G(J)*K(J+1)
    G(J+1)=G(J)+F(J)*K(J+1)
    G(0)=F(0)
  10  Y(I)=F(0)
  20
```

APPENDIX B
MAP-300 BUFFERS

The table contained in this Appendix describes the characteristics of each data buffer used in the MAP-300 speech coder implementation. The table entry labeled "BID" indicates the buffer identification number (for SNAP-accessible buffers) or is blank (for non-SNAP buffers). The entries labeled "Written by:" and "Read by:" indicate subroutines, function lists, or program modules that so access each buffer. (The A/D, D/A, transmitter modem, and receiver modem interrupt service routines are specified in these tables by "ADINT", "DAINT", "TMINT", and "RMINT", respectively.)

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
CTHR1 } CTHR3 }	-	Coding Table of Pitch Tap Thresholds K(1, 3)	Fixed Address	--	ANLZ (Pitch)	1	RL LING	8	38912	1	(see BBN161)	1
CTHR2	--	Coding Table of Pitch Tap Thresholds K(2)	Fixed Address	--	ANLZ (Pitch)	1	RL LING	16	38928	1	(see BBN161)	1
CTHR1	--	Coding Table of K(1) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LING	64	38960	1	(see BBN161)	1
CTHR2	--	Coding Table of K(2) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LING	32	39088	1	(see BBN161)	1
CTHR3	--	Coding Table of K(3) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LING	16	39152	1	(see BBN161)	1
CTHR4	--	Coding Table of K(4) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LING	16	39184	1	(see BBN161)	1
CTHR5	--	Coding Table of K(5) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LING	16	39216	1	(see BBN161)	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
CTHR6	--	Coding Table of K(6) Thresholds	Fixed Address	--	AMLZ (Spec)	1	RL LNG	16	39248	1	(see BBR16T)	1
CTHG	--	Coding Table of Energy Thresholds	Fixed Address	--	AMLZ (Gain)	1	RL LNG	64	39280	1	(see BBR16T)	1
CTHNG	--	Coding Table of Delta Gain Thresholds	Fixed Address	--	AMLZ (Gain)	1	RL LNG	4	39408	1	(see BBR16T)	1
CTHU	--	Coding Table of Residual Sample Thresholds	Fixed Address	--	AMLZ (APC)	1	RL LNG	4	39416	1	(see BBR16T)	1
DWL C1 DWL C3	--	Decoding Table of Pitch Tap Values C1, C3 (3bits)	Fixed Address	--	AMLZ (Pitch) CORPAR	1	RL LNG	8	39424	1	(see BBR16T)	1
DWL C2	--	Decoding Table of Pitch Tap Values C2 (4bits)	Fixed Address	--	AMLZ (Pitch) CORPAR	1	RL LNG	16	39440	1	(see BBR16T)	1
DWL K1	--	Decoding Table of Reflection Coefficient Values K(1) (6 bits)	Fixed Address	--	AMLZ (Spec) CORPAR	1	RL LNG	64	39472	1	(see BBR16T)	1

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Increment	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
DWL K2	--	Decoding Table of Reflection Coefficient Values K(2) (5 bits)	Fixed Address	--	ANL Z (Spec) CORPAR	1	RL LING	32	39600	1	(see BBN161)	1
DWL K3	--	Decoding Table of Reflection Coefficient Values K(3) (4 bits)	Fixed Address	--	ANL Z (Spec) CORPAR	1	RL LING	16	39664	1	(see BBN161)	1
DWL K4	--	Decoding Table of Reflection Coefficient Values K(4) (4 bits)	Fixed Address	--	ANL Z (Spec) CORPAR	1	RL LING	16	39696	1	(see BBN161)	1
DWL K5	--	Decoding Table of Reflection Coefficient Values K(5) (4 bits)	Fixed Address	--	ANL Z (Spec) CORPAR	1	RL LING	16	39728	1	(see BBN161)	1
DWL K6	--	Decoding Table of Reflection Coefficient Values K(6) (4 bits)	Fixed Address	--	ANL Z (Spec) CORPAR	1	RL LING	16	39760	1	(see BBN161)	1
DWL G	--	Decoding Table of Energy Value (6 bits)	Fixed Address	--	ANL Z (Gain) CORPAR	1	RL LING	64	39792	1	(see BBN161)	1
DWL DG	--	Decoding Table of Delta Gain Values (2 bits)	Fixed Address	--	ANL Z (Gain) CORPAR	1	RL LING	4	39920	1	(see BBN161)	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BIT	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
DVLU	--	Decoding Table of Residual Samples Values (2 bits)	Fixed Address	--	ANLZ (APC)	1	RL LNG	4	39928	1	(see BBN161)	1
DVLUF	--	Folded Decoding Table of Residual Samples	Fixed Address	--	DECODE/B	1	RL LNG	4	39936	1	(see BBN161)	1
HISTO	--	Histogram Data	Fixed Address	PROPAR	HIST16.SAV	1	RL LNG	160	39944	1	zeros	2
TADBA	--	A/D Input from ADAM	--	ADAM Scroll Program	ADINT	1	RL SHRT	216	40264	1	--	2
TADRB	--	A/D Input from ADAM	--	ADAM Scroll Program	ADINT	1	RL SHRT	216	40480	1	--	2
TADBC	--	A/D "On-hook" Tone Data	--	BBN161	ADINT	1	RL SHRT	216	40696	1	Approx. 240 Hz Square wave	1
TSRA	1	Input Speech to ANLZA	--	ADINT	ANLZ (Premp)	1	RL SHRT	216	40912	1	--	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TSRB	2	Input Speech to ANL ZB	--	ADINT	ANLZ (PREMP)	1	RL SHRT	216	41128	1	--	2
TUNLS	3	Unity Vector	(Constant vector)	BBR161	ANLZ (PREMP) (SPEC)	0	RL LNG	(216)	0	3	Ones (Constant vector)	1
TSP0	4	Preemphasized speech (current frame)		ANLZ (PREMP)	ANLZ (PITCH)	1	RL LNG	(216)	434	3	--	2
TSP1	5	Preemphasized speech (last frame and current frame)	(see TSP0)	ANLZ (PREMP)	ANLZ (PITCH) (INVPF)	1	RL LNG	432	2	3	zeros	2
TSP	6	Preemphasized speech (last 49 of last frame, all 216 of current frame)	(see TSP0)	ANLZ (THIST) (PREMP)	ANLZ (PITCH)	1	RL LNG	(265)	336	3	--	2
THAMP	7	Hanning window coefficients for 265 sample window	--	BBR161	ANLZ (PITCH)	1	RL LNG	265	0	2	265 Hanning Window Coefficients	1
TWSP	8	Windowed speech		ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	(265)	866	3	--	2

MAP-300 Buffers

Usage key: 1 - Constant
2 - Variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
WSPZ	9	Windowed speech with appended zeros	(see TWSP)	BBN161	ANLZ (PITCH)	1	RL LNG	512	866	3	zeros	2/1
TCOST	10	Cosine table for FFT	--	BBN161	ANLZ (PITCH)	1	RL LNG	256	1890	3	cosine table	1
TXSP	11	Frequency domain representation of: TWSPZ (this buffer contains 256 complex points; a 257th complex point will be assumed present directly following it).	Each complex sample: $\left[\begin{array}{c} \text{real} \\ \text{imag.} \end{array} \right]$ TXSPR(i) TXSPI(i)	ANLZ (PITCH)	ANLZ (PITCH)	1	CMPLX LNG	(256)	2402	3	--	2
TXSPR	12	Real part of TXSP, including 257th point	(see TXSP)	ANLZ (PITCH)	ANLZ (PITCH)	2	RL LNG	257	2402	3	--	2
TXSPI	13	Imaginary part of TXSP, including 257th point	(see TXSP)	ANLZ (PITCH)	ANLZ (PITCH)	2	RL LNG	257	2404	3	--	2
TPSP	14	Complex buffer of TPSPR (this buffer contains 256 complex points; a 257th complex point will be assumed present directly following it).	Each complex sample: $\left[\begin{array}{c} \text{real} \\ \text{imag.} \end{array} \right]$ TPSPR(i) TPSPI(i)	ANLZ (PITCH)	ANLZ (PITCH)	1	CMPLX LNG	(256)	3430	3	--	2
TPSPR	15	Power spectrum of TXSP comprising real part of TPSP, including 257th point.	(see TPSP)	ANLZ (PITCH)	ANLZ (PITCH)	2	RL LNG	257	3430	3	--	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TPSPI	16	Zeros, comprising imaginary part of TPSP, including 257th point	(see TPSP)	BBN161	ANLZ (PITCH)	2	RL LNG	257	3432	3	zeros	2
TRP	17	Autocorrelation coefficients for pitch determination (IFFT (TPSP))	--	ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	512	4458	3	--	2
TC	18	Pitch predictor coeffs	--	ANLZ (PITCH)	ANLZ (INVPF)	1	RL LNG	3	5482	3	--	2
TRPP	19	Autocorrelation coeffs. -TRP(pitch lag -1), -TRP(pitch lag), -TRP(pitch lag +1)	--	ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	3	5488	3	--	2
TWORK	20	Work area for MLWL	--	ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	6	5494	3	--	2
TIC	21	Coded pitch predictor coefficients	--	ANLZ (PITCH)	ANLZ (GTHR)	1	FXD LNG	3	5506	3	--	2
TCH	22	Quantized pitch predictor coefficients	$\begin{array}{c} \boxed{1} \leftarrow 6 \rightarrow \boxed{+3} \rightarrow \boxed{+6} \rightarrow \\ \leftarrow \boxed{+1} \boxed{+2} \boxed{+3} \boxed{+4} \boxed{+5} \boxed{+6} \rightarrow \\ \boxed{+1} \boxed{+2} \boxed{+3} \boxed{+4} \boxed{+5} \boxed{+6} \rightarrow \\ \boxed{+1} \boxed{+2} \boxed{+3} \boxed{+4} \boxed{+5} \boxed{+6} \rightarrow \end{array}$	ANLZ (PITCH)	ANLZ (INVPF) (APC)	1	RL LNG	3	5524	3	--	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	ID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TE10	23	First residual (current frame)		ANLZ (INVPF)	ANLZ (SPEC)	1	RL LMG	(216)	5554	3	--	2
TE11	24	First residual (last 6 samples of current frame)	(see TE10)	ANLZ (INVPF)	ANLZ (INVSF)	1	RL LMG	(6)	5974	3	--	2
TE1	25	First residual (last 6 samples of last frame, plus all of current frame)	(see TE10)	ANLZ (INVPF) (THIST)	ANLZ (INVSF)	1	RL LMG	222	5542	3	zeros	2
TI0M1	26	Hanning window coeffs for 216 sample window	--	BHRT61	ANLZ (SPEC)	1	RL LMG	216	530	2	216 Hanning Window Coefficients	1
TF1	27	Windowed First Residual	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LMG	216	962	2	--	2
TR5	28	Autocorrelation coeffs for Spectral Analysis	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LMG	7	5986	3	--	2
TI0M1	29	LAMBDA*MI(n) for HFC	--	BHRT61	ANLZ (SPEC)	1	RL LMG	3	6000	3	$\begin{pmatrix} .035 \\ .035 \end{pmatrix} * \begin{pmatrix} .375 \\ -.25 \\ .0625 \end{pmatrix}$	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TRSP	30	HFC'd Autocorrelation Coeffs.	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LNG	7	6006	3	--	2
TKE	31	Reflection Coeffs. Error terms from PML Q6	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RI LNG	12	6020	3	--	2
TKH	32	Quantized Reflection Coeffs.	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LNG	6	6044	3	--	2
TIK	33	Coded Reflection Coeffs.	--	ANLZ (SPEC)	ANLZ (GTHR)	1	FXD LNG	6	6056	3	--	2
TAH (THH1)	34	Predictor Coeffs (also start of filter coeff block)	(see TCH)	ANLZ (SPEC)	ANLZ (SPEC) (NS) (APC)	1	RL LNG	(6)	5512	3	--	?
TAHR	35	Reverse Buffer of TAH with last element = 1.	(see TCH)	ANLZ (SPEC)	ANLZ (SPEC) (NS) (APC)	-1	RL LNG	7	5510	3	ones	2
TEP	36	6 cond residual	--	ANLZ (THV5f)	ANLZ (GATH)	1	PL LNG	216	6062	3	--	?

MAP-300 Buffers

Usage key: 1 - Constant
2 - Variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Increment	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
IFAC	37	Factors for Noise Shaping Analysis	--	BBN161	ANLZ (NS)	1	RL LNG	6	6494	3	.684128 .468032 .320194 .219054 .149861 .102524	1
TANS	38	Noise shaping filter Coeffs	(see TCH)	ANLZ (NS)	ANLZ (APC)	1	RL LNG	6	5530	3	--	2
TGFAC	39	Quantizer Scale factors alternating with their reciprocals	TGFAC: GFAC(1), 1/GFAC(1) GFAC(2), 1/GFAC(2) GFAC(3), 1/GFAC(3)	ANLZ (GAIN)	ANLZ (APC)	1	RL LNG	6	6506	3	--	2
TIMG	40	Coded Delta Gains	--	ANLZ (GAIN)	ANLZ (GTHR)	1	FXD LNG	3	6518	3	--	2
TUD	41	Unquantized, normalized Residual (for debugging only)	--	ANLZ (APC)	SYNZ (SPECF)	1	RL LNG	216	1394	2	--	2
TUJ	42	History for Noise-Shaping Filter	--	BBN161 ANLZ (APC)	ANLZ (APC)	1	RL LNG	6	6522	3	zeros	2
TUH	43	History for Spectral Filter	--	BBN161 ANLZ (APC)	ANLZ (APC)	1	RL LNG	6	6534	3	zeros	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TRIP	44	APC Pitch Filter History (Current frame)		ANLZ (APC)	ANLZ (APC)	1	RL LMG	(216)	6978	3	--	2
TRI	45	APC Pitch Filter History (last frame, current frame)	(see TRIP)	BBR161 ANLZ (APC) (TH151)	ANLZ (APC)	1	RL LMG	432	6546	3	zeros	2
TSNA (T10A)	46	Coded transmission data for ANLZA: T10A, T16, T10G, T1M, T1C, T1K	--	ANLZ (APC) (GTHR)	PROPAR, PRORES	1	FXD LMG	230	41344	1	zeros	2
TSNB (T10B)	47	Coded transmission data for ANLZB: T10B, T16, ...	--	ANLZ (APC) (GTHR)	PROPAR, PRORES	1	FXD LMG	230	41574	1	zeros	2
TBTA	--	TBTS A Buffer	--	PROPAR, PRORES	TMINT	1	FXD LMG	522	41804	1	--	2
TBTB	--	TBTS B Buffer	--	PROPAR, PRORES	TMINT	1	FXD LMG	522	42326	1	--	2
TBT	--	TBTS C Buffer	--	BBR161	TMINT	1	FXD LMG	522	42848	1	Bitstream equivalent of "Silence" (coded energy 0)	1

MAP-300 Buffers

Usage key: 1 - Constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
IMDMA	--	MODEM A Buffer	--	TMINT	MODEM Scroll Pgm	1	FXD LRG	522	43370	1	Bitstream equivalent of "Silence" (coded energy - 0) sync bit - 0	2
IMDMB	--	MODEM B Buffer	--	TMINT	MODEM Scroll Pgm	1	FXD LRG	522	43892	1	Bitstream equivalent of "Silence" (coded energy - 0) sync bit - 1	2
RMDMA	--	MODEM A Buffer	--	Modem Scroll Pgm	RMINT	1	FXD LRG	522	44414	1	--	2
RMDMB	--	MODEM B Buffer	--	Modem Scroll Pgm	RMINT	1	FXD LRG	522	44936	1	--	2
RMDMC	--	MODEM C Buffer	--	Modem Scroll Pgm	RMINT	1	FXD LRG	522	45458	1	--	2
MDMA	--	DATA A Buffer	--	RMINT	DECODE/B, CORPAR	1	FXD LRG	522	45980	1	--	2
MDMB	--	DATA B Buffer	--	RMINT	DECODE/B, CORPAR	1	FXD LRG	522	46502	1	--	2

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	AID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RRA (RRAA)	48	Received decoded transmission data: RRAA, RRA, RRAH, RRAH, RRAH, RRAH	--	DECODEA/B, CORPAR	SYNZ (SPECF) (DEAL)	1	RL LING	230	47024	1	zeros	2
RRAH (RRAH)	49	Received decoded transmission data: RRAH, RRAH, RRAH, RRAH, RRAH, RRAH	--	DECODEA/B, CORPAR	SYNZ (SPECF) (DEAL)	1	RL LING	230	47484	1	zeros	2
RRAH	50	Received Decoded $\sqrt{\text{Delta Gains}}$	--	SYNZ (DEAL)	SYNZ (SPECF)	1	RL LING	3	7410	3	--	2
RRI	51	Received Decoded Pitch Predictor Coefficients	--	SYNZ (DEAL)	SYNZ (PITF)	1	RL LING	3	7416	3	--	2
RRI	52	Received Decoded Reflection Coefficients	--	SYNZ (DEAL)	SYNZ (SPECF)	1	RL LING	8	7422	3	zeros	2
RRI	53	Scaled Residual Samples	--	SYNZ (SPECF)	SYNZ (SPECF)	1	RL LING	216	1826	2	--	2
RRI	54	Spectral Synthesis Filter Memory	--	SYNZ (SPECF)	SYNZ (SPECF)	1	RL LING	8	7438	3	zeros	2

MAP-300 Buffers

Usage key: 1 = Constant
2 = Variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RPH	55	Output of Spectral Synthesis Filter	--	SYN/ (SPCTF)	SYN/ (PTF)	1	RL LRG	216	7454	3	--	2
RPH0	56	Output of Pitch Synthesis Filter (current frame)		SYN/ (PTF)	SYN/ (DEMP) (RHTST)	1	RL LRG	(216)	2690	2	--	2
RPH	57	Output of Pitch Synthesis Filter (last frame and current filter)	(see RPH0)	SYN/ (RHTST) (PTF)	SYN/ (PTF)	1	RL LRG	432	2258	2	zeros	2
RPHA (RPHAL)	58	Preemphasized Synthetic Speech	--	SYN/ (DEMP)	DAHT	1	RL SHRT	216	47944	1	--	2
RPHB (RPHBL)	59	Decompasized Synthetic Speech	--	SYN/ (DEMP)	DAHT	1	RL SHRT	216	48160	1	--	2
RPHC	--	Synthetic "silence" data	--	RHTST	DAHT	1	RL SHRT	216	48376	1	zeros	1
RPHDA	--	D/A Output to ADM	--	DAHT	ADM Suball Program	1	RL SHRT	216	48592	1	zeros	2

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
MDAIB	--	D/A Output to AUM	--	DAINT	AUM Scroll Program	1	RL SHRT	216	48808	1	zeros	2

APPENDIX C
MAP-300 SCALARS

The tables contained in this Appendix describe the characteristics of each user-defined real and integer scalar referenced in the MAP-300 speech coder implementation. The table entry labeled "SID" or "ISID" indicates the real or integer scalar identification number. The entries labeled "Written by:" and "Read by:" indicate subroutines, function lists, or program modules that so access each scalar. (The A/D, D/A, transmitter modem, and receiver modem interrupt service routines are specified in these tables by "ADINT", "DAINT", "TMINT", and "RMINT", respectively.)

MAP-300 REAL SCALARS
Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TALPH	50	Premphasis filter coefficient	-	BBN161	ANLZ (PREMP)	0.4	1
TPHST	51	Premphasis filter history	-	ANLZ (PREMP)	ANLZ (PREMP)	0	2
TDCN	52	Negative of current frame's dc term	-	ANLZ (PITCH)	ANLZ (PITCH)	-	2
TPSZ1	53	1 - (Size of Pitch Extraction Buffer)	-	BBN161	ANLZ (PITCH)	1.0 - 265.	1
TMH	54	Pitch lag (in samples)	-	ANLZ (PITCH)	ANLZ (TRVFF) (AFC)	-	2
TIM	55	Coded pitch lag (integer in left inward)	-	ANLZ (PITCH)	ANLZ (GTER)	-	2
PI4	56	Constant 14	-	BBN161	ANLZ (PITCH)	14.	1

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TKTHR	57	Threshold value for MEWL and MWLQ6	-	BBN161	ANLZ (PITCH) (SPEC)	.9995	1
TIG	58	Cooled Frame Energy (Integer in left word)	-	ANLZ (GAIN)	ANLZ (GTHR)	-	2
TC2P	59	Pitch Predictor Coefficient for Single Tap Case	-	ANLZ (PITCH)	ANLZ (PITCH)	-	2
T72R	60	Constant: Reciprocal of 72	-	BBN161	ANLZ (APC)	$1 \div 72$	1
T3R	61	Constant: Reciprocal of 3	-	BBN161	ANLZ (APC)	$1 \div 3$	1
Rgll	65	Received Frame Energy	-	SYNZ (DEAL)	SYNZ (SPEC)	-	2
RMI	66	Received Pitch	-	SYNZ (DEAL)	SYNZ (PITCH)	-	2

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RRCF0	67	Deemphasis filter coefficient SA0	Must be consecutive ↕	BBN161	SYNZ (DEMP)	0.	1
RRCF1	68	Deemphasis filter coefficient SA1	Must be consecutive ↕	BBN161	SYNZ (DEMP)	0.	1
RRCF2	69	Deemphasis filter coefficient SA2	Must be consecutive ↕	BBN161	SYNZ (DEMP)	- ALPHA = -0.4	1
RRCF3	70	Deemphasis filter coefficient SA3	Must be consecutive ↕	BBN161	SYNZ (DEMP)	0.	1
RMY0	71	Deemphasis filter memory Y(-2)	Must be consecutive ↕	SYNZ (DEMP)	SYNZ (DEMP)	0.	2
RMY1	72	Deemphasis filter memory Y(-1)	Must be consecutive ↕	SYNZ (DEMP)	SYNZ (DEMP)	0.	2
RMY2	73	Deemphasis filter memory U(-2)	Must be consecutive ↕	SYNZ (DEMP)	SYNZ (DEMP)	0.	2

MAP-300 REAL SCALARS Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RMY1	74	Decomphasis filter memory 0(-1)	↑ Must be consecutive	SYN2 (DEMP)	SYN2 (DEMP)	0.	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TSRFA	50	TSRA Buffer Status Flag (0 => Empty)	-	ANLZ, ADINT	BBRLP, ADINT	0	2
TSRFB	51	TSRB Buffer Status Flag (0 => Empty)	-	ANLZ, ADINT	BBRLP, ADINT	0	2
TBTF A	52	TBTA Buffer Status Flag (0 => Empty)	-	ANLZ, TMINT	BBNLP, TMINT	0	2
TBTF B	53	TBTB Buffer Status Flag (0 => Empty)	-	ANLZ, TMINT	BBNLP, TMINT	0	2
TTIC	54	MMI output: Number of iterations not completed	Must be consecutive ↕	ANLZ (PITCH)	-	-	2
TEFAG	55	MMI output: Fractional energy E(R) / R(0)	Must be consecutive	ANLZ (PITCH)	-	-	2
-	56	(Unused)	-	-	-	-	-

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RRTFA	57	RRTFA Buffer Status Flag (0 -> Empty)	-	SYNZ, RMINT	BRNLP, RMINT	0	2
RRTFB	58	RRTFB Buffer Status Flag (0 -> Empty)	-	SYNZ, RMINT	BRNLP, RMINT	0	2
RSRFA	59	RSRFA Buffer Status Flag (0 -> Empty)	-	SYNZ, DAINT	BRNLP, DAINT	0	2
RSRFB	60	RSRFB Buffer Status Flag (0 -> Empty)	-	SYNZ, DAINT	BRNLP, DAINT	0	2
RUR	61	System Run Flag (0 -> Stop)	Must be consecutive ↕	BBRTS, BBRTLE	BBRTS	0	2
-	62	(left unused for MPTM)	Must be consecutive	-	-	-	-
-	63	(Unused)	-	-	-	-	-

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
ADPO	64	TSRA/B Buffer Pointer (0->B; -2->A)	-	ADINT	ADINT	0	2
TSRPO	65	TSRA/B Buffer Pointer (0->B; -2->A)	-	ADINT	ADINT	-2	2
TBPO	66	TBPA/B Buffer Pointer (0->B; -2->A)	-	TMINT	TMINT	0	2
TMPO	67	TMMA/B Buffer Pointer (0->B; -2->A)	-	TMINT	TMINT	0	2
RMPO	68	RMMA/B/C Buffer Pointer (0->C; -2->B; -4->A)	-	RMINT	RMINT	0	2
RBPO	69	RBMA/B Buffer Pointer (0->B; -2->A)	-	RMINT	RMINT	-2	2
BRPO	70	BRMA/B Buffer Pointer (0->B; -2->A)	-	DAINT	DAINT	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
DAI0	71	DATA/B Buffer Pointer (0->B; -2->A)	-	DAINT	DAINT	0	2
TAID0	72	A/D Frame Discard Counter	-	ADINT	BBN16E	0	2
TNIFC	73	TIMEOUT Frame Counter	-	TNINT	BBN16E	0	2
RMFIC	74	RMODEM Frame Discard Counter	-	RMINT	BBN16E	0	2
BSRR	75	BSINK Data- Not-Ready Counter	-	DAINT	BBN16E	0	2
TTCTR	76	Transmitter Frame Counter	-	ADINT	BBN16E	0	2
RCCTR	77	Receiver Frame Counter	-	RMINT	BBN16E	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RLCTR	78	Receiver Lost-Sync Counter	-	RMINT	BBN16E	0	2
RORR	79	Local Handset On-Block State (0->Handset Off->Block)	-	RMINT	ADINT, BBN16E	0	2
RSTRE	80	State of Receiver Sync 0->Lost Sync -1->Searching Sync +1->Found Sync	-	RMINT	RMINT BBN16E	0	2
RROFF	81	Beginning-of-Frame Offset (Sync bit position in RMODEM buffer)	Must be consecutive ↕ Must be consecutive	RMINT	RMINT BBN16E	-	2
-	82	Left unused for R-CTM	-				
ALCTR	83	A/D (ADAM) Interrupt Counter	-	Device 23, line 1 interrupt service routine	BBN16E	0	2
TMRCTR	84	TMODEM (TOS-2) Interrupt Counter	-	Device 16, line 1 interrupt service routine	BBN16E	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISIO	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RINTPR	85	PRINTER (IOS-2) Interrupt Counter	-	Device 16 line 2 interrupt service routine	BBN16E	0	2
DSACTW	86	D.A. (AOM) Interrupt Counter	-	Device 22 line 1 interrupt service routine	BBN16E	0	2
TADPEK	87	A/D Peak Input Level	-	ADPEAK, BBN16E	BBN16E	0	2
-	88- 122	(Unused)	-	-	-	-	-
RRCOR	123	Error-Correction switch (0 > Error Correction)	Must be consecutive ↑	BBN16E	COMPAR	0	2
-	124	(Left unused for MPTM)	Must be consecutive ↑	-	-	-	-
ERRG	125	Set to non-zero to cause channel error stimulation	Must be consecutive ↑	BBN16E	RMINT	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
-	126	(left unused for MPTM)	↓ Must be consecutive	-	-	-	-
-	127	(unused)	-	-	-	-	-

APPENDIX D
PROGRAM LISTINGS

This Appendix contains listings of the BBN-written (and modified) MAP-300 and PDP-11 (FORTRAN) programs referred to in Sections 2.4 and 3.

	Page
BBN16M.MLI	139
BBN16P.MLI	299
BBN16T.MLI	305
BBN16U.MLI	318
BBN16C.FOR	374
BBN16D.FOR	385
BBN16E.FOR	387
BBN16F.FOR	395
BBN16H.FOR	404
BBN16I.FOR	410
BBN16R.FOR	415
HIST16.FOR	417

T A B L E O F C O N T E N T S

ARRAY FUNCTION DISPATCH TABLE PATCHES	PAGE 4
NON-ARRAY FUNCTION DISPATCH TABLE PATCHES	PAGE 6
SYMBOL DEFINITIONS FOR BBN16 CODING/DECODING TABLE ADDRESSES	PAGE 7
APU, APS, AND CSPD CODE FOR ADDED FUNCTIONS	PAGE 8
APUJ-VLTSY VECTOR LATTICE SYNTHESIS FILTER	PAGE 9
APS3-V328A APS PROGRAM FOR VLTSY	PAGE 13
APU3-VKTOA(V,U) CONVERT REFL*N (K) TO LPC (A) COEFFICIENTS	PAGE 16
APS3-VKTOA(V,U) APS PROGRAM FOR VKTOA	PAGE 18
APU3-PITCH(V,A,U,B,C,D) COMPUTE AND CODE PITCH	PAGE 20
APS3-PITCH(V,A,U,B,C,D)	PAGE 25
MWLQ6(V,A,U,V,M) WEINER-LEVINSON MATRIX SOL'N & P=6 CODING/QUANT.	PAGE 29
MWLQ6-APS PROGRAM	PAGE 34
MWLQSSM - SPECIAL INTERMEDIATE SUPPORT FOR MWLQ6	PAGE 39
MWL(V,I,U,R,V,M)	PAGE 40
MEWL - APS PROGRAM	PAGE 42
SMCHMEWL - SPECIAL SUPPORT ROUTINE FOR MEWL	PAGE 47
APU3 - STAR(V,A,U,V)	PAGE 51
APS3 - STARS - 3-TAP STABILITY CHECK	PAGE 53
APU3-INVP(V,A,U,V) 3-TAP PITCH INVERSE FILTER	PAGE 56
APS3-INVP(V,A,U,V) APS PROGRAM FOR INVP	PAGE 58
APU3-ADPEAK(U) MONITOR PEAK ABS VAL OF INPUT SPEECH	PAGE 60
APS3-ADPEAK	PAGE 62
APUDMP(V) -- DEBUGGING FUNCTION THAT DUMPS THE APU REGISTERS	PAGE 64
APU3 - HFC(V,U,V)	PAGE 65
APS3 - HFC(V,U,V)	PAGE 67
APU3 - GAIN(V,A,U,B,V,C)	PAGE 70
APS3 - GAIN(V,A,U,B,V,C)	PAGE 72
SPECIAL SUPPORT MODULES FOR FEOL & FEOL	PAGE 79
APU3 - FFUSL FOURIER EVEN-ODD SEPARATE LONG (MAP-300)	PAGE 82
APS3 - FFIAL APS PROGRAM FOR FFUSL	PAGE 83
APU3 - APC(V,A,U,V,M,R,S,T) APC (WITH NOISE SHAPING)	PAGE 86
APS3 - APC(V,A,U,V,M,R,S,T) APC (WITH NOISE SHAPING)	PAGE 89
SMJAPC SPECIAL BINDING MODULE FOR APC	PAGE 100
APU3 - GTHR(V,A,U,B,V,W)	PAGE 107
APS3 - GTHR(V,A,U,B,V,W)	PAGE 113
APU3 - THIST(V,U,V,M,R,S) SET UP XMITTER FRAME HISTORIES	PAGE 114
APS3 - THIST(V,U,V,M,R,S) APS PROGRAM FOR THIST	PAGE 117
SUNSTH SPECIAL BINDING MODULE FOR THIST	PAGE 118
APU3 - SCIPES(V,A,U,V)	PAGE 120
APS3 - SCIPES(V,A,U,V)	PAGE 123
APU3-PITSYN(V,A,U,V) 3-TAP PITCH SYNTHESIS FILTER	PAGE 127
APS3-PITSYN(V,A,U,V) APS PROGRAM FOR PITSYN	PAGE 129
APU3 - DEAI(V,A,U,B,V,W)	PAGE 131
APS3 - DEAI(V,A,U,B,V,W)	PAGE 132
APU3-DCORZ DISCRETE CORRELATION APU PROGRAM	PAGE 134
APS3-DCORZ APS PROGRAM FOR DISCRETE CORRELATION WITH AUTO ZERO FILL	PAGE 138
SBDCORZ - SPECIAL BINDING MODULE FOR DCORZ	PAGE 141
MPIFFS MODULE TO PROCESS THE MPIFF(ISA,ISB,FLID) FCB	PAGE 144
MPGSC -- G-FLAG SET/CLEAR	PAGE 145
DEBUGGING "BREAKPOINT" FACILITY	PAGE 146
TOP OF EXECUTIVE DEFINITION	PAGE 147

(000001) {[BBNDI<DCA16>BBN16N.MSU.2, 29-Dec-80 14:30:40, Pd: WOLF

DEFINE 'SAP' (SET ADDRESS FIELD) AND 'LAP' (LOAD ADDRESS FIELD) INSTRUCTIONS TO ASSEMBLER. THESE INSTRUCTIONS ARE NEW CSU INSTRUCTIONS AVAILABLE WITH 'REV 18' CSU MICROCODE. THE SETLOAD THE LOW ORDER 17 BITS OF THE REFERENCED FULL-WORD FROM INTO THE INDICATED REGISTER.

OPADD SAF,(1 .LS. 14) + (29 .LS. 8) + \$ZF
OPADD LAF,(1 .LS. 14) + (29 .LS. 8) + \$ZF

DEFINE SYMBOLS FOR ARRAY FUNCTION ASSEMBLIES

FROM THE SNAP-II EXECUTIVE --- REL. 3.05 --- 5/22/79

AFDT\$ORC = \$8E8

AP\$ASSS = \$245

AP\$RNDR = \$EFA
AP\$RNDR = CF63

AP\$BNDRI = \$F63
AP\$BNDRI = \$F20

AP\$HSL = \$240

APSCSSC = \$248

AP\$DINE = \$FBI

AP\$DNFR = \$FBC
AP\$CB = CC

APSG1 = SD
APSG9 = SC

AP\$LLUC = \$11

AP\$PFF = \$10

AP\$SAID = \$A
ID\$CCU = CE

APSSS = \$9
APSSS = \$5

APSRM1 = \$1

APSLA = \$1FPCD

APSLIC = \$1FFC0

APSWM = \$1PFCB
APSW = \$1PFCB

071115 = NSJY

BCT\$AD = \$6W4

ACT\$AT = \$686

8CT\$BA = \$582
DT\$CB = 69

BITSRC = \$M
BITSCD = \$M

RTSWS = \$3

1

CK40B\$ = \$1A06

CLRSCT = \$792
CNS = \$7194

DATE = 5/11/77

100

CSPUSWDS = \$21FC	(00054)	*
DWY\$ = \$794	(00056)	*
ERRUN\$ = \$1AFA	(00057)	*
FIAS\$ = \$3110	(00058)	*
FDT\$ = \$7E0	(00059)	*
FEOS\$ = \$3040	(00060)	*
FFTS0SZ\$ = \$79A	(00061)	*
FLG\$CLR = \$0	(00062)	*
FLG\$CU = \$4	(00063)	*
FLG\$CI = \$5	(00064)	*
FLG\$C2 = \$6	(00065)	*
FLG\$G3 = \$7	(00066)	*
FLG\$RI = \$11	(00067)	*
FLG\$SET = \$20	(00068)	*
GATHEN\$ = \$1B3C	(00069)	*
HS\$ = 1	(00070)	*
ISAT\$ = \$502	(00071)	*
LOADSAP = \$1B7E	(00072)	*
LOADSAP1 = \$1B8F	(00073)	*
MSK\$DB8 = \$6	(00074)	*
MSK\$DBYT = \$FF00	(00075)	*
MSK\$DBYT = \$00FF	(00076)	*
M\$ = 0	(00077)	*
OQE = \$1B	(00078)	*
S0111\$ = \$35EA	(00079)	*
SHTL\$R5 = \$7AE	(00080)	*
SINCDS\$ = \$23FA	(00081)	*
SIN\$ = \$3192	(00082)	*
SVT\$ = \$302	(00083)	*
SVT\$UNI = \$3CE	(00084)	*
SVS\$PLGS = \$1FFCE	(00085)	*
TEM\$0 = \$704	(00086)	*
TUE\$ = \$21FE	(00087)	*
TUE\$PTR = \$208	(00088)	*
VALBWS\$ = \$1CSA	(00089)	*
VSMAZ\$ = \$2070	(00090)	*
W\$ = \$2	(00091)	*
XFL\$01 = \$192C	(00092)	*

PAGE 3: [RUNDJ] <DCAL16> DBN16M.MSD.2, 29-Dec-88 14:38:48, Ed: WOLF

```

000007BA (00107) *
          (00108) *
          (00109) *
          (00110) *
0000028B (00111) BL = T0ESPTR
          (00112) *
0028B 001073C0 (00113) *
          (00114) *
          (00115) *
          (00116) *
          (00117) *
          (00118) *
          (00119) *

          ZERO = $7BA
          ADDR T0ESCUR(,1)
          RM = 3
          UPDATE TOP OF EXEC POINTER

```

(00120) "ARRAY FUNCTION DISPATCH TABLE PATCHES		
(00121) *		
(00122) *		
(00123) *		
00000900 (00124)	FL = AFDT\$ORC+3*W\$*(132-120)	JFCB 132 (VLTSV)
00000900 (00125)	ADDR VLTSV\$(R7,1)	
00000900 (00126)	ADDR V320W\$(R7,1)	
00000900 (00127)	ADDR CSPUSNOS(,1,0)	
00000900 (00128)		
00000906 (00129)	FL = AFDT\$ORC+3*W\$*(133-120)	JFCB 133 (VKTOA)
00000906 (00130)	ADDR VKTOA\$(R7,1)	
00000906 (00131)	ADDR VKASI(R7,1)	
00000906 (00132)	ADDR CSPUSNOS(,1,0)	
00000906 (00133)		
0000090C (00134)	FL = AFDT\$ORC+3*W\$*(134-120)	JFCB 134 (PITCH)
0000090C (00135)	ADDR PITCH\$(R7,1)	
0000090C (00136)	ADDR PITCH\$(R7,1)	
0000090C (00137)	ADDR CSPUSNOS(,1,0)	
0000090C (00138)		
00000912 (00139)	FL = AFDT\$ORC+3*W\$*(135-120)	JFCB 135 (HML06)
00000912 (00140)	ADDR HMLF\$APU(R7,1)	
00000912 (00141)	ADDR HMLF\$APS(R7,1)	
00000912 (00142)	ADDR HMLQ\$SSM(,1,1)	
00000912 (00143)		
0000091C (00144)	FL = AFDT\$ORC+3*W\$*(142-120)	JFCB 142 (HML)
0000091C (00145)	ADDR HMLW\$SS(R7,1)	
0000091C (00146)	ADDR HMLW\$(R7,1)	
0000091C (00147)	ADDR SOW\$HML(,1,0)	
0000091C (00148)		
0000096C (00149)	FL = AFDT\$ORC+3*W\$*(150-120)	JFCB 150 (STAR)
0000096C (00150)	ADDR STAB\$(R7,1)	
0000096C (00151)	ADDR STAB\$(R7,1)	
0000096C (00152)	ADDR CSPUSNOS(,1,0)	
0000096C (00153)		
000009D2 (00154)	FL = AFDT\$ORC+3*W\$*(167-120)	JFCB 167 (INWPF)
000009D2 (00155)	ADDR INWPF\$(R7,1)	
000009D2 (00156)	ADDR INWPSI(R7,1)	
000009D2 (00157)	ADDR CSPUSNOS(,1,0)	
000009D2 (00158)		
000009D2C (00159)	FL = AFDT\$ORC+3*W\$*(182-120)	JFCB 182 (ADPEAK)
000009D2C (00160)	ADDR ADPEAK(R7,1)	
000009D2C (00161)	ADDR ADPEAK(R7,1)	
000009D2C (00162)	ADDR CSPUSNOS(,1,0)	
000009D2C (00163)		
000009D32 (00164)	FL = AFDT\$ORC+3*W\$*(183-120)	JFCB 183 (APUDHP)
000009D32 (00165)	ADDR APUDHP\$(R7,1)	
000009D32 (00166)	ADDR DUMPS(R7,1)	
000009D32 (00167)	ADDR CSPUSNOS(,1,0)	
000009D32 (00168)		
000009D5C (00169)	FL = AFDT\$ORC+3*W\$*(190-120)	JFCB 190 (HFC)
000009D5C (00170)	ADDR HFC\$(R7,1)	
000009D5C (00171)	ADDR HFC\$(R7,1)	
000009D5C (00172)	ADDR CSPUSNOS(,1,0)	

000000A00 (00173) *	HL = AFDTSORG+3*W5*(196-128)	JFCB 196 (GAIN)
000000A00 (00174)	ADDR GAINS(R7,1)	
000000A00 (00175)	ADDR GAINS(R7,1)	
000000A00 (00176)	ADDR CSPUSNUS(,1,0)	
000000A00 (00177)	ADDR CSPUSNUS(,1,0)	
000000A00 (00178) *	HL = AFDTSORG+3*W5*(197-128)	JFCB 197 (FEOPFL)
000000A00 (00179)	ADDR FEOPFL(R7,1)	
000000A00 (00180)	ADDR FEOPFL(R7,1)	
000000A00 (00181)	ADDR FEOPFL(R7,1)	
000000A00 (00182)	ADDR FEOPFL(R7,1)	
000000A00 (00183) *	HL = AFDTSORG+3*W5*(198-128)	JFCB 198 (FEOPIL)
000000A00 (00184)	ADDR FEOPIL(R7,1)	
000000A00 (00185)	ADDR FEOPIL(R7,1)	
000000A00 (00186)	ADDR FEOPIL(R7,1)	
000000A00 (00187)	ADDR FEOPIL(R7,1)	
000000A00 (00188) *	HL = AFDTSORG+3*W5*(199-128)	JFCB 199 (APC)
000000A00 (00189)	ADDR APC(R7,1)	
000000A00 (00190)	ADDR APC(R7,1)	
000000A00 (00191)	ADDR APC(R7,1)	
000000A00 (00192)	ADDR APC(R7,1)	
000000A00 (00193) *	HL = AFDTSORG+3*W5*(200-128)	JFCB 200 (GTHR)
000000A00 (00194)	ADDR GTHR(R7,1)	
000000A00 (00195)	ADDR GTHR(R7,1)	
000000A00 (00196)	ADDR GTHR(R7,1)	
000000A00 (00197)	ADDR GTHR(R7,1)	
000000A00 (00198) *	HL = AFDTSORG+3*W5*(201-128)	JFCB 201 (THIST)
000000A00 (00199)	ADDR THIST(R7,1)	
000000A00 (00200)	ADDR THIST(R7,1)	
000000A00 (00201)	ADDR THIST(R7,1)	
000000A00 (00202)	ADDR THIST(R7,1)	
000000A00 (00203) *	HL = AFDTSORG+3*W5*(202-128)	JFCB 202 (SCLRES)
000000A00 (00204)	ADDR SCLRES(R7,1)	
000000A00 (00205)	ADDR SCLRES(R7,1)	
000000A00 (00206)	ADDR SCLRES(R7,1)	
000000A00 (00207)	ADDR SCLRES(R7,1)	
000000A00 (00208) *	HL = AFDTSORG+3*W5*(203-128)	JFCB 203 (PITSYN)
000000A00 (00209)	ADDR PITSYN(R7,1)	
000000A00 (00210)	ADDR PITSYN(R7,1)	
000000A00 (00211)	ADDR PITSYN(R7,1)	
000000A00 (00212)	ADDR PITSYN(R7,1)	
000000A00 (00213) *	HL = AFDTSORG+3*W5*(212-128)	JFCB 212 (DEAL)
000000A00 (00214)	ADDR DEAL(R7,1)	
000000A00 (00215)	ADDR DEAL(R7,1)	
000000A00 (00216)	ADDR DEAL(R7,1)	
000000A00 (00217)	ADDR DEAL(R7,1)	
000000A00 (00218) *	HL = AFDTSORG+3*W5*(213-128)	JFCB 213 (DCURZ)
000000A00 (00219)	ADDR DCURZ(R7,1)	
000000A00 (00220)	ADDR DCURZ(R7,1)	
000000A00 (00221)	ADDR DCURZ(R7,1)	
000000A00 (00222)	ADDR DCURZ(R7,1)	
000000A00 (00223) *	HL = AFDTSORG+3*W5*(214-128)	JFCB 214 (DCURZ)
000000A00 (00224)	ADDR DCURZ(R7,1)	
000000A00 (00225)	ADDR DCURZ(R7,1)	
000000A00 (00226)	ADDR DCURZ(R7,1)	
000000A00 (00227)	ADDR DCURZ(R7,1)	
000000A00 (00228)	ADDR DCURZ(R7,1)	
000000A00 (00229)	ADDR DCURZ(R7,1)	
000000A00 (00230)	ADDR DCURZ(R7,1)	
000000A00 (00231)	ADDR DCURZ(R7,1)	
000000A00 (00232)	ADDR DCURZ(R7,1)	
000000A00 (00233)	ADDR DCURZ(R7,1)	

```

PAGE 6: [BBND]DCAL16>DBW16M.NSU.2, 29-Dec-88 14:38:48, Ed: VMF
NON-ARRAY FUNCTION DISPATCH TABLE PATCHES

(00224) *NON-ARRAY FUNCTION DISPATCH TABLE PATCHES
(00225) *
(00226) *
(00227) *
00000001 (00228) JL = FDT$ + (WS * 105) JFCB 105 (MPIP)
00000002 (00229) ADDR MPIP$
00000003 (00230) *
00000004 (00231) JL = FDT$ + (WS * 106) JFCB 106 (MPGSC)
00000005 (00232) ADDR MPGSC
00000006 (00233) *

```


7: (BMD)(<DCAL)(>BBM16M-MSU.2, 29-Dec-88 14:30:40, Ed: WULF
SYNRM DEFINITIONS FOR BBM16 CODING/DECODING TABLE ADDRESSES
SYNRM DEFINITIONS FOR BBM16 CODING/DECODING TABLE ADDRESSES
(00234) *SYNRM DEFINITIONS FOR BBM16 CODING/DECODING TABLE ADDRESSES
(00235))
(00236)) TABLES RESIDE IN BUS1, STARTING AT \$9800.
(00237)) TABLE CONTENTS ARE DEFINED IN MODULE "BBM16T".
(00238))

CODING TABLES

00009000 (00239) CTHC1=\$9800
00009010 (00240) CTHC2=CTHC1+ 8*W\$
00009020 (00241) CTHC3=CTHC1
00009030 (00242) CTHK1=CTHC2+16*W\$
00009040 (00243) CTHK2=CTHC1+64*W\$
00009050 (00244) CTHK3=CTHC2+32*W\$
00009060 (00245) CTHK4=CTHC3+16*W\$
00009070 (00246) CTHK5=CTHC4+16*W\$
00009080 (00247) CTHK6=CTHC5+16*W\$
00009090 (00248) CTHG =CTHC6+16*W\$
00009100 (00249) CTHDG=CTHC +64*W\$
00009110 (00250) CTHU =CTHDG+ 4*W\$
00009120 (00251) DVLC1=CTHU + 4*W\$
00009130 (00252) DVLC2=DVLC1+ 8*W\$
00009140 (00253) DVLC3=DVLC1
00009150 (00254) DVLK1=DVLC2+16*W\$
00009160 (00255) DVLK2=DVLC1+64*W\$
00009170 (00256) DVLK3=DVLC2+32*W\$
00009180 (00257) DVLK4=DVLC3+16*W\$
00009190 (00258) DVLK5=DVLC4+16*W\$
00009200 (00259) DVLK6=DVLC5+16*W\$
00009210 (00260) DVLC =DVLC6+16*W\$
00009220 (00261) DVLDG=DVLC +64*W\$
00009230 (00262) DVLU =DVLDG+ 4*W\$
00009240 (00263) DVLUF=DVLU + 4*W\$

DECODING TABLES

PAGE 8: (00NDJ00CA16>R0016M.N50.2, 29-Dec-80 14:30:40, Ed: WOLF
APU, APS, AND CSPU CODE FOR ADDED FUNCTIONS

(00264) *APU, APS, AND CSPU CODE FOR ADDED FUNCTIONS
(00265) *
(00266) * (FUNCTIONS ARE ORDERED BY PCN NUMBER, WITH
(00267) * ARRAY FUNCTIONS FIRST, THEN NON-ARRAY FUNCTIONS.)
(00268) *
(00269) *
(00270) * SET START ADDRESS FOR ADDED FUNCTIONS.
(00271) * REL 3.5 LEAVES EMPTY SPACE (ON BUS 1) AT:
(00272) * \$3000 - \$4000
(00273) * \$4000 - \$4A00
(00274) * \$5000 - \$C000
(00275) * PUT ADDED FUNCTIONS IN THIS SPACE.
(00276) *
00005D00 (00277) * BL = \$5000
00005D00 (00278) *

```

PAGE          9:  [RRND]COCAL>NNH16M.N50.2, 29-Dec-80 14:30:40, Fd: WULF
                VECTOR LATTICE SYNTHESIS FILTER
                APJ-VLTSY      VECTOR LATTICE SYNTHESIS FILTER
                                CUCED BY LCUSELL, 3/79
                                BINDS TO APS3-W3200
                                PERFORMS LATTICE SYNTHESIS FILTER, USING REFL*N COEFFS
                                IMPLEMENTS THE FOLLOWING FORTRAN CODE:

                                DU 20 I=1,N
                                  F(7)=W(I)-G(7)*K(8)
                                  DO 10 J=6,0,-1
                                    F(J)=F(J+1)-G(J)*K(J+1)
                                  G(J+1)=G(J)+F(J)*K(J+1)
                                  G(8)=F(8)
                                  V(I)=F(8)
                                *
                                * THERE ARE THREE INPUT VECTORS:
                                K (LENGTH 8) REFLECTION COEFFICIENTS (V BID)
                                K (LENGTH 8) FILTER MEMORY (U BID)
                                W (LENGTH N) INPUT RESIDUAL SAMPLES (W BID)
                                *THERE ARE TWO OUTPUT ARRAYS:
                                G AS BEFORE
                                Y OUTPUT SYNTHETIC SPEECH SAMPLES (LENGTH N) (Y BID)
                                *
                                * INPUT STREAM:
                                G(7)
                                G(6)
                                ...
                                G(8)
                                K(8)
                                K(7)
                                ...
                                K(1)
                                W(1)
                                W(2)
                                ...
                                W(N).
                                *
                                * OUTPUT STREAM:
                                Y(1)
                                Y(2)
                                ...
                                Y(N)
                                G(7)
                                G(6)
                                ...
                                G(8).
                                *
                                REGISTER USAGE:
                                (00279) *
                                (00280) *
                                (00281) *
                                (00282) *
                                (00283) *
                                (00284) *
                                (00285) *
                                (00286) *
                                (00287) *
                                (00288) *
                                (00289) *
                                (00290) *
                                (00291) *
                                (00292) *
                                (00293) *10
                                (00294) *
                                (00295) *20
                                (00296) *
                                (00297) *
                                (00298) *
                                (00299) *
                                (00300) *
                                (00301) *
                                (00302) *THERE ARE TWO OUTPUT ARRAYS:
                                (00303) *
                                (00304) *
                                (00305) *
                                (00306) *INPUT STREAM:
                                (00307) *
                                (00308) *
                                (00309) *
                                (00310) *
                                (00311) *
                                (00312) *
                                (00313) *
                                (00314) *
                                (00315) *
                                (00316) *
                                (00317) *
                                (00318) *
                                (00319) *
                                (00320) *OUTPUT STREAM:
                                (00321) *
                                (00322) *
                                (00323) *
                                (00324) *
                                (00325) *
                                (00326) *
                                (00327) *
                                (00328) *
                                (00329) *
                                (00330) *
                                (00331) *

```

```

(00332) *LEFT      RIGHT
(00333) *          M0=G0  M0=G0
(00334) *          M1=G1  M1=G1
(00335) *          M2=G2  M2=G2
(00336) *          M3=G3  M3=G3
(00337) *          M4=G4  M4=G4
(00338) *          M5=G5  M5=G5
(00339) *          M6=G6  M6=G6
(00340) *          M7=G7  M7=G7
(00341) *          M8=G8  M8=G8
(00342) *          M9=G9  M9=G9
(00343) *          M10=G10 M10=G10
(00344) *          M11=G11 M11=G11
(00345) *          M12=G12 M12=G12
(00346) *          M13=G13 M13=G13
(00347) *          M14=G14 M14=G14
(00348) *          M15=G15 M15=G15
(00349) *          M16=G16 M16=G16
(00350) *          M17=G17 M17=G17
(00351) *          M18=G18 M18=G18
(00352) *          M19=G19 M19=G19
(00353) *          M20=G20 M20=G20
(00354) *          M21=G21 M21=G21
(00355) *          M22=G22 M22=G22
(00356) *          M23=G23 M23=G23
(00357) *          M24=G24 M24=G24
(00358) *          M25=G25 M25=G25
(00359) *          M26=G26 M26=G26
(00360) *          M27=G27 M27=G27
(00361) *          M28=G28 M28=G28
(00362) *          M29=G29 M29=G29
(00363) *          M30=G30 M30=G30
(00364) *          M31=G31 M31=G31
(00365) *          M32=G32 M32=G32
(00366) *          M33=G33 M33=G33
(00367) *          M34=G34 M34=G34
(00368) *          M35=G35 M35=G35
(00369) *          M36=G36 M36=G36
(00370) *          M37=G37 M37=G37
(00371) *          M38=G38 M38=G38
(00372) *          M39=G39 M39=G39
(00373) *          M40=G40 M40=G40
(00374) *          M41=G41 M41=G41
(00375) *          M42=G42 M42=G42
(00376) *          M43=G43 M43=G43
(00377) *          M44=G44 M44=G44
(00378) *          M45=G45 M45=G45
(00379) *          M46=G46 M46=G46
(00380) *          M47=G47 M47=G47
(00381) *          M48=G48 M48=G48
(00382) *          M49=G49 M49=G49
(00383) *          M50=G50 M50=G50
(00384) *          M51=G51 M51=G51
(00385) *          M52=G52 M52=G52
(00386) *          M53=G53 M53=G53
(00387) *          M54=G54 M54=G54
(00388) *          M55=G55 M55=G55
(00389) *          M56=G56 M56=G56
(00390) *          M57=G57 M57=G57
(00391) *          M58=G58 M58=G58
(00392) *          M59=G59 M59=G59
(00393) *          M60=G60 M60=G60
(00394) *          M61=G61 M61=G61
(00395) *          M62=G62 M62=G62
(00396) *          M63=G63 M63=G63
(00397) *          M64=G64 M64=G64
(00398) *          M65=G65 M65=G65
(00399) *          M66=G66 M66=G66
(00400) *          M67=G67 M67=G67
(00401) *          M68=G68 M68=G68
(00402) *          M69=G69 M69=G69
(00403) *          M70=G70 M70=G70
(00404) *          M71=G71 M71=G71
(00405) *          M72=G72 M72=G72
(00406) *          M73=G73 M73=G73
(00407) *          M74=G74 M74=G74
(00408) *          M75=G75 M75=G75
(00409) *          M76=G76 M76=G76
(00410) *          M77=G77 M77=G77
(00411) *          M78=G78 M78=G78
(00412) *          M79=G79 M79=G79
(00413) *          M80=G80 M80=G80
(00414) *          M81=G81 M81=G81
(00415) *          M82=G82 M82=G82
(00416) *          M83=G83 M83=G83
(00417) *          M84=G84 M84=G84
(00418) *          M85=G85 M85=G85
(00419) *          M86=G86 M86=G86
(00420) *          M87=G87 M87=G87
(00421) *          M88=G88 M88=G88
(00422) *          M89=G89 M89=G89
(00423) *          M90=G90 M90=G90
(00424) *          M91=G91 M91=G91
(00425) *          M92=G92 M92=G92
(00426) *          M93=G93 M93=G93
(00427) *          M94=G94 M94=G94
(00428) *          M95=G95 M95=G95
(00429) *          M96=G96 M96=G96
(00430) *          M97=G97 M97=G97
(00431) *          M98=G98 M98=G98
(00432) *          M99=G99 M99=G99
(00433) *          M100=G100 M100=G100
(00434) *          M101=G101 M101=G101
(00435) *          M102=G102 M102=G102
(00436) *          M103=G103 M103=G103
(00437) *          M104=G104 M104=G104
(00438) *          M105=G105 M105=G105
(00439) *          M106=G106 M106=G106
(00440) *          M107=G107 M107=G107
(00441) *          M108=G108 M108=G108
(00442) *          M109=G109 M109=G109
(00443) *          M110=G110 M110=G110
(00444) *          M111=G111 M111=G111
(00445) *          M112=G112 M112=G112
(00446) *          M113=G113 M113=G113
(00447) *          M114=G114 M114=G114
(00448) *          M115=G115 M115=G115
(00449) *          M116=G116 M116=G116
(00450) *          M117=G117 M117=G117
(00451) *          M118=G118 M118=G118
(00452) *          M119=G119 M119=G119
(00453) *          M120=G120 M120=G120
(00454) *          M121=G121 M121=G121
(00455) *          M122=G122 M122=G122
(00456) *          M123=G123 M123=G123
(00457) *          M124=G124 M124=G124
(00458) *          M125=G125 M125=G125
(00459) *          M126=G126 M126=G126
(00460) *          M127=G127 M127=G127
(00461) *          M128=G128 M128=G128
(00462) *          M129=G129 M129=G129
(00463) *          M130=G130 M130=G130
(00464) *          M131=G131 M131=G131
(00465) *          M132=G132 M132=G132
(00466) *          M133=G133 M133=G133
(00467) *          M134=G134 M134=G134
(00468) *          M135=G135 M135=G135
(00469) *          M136=G136 M136=G136
(00470) *          M137=G137 M137=G137
(00471) *          M138=G138 M138=G138
(00472) *          M139=G139 M139=G139
(00473) *          M140=G140 M140=G140
(00474) *          M141=G141 M141=G141
(00475) *          M142=G142 M142=G142
(00476) *          M143=G143 M143=G143
(00477) *          M144=G144 M144=G144
(00478) *          M145=G145 M145=G145
(00479) *          M146=G146 M146=G146
(00480) *          M147=G147 M147=G147
(00481) *          M148=G148 M148=G148
(00482) *          M149=G149 M149=G149
(00483) *          M150=G150 M150=G150
(00484) *          M151=G151 M151=G151
(00485) *          M152=G152 M152=G152
(00486) *          M153=G153 M153=G153
(00487) *          M154=G154 M154=G154
(00488) *          M155=G155 M155=G155
(00489) *          M156=G156 M156=G156
(00490) *          M157=G157 M157=G157
(00491) *          M158=G158 M158=G158
(00492) *          M159=G159 M159=G159
(00493) *          M160=G160 M160=G160
(00494) *          M161=G161 M161=G161
(00495) *          M162=G162 M162=G162
(00496) *          M163=G163 M163=G163
(00497) *          M164=G164 M164=G164
(00498) *          M165=G165 M165=G165
(00499) *          M166=G166 M166=G166
(00500) *          M167=G167 M167=G167
(00501) *          M168=G168 M168=G168
(00502) *          M169=G169 M169=G169
(00503) *          M170=G170 M170=G170
(00504) *          M171=G171 M171=G171
(00505) *          M172=G172 M172=G172
(00506) *          M173=G173 M173=G173
(00507) *          M174=G174 M174=G174
(00508) *          M175=G175 M175=G175
(00509) *          M176=G176 M176=G176
(00510) *          M177=G177 M177=G177
(00511) *          M178=G178 M178=G178
(00512) *          M179=G179 M179=G179
(00513) *          M180=G180 M180=G180
(00514) *          M181=G181 M181=G181
(00515) *          M182=G182 M182=G182
(00516) *          M183=G183 M183=G183
(00517) *          M184=G184 M184=G184
(00518) *          M185=G185 M185=G185
(00519) *          M186=G186 M186=G186
(00520) *          M187=G187 M187=G187
(00521) *          M188=G188 M188=G188
(00522) *          M189=G189 M189=G189
(00523) *          M190=G190 M190=G190
(00524) *          M191=G191 M191=G191
(00525) *          M192=G192 M192=G192
(00526) *          M193=G193 M193=G193
(00527) *          M194=G194 M194=G194
(00528) *          M195=G195 M195=G195
(00529) *          M196=G196 M196=G196
(00530) *          M197=G197 M197=G197
(00531) *          M198=G198 M198=G198
(00532) *          M199=G199 M199=G199
(00533) *          M200=G200 M200=G200
(00534) *          M201=G201 M201=G201
(00535) *          M202=G202 M202=G202
(00536) *          M203=G203 M203=G203
(00537) *          M204=G204 M204=G204
(00538) *          M205=G205 M205=G205
(00539) *          M206=G206 M206=G206
(00540) *          M207=G207 M207=G207
(00541) *          M208=G208 M208=G208
(00542) *          M209=G209 M209=G209
(00543) *          M210=G210 M210=G210
(00544) *          M211=G211 M211=G211
(00545) *          M212=G212 M212=G212
(00546) *          M213=G213 M213=G213
(00547) *          M214=G214 M214=G214
(00548) *          M215=G215 M215=G215
(00549) *          M216=G216 M216=G216
(00550) *          M217=G217 M217=G217
(00551) *          M218=G218 M218=G218
(00552) *          M219=G219 M219=G219
(00553) *          M220=G220 M220=G220
(00554) *          M221=G221 M221=G221
(00555) *          M222=G222 M222=G222
(00556) *          M223=G223 M223=G223
(00557) *          M224=G224 M224=G224
(00558) *          M225=G225 M225=G225
(00559) *          M226=G226 M226=G226
(00560) *          M227=G227 M227=G227
(00561) *          M228=G228 M228=G228
(00562) *          M229=G229 M229=G229
(00563) *          M230=G230 M230=G230
(00564) *          M231=G231 M231=G231
(00565) *          M232=G232 M232=G232
(00566) *          M233=G233 M233=G233
(00567) *          M234=G234 M234=G234
(00568) *          M235=G235 M235=G235
(00569) *          M236=G236 M236=G236
(00570) *          M237=G237 M237=G237
(00571) *          M238=G238 M238=G238
(00572) *          M239=G239 M239=G239
(00573) *          M240=G240 M240=G240
(00574) *          M241=G241 M241=G241
(00575) *          M242=G242 M242=G242
(00576) *          M243=G243 M243=G243
(00577) *          M244=G244 M244=G244
(00578) *          M245=G245 M245=G245
(00579) *          M246=G246 M246=G246
(00580) *          M247=G247 M247=G247
(00581) *          M248=G248 M248=G248
(00582) *          M249=G249 M249=G249
(00583) *          M250=G250 M250=G250
(00584) *          M251=G251 M251=G251
(00585) *          M252=G252 M252=G252
(00586) *          M253=G253 M253=G253
(00587) *          M254=G254 M254=G254
(00588) *          M255=G255 M255=G255
(00589) *          M256=G256 M256=G256
(00590) *          M257=G257 M257=G257
(00591) *          M258=G258 M258=G258
(00592) *          M259=G259 M259=G259
(00593) *          M260=G260 M260=G260
(00594) *          M261=G261 M261=G261
(00595) *          M262=G262 M262=G262
(00596) *          M263=G263 M263=G263
(00597) *          M264=G264 M264=G264
(00598) *          M265=G265 M265=G265
(00599) *          M266=G266 M266=G266
(00600) *          M267=G267 M267=G267
(00601) *          M268=G268 M268=G268
(00602) *          M269=G269 M269=G269
(00603) *          M270=G270 M270=G270
(00604) *          M271=G271 M271=G271
(00605) *          M272=G272 M272=G272
(00606) *          M273=G273 M273=G273
(00607) *          M274=G274 M274=G274
(00608) *          M275=G275 M275=G275
(00609) *          M276=G276 M276=G276
(00610) *          M277=G277 M277=G277
(00611) *          M278=G278 M278=G278
(00612) *          M279=G279 M279=G279
(00613) *          M280=G280 M280=G280
(00614) *          M281=G281 M281=G281
(00615) *          M282=G282 M282=G282
(00616) *          M283=G283 M283=G283
(00617) *          M284=G284 M284=G284
(00618) *          M285=G285 M285=G285
(00619) *          M286=G286 M286=G286
(00620) *          M287=G287 M287=G287
(00621) *          M288=G288 M288=G288
(00622) *          M289=G289 M289=G289
(00623) *          M290=G290 M290=G290
(00624) *          M291=G291 M291=G291
(00625) *          M292=G292 M292=G292
(00626) *          M293=G293 M293=G293
(00627) *          M294=G294 M294=G294
(00628) *          M295=G295 M295=G295
(00629) *          M296=G296 M296=G296
(00630) *          M297=G297 M297=G297
(00631) *          M298=G298 M298=G298
(00632) *          M299=G299 M299=G299
(00633) *          M300=G300 M300=G300
(00634) *          M301=G301 M301=G301
(00635) *          M302=G302 M302=G302
(00636) *          M303=G303 M303=G303
(00637) *          M304=G304 M304=G304
(00638) *          M305=G305 M305=G305
(00639) *          M306=G306 M306=G306
(00640) *          M307=G307 M307=G307
(00641) *          M308=G308 M308=G308
(00642) *          M309=G309 M309=G309
(00643) *          M310=G310 M310=G310
(00644) *          M311=G311 M311=G311
(00645) *          M312=G312 M312=G312
(00646) *          M313=G313 M313=G313
(00647) *          M314=G314 M314=G314
(00648) *          M315=G315 M315=G315
(00649) *          M316=G316 M316=G316
(00650) *          M317=G317 M317=G317
(00651) *          M318=G318 M318=G318
(00652) *          M319=G319 M319=G319
(00653) *          M320=G320 M320=G320
(00654) *          M321=G321 M321=G321
(00655) *          M322=G322 M322=G322
(00656) *          M323=G323 M323=G323
(00657) *          M324=G324 M324=G324
(00658) *          M325=G325 M325=G325
(00659) *          M326=G326 M326=G326
(00660) *          M327=G327 M327=G327
(00661) *          M328=G328 M328=G328
(00662) *          M329=G329 M329=G329
(00663) *          M330=G330 M330=G330
(00664) *          M331=G331 M331=G331
(00665) *          M332=G332 M332=G332
(00666) *          M333=G333 M333=G333
(00667) *          M334=G334 M334=G334
(00668) *          M335=G335 M335=G335
(00669) *          M336=G336 M336=G336
(00670) *          M337=G337 M337=G337
(00671) *          M338=G338 M338=G338
(00672) *          M339=G339 M339=G339
(00673) *          M340=G340 M340=G340
(00674) *          M341=G341 M341=G341
(00675) *          M342=G342 M342=G342
(00676) *          M343=G343 M343=G343
(00677) *          M344=G344 M344=G344
(00678) *          M345=G345 M345=G345
(00679) *          M346=G346 M346=G346
(00680) *          M347=G347 M347=G347
(00681) *          M348=G348 M348=G348
(00682) *          M349=G349 M349=G349
(00683) *          M350=G350 M350=G350
(00684) *          M351=G351 M351=G351
(00685) *          M352=G352 M352=G352
(00686) *          M353=G353 M353=G353
(00687) *          M354=G354 M354=G354
(00688) *          M355=G355 M355=G355
(00689) *          M356=G356 M356=G356
(00690) *          M357=G357 M357=G357
(00691) *          M358=G358 M358=G358
(00692) *          M359=G359 M359=G359
(00693) *          M360=G360 M360=G360
(00694) *          M361=G361 M361=G361
(00695) *          M362=G362 M362=G362
(00696) *          M363=G363 M363=G363
(00697) *          M364=G364 M364=G364
(00698) *          M365=G365 M365=G365
(00699) *          M366=G366 M366=G366
(00700) *          M367=G367 M367=G367
(00701) *          M368=G368 M368=G368
(00702) *          M369=G369 M369=G369
(00703) *          M370=G370 M370=G370
(00704) *          M371=G371 M371=G371
(00705) *          M372=G372 M372=G372
(00706) *          M373=G373 M373=G373
(00707) *          M374=G374 M374=G374
(00708) *          M375=G375 M375=G375
(00709) *          M376=G376 M376=G376
(00710) *          M377=G377 M377=G377
(00711) *          M378=G378 M378=G378
(00712) *          M379=G379 M379=G379
(00713) *          M380=G380 M380=G380
(00714) *          M381=G381 M381=G381
(00715) *          M382=G382 M382=G382
(00716) *          M383=G383 M383=G383
(00717) *          M384=G384 M384=G384
(00718) *          M385=G385 M385=G385
(00719) *          M386=G386 M386=G386
(00720) *          M387=G387 M387=G387
(00721) *          M388=G388 M388=G388
(00722) *          M389=G389 M389=G389
(00723) *          M390=G390 M390=G390
(00724) *          M391=G391 M391=G391
(00725) *          M392=G392 M392=G392
(00726) *          M393=G393 M393=G393
(00727) *          M394=G394 M394=G394
(00728) *          M395=G395 M395=G395
(00729) *          M396=G396 M396=G396
(00730) *          M397=G397 M397=G397
(00731) *          M398=G398 M398=G398
(00732) *          M399=G399 M399=G399
(00733) *          M400=G400 M400=G400
(00734) *          M401=G401 M401=G401
(00735) *          M402=G402 M402=G402
(00736) *          M403=G403 M403=G403
(00737) *          M404=G404 M404=G404
(00738) *          M405=G405 M405=G405
(00739) *          M406=G406 M406=G406
(00740) *          M407=G407 M407=G407
(00741) *          M408=G408 M408=G408
(00742) *          M409=G409 M409=G409
(00743) *          M410=G410 M410=G410
(00744) *          M411=G411 M411=G411
(00745) *          M412=G412 M412=G412
(00746) *          M413=G413 M413=G413
(00747) *          M414=G414 M414=G414
(00748) *          M415=G415 M415=G415
(00749) *          M416=G416 M416=G416
(00750) *          M417=G417 M417=G417
(00751) *          M418=G418 M418=G418
(00752) *          M419=G419 M419=G419
(00753) *          M420=G420 M420=G420
(00754) *          M421=G421 M421=G421
(00755) *          M422=G422 M422=G422
(00756) *          M423=G423 M423=G423
(00757) *          M424=G424 M424=G424
(00758) *          M425=G425 M425=G425
(00759) *          M426=G426 M426=G426
(00760) *          M427=G427 M427=G427
(00761) *          M428=G428 M428=G428
(00762) *          M429=G429 M429=G429
(00763) *          M430=G430 M430=G430
(00764) *          M431=G431 M431=G431
(00765) *          M432=G432 M432=G432
(00766) *          M433=G433 M433=G433
(00767) *          M434=G434 M434=G434
(00768) *          M435=G435 M435=G435
(00769) *          M436=G436 M436=G436
(00770) *          M437=G437 M437=G437
(00771) *          M438=G438 M438=G438
(00772) *          M439=G439 M439=G439
(00773) *          M440=G440 M440=G440
(00774) *          M441=G441 M441=G441
(00775) *          M442=G442 M442=G442
(00776) *          M443=G443 M443=G443
(00777) *          M444=G444 M444=G444
(00778) *          M445=G445 M445=G445
(00779) *          M446=G446 M446=G446
(00780) *          M447=G447 M447=G447
(00781) *          M448=G448 M448=G448
(00782) *          M449=G449 M449=G449
(00783) *          M450=G450 M450=G450
(00784) *          M451=G451 M451=G451
(00785) *          M452=G452 M452=G452
(00786) *          M453=G453 M453=G453
(00787) *          M454=G454 M454=G454
(00788) *          M455=G455 M455=G455
(00789) *          M456=G456 M456=G456
(00790) *          M457=G457 M457=G457
(00791) *          M458=G458 M458=G458
(00792) *          M459=G459 M459=G459
(00793) *          M460=G460 M460=G460
(00794) *          M461=G461 M461=G461
(00795) *          M462=G462 M462=G462
(00796) *          M463=G463 M463=G463
(00797) *          M464=G464 M464=G464
(00798) *          M465=G465 M465=G465
(00799) *          M466=G466 M466=G466
(00800) *          M467=G467 M467=G467
(00801) *          M468=G468 M468=G468
(00802) *          M469=G469 M469=G469
(00803) *          M470=G470 M470=G470
(00804) *          M471=G471 M471=G471
(00805) *          M472=G472 M472=G472
(00806) *          M473=G473 M473=G473
(00807) *          M474=G474 M474=G474
(00808) *          M475=G475 M475=G475
(00809) *          M476=G476 M476=G476
(00810) *          M477=G477 M477=G477
(00811) *          M478=G478 M478=G478
(00812) *          M479=G479 M479=G479
(00813) *          M480=G480 M480=G480
(00814) *          M481=G481 M481=G481
(00815) *          M482=G482 M482=G482
(00816) *          M483=G483 M483=G483
(00817) *          M484=G484 M484=G484
(00818) *          M485=G485 M485=G485
(00819) *          M486=G486 M486=G486
(00820
```

```

113 05078 00J000E (00385)  MUP\MOV(IQ,M6) 3 ,F5
114 05079 05E05E0 (00386)  MUL(M3,M7)\MUL(M3,M7) 3G7*K0,G6*K7
115 05079 08E0000 (00387)  MOV(IQ,M5)\NOP IK4
116 05079 0000000 (00388)  MUP\MOV(IQ,M5) 3 ,F3
117 05079 08E0000 (00389)  MOV(IQ,M4)\NOP IK2
118 05079 0000000 (00390)  MUP\MOV(IQ,M4) 3 ,K1
119 05079 08E0000 (00391)  MOV(IQ,M4)\NOP IF8=INPUT
120 05079 08E0000 (00392)  MOV(P,A6)\NOP 3G7*K0
121 05079 08E0000 (00393)  SUB(A0,A6)\NOP 3F8-G7*K0>F7
122 05079 08E0000 (00394)  MUL(M2,M6)\MOV(A6),MUL(M2,M6) 3G5*K6,G4*K5
123 05079 08E0000 (00395)  MOV(R,EX0)\NOP 3F7
124 05079 08E0000 (00396)  MUP\MOV(EXI,A0) 3 ,F7
125 05079 08E0000 (00397)  MUP\SUB(A0,A6) 3 ,F7-G6*K7>F6
126 05079 08E0000 (00398)  MUP\MOV(R,M3) 3 ,F6
127 05079 08E0000 (00399)  MOV(A4),MUL(M1,M5)\MOV(A4),MUL(M3,M7) 3(G5*K6)G3*K4,(G4K5)F6K7
128 05079 08E0000 (00400)  MUP\MOV(R,EX0) 3 ,F6
129 05079 08E0000 (00401)  MOV(EXI,A6)\NOP 3F6
130 05079 08E0000 (00402)  SUB(A6,A4)\NOP 3F6-G5K6>F5
131 05079 08E0000 (00403)  MUP(R,FX0)\NOP 3F5
132 05079 08E0000 (00404)  MOV(R,M2)\MOV(EXI,A2) 3F5,F5
133 05079 08E0000 (00405)  MOV(A2),MUL(M2,M6)\MOV(A6),MUL(M1,M5) 3(G3K4)F5K6,(F6K7)G2K3
134 05079 08E0000 (00406)  MUP\SUB(A2,A4) 3 ,F5-G4K5>F4
135 05079 08E0000 (00407)  MUP\MOV(R,EX0) 3 ,F4
136 05079 08E0000 (00408)  MOV(EXI,A4)\NOP 3F4
137 05079 08E0000 (00409)  SUB(A4,A2)\MOV(M2),ADD(A7,A6) 3F4-G3K4>F3,(F4)G6+P6K7>G7
138 05079 08E0000 (00410)  MOV(A6),MUL(M2,M4)\MOV(A2),MUL(M2,M6) 3(F5K6)G1K2,(G2K3)F4K5
139 05079 08E0000 (00411)  MOV(R,EX0)\MOV(R,EX0) 3F3,G7
140 05079 08E0000 (00412)  MOV(EXI,M3)\MOV(EXI,A4) 3G7(NEW),F3
141 05079 08E0000 (00413)  MOV(M1),ADD(A5,A6)\SUB(A4,A2) 3(F3)G5+FSK6>G6,F3-G2K3>F2
142 05079 08E0000 (00414)  MOV(EXI,A7)\NOP 3G7(NEW)
143 05079 08E0000 (00415)  MOV(A0),MUL(M1,M5)\MOV(A4),MUL(M0,M4) 3(G1K2)F3K4,(F4K5)G0K1
144 05079 08E0000 (00416)  MUP\MOV(R,EX0) 3 ,F2
145 05079 08E0000 (00417)  MOV(EXI,A2)\NOP 3F2
146 05079 08E0000 (00418)  MOV(EX0),SUB(A2,A0)\MOV(M1),ADD(A5,A4) 3(G6)F2-G1K2>F1,(F2)G4+
147 05079 08E0000 (00419)  3F4K5>G5
148 05079 08E0000 (00420)  MUP\MOV(EXI,A7) 3 ,G6(NEW)
149 05079 08E0000 (00421)  MUP\MOV(EXI,M3) 3 ,G6(NEW)
150 05079 08E0000 (00422)  MOV(R,EX0)\MOV(R,EX0) 3F1,G5
151 05079 08E0000 (00423)  MOV(R,M0)\MOV(EXI,A2) 3F1,F1
152 05079 08E0000 (00424)  MOV(A4),MUL(M0,M4)\MOV(A0),MUL(M1,M5) 3(F3K4)F1K2,(G0K1)F2K3
153 05079 08E0000 (00425)  ADD(A3,A4)\SUB(A2,A0) 3G3+P3K4>G4,F1-G0K1>F0
154 05079 08E0000 (00426)  MOV(EXI,A5)\NOP 3G5(NEW)
155 05079 08E0000 (00427)  MOV(EXI,M2)\NOP 3G5(NEW)
156 05079 08E0000 (00428)  MUP(R,EX0)\MOV(M,M0) 3G4,F0=00(NEW)
157 05079 08E0000 (00429)  MOV(P,A2)\MOV(A2),MUL(M0,M4) 3F1K2,F0K1
158 05079 08E0000 (00430)  ADD(A1,A2)\MOV(EX0),ADD(A3,A2) 3G1+P1K2>G2,(F0)G2+P2K3>G3
159 05079 08E0000 (00431)  MOV(EXI,M0)\MOV(EXI,M2) 3F0 OUT,F4(NEW)
160 05079 08E0000 (00432)  MOV(EXI,A1)\MOV(EXI,A5) 3F0=00,G4(NEW)
161 05079 08E0000 (00433)  MOV(R,EX0)\NOP 3C2
162 05079 08E0000 (00434)  JUMPS(LD0NE,F1) 3END IF INPUT USED UP
163 05079 08E0000 (00435)  MOV(IQ,A0)\NOP 3F0(NEW)=INPUT
164 05079 08E0000 (00436)  MUL(M3,M7)\MOV(A0),MUL(M3,M7) 3G7K0,(F0K1)G6K7
165 05079 08E0000 (00437)  R(A1)\MOV(EX0),ADD(A1,A0) 3F0=00,(G3)G0+F0K1>G1

```

PAGE 12: [RND]<DCA16>RBM16M.MSU.2, 29-Dec-80 14:38:40, Ed: WULF
APU3-VLTSY VECTOR LATTICE SYNTHESIS FILTER

```

A47 05D98 08490849 (00438)
A48 05D92 08510853 (00439)
A49 05D94 08900890 (00440)
A4A 05D96 08400800 (00441)
A4B 05D98 08510851 (00442)
A4C 05D9A 08500856 (00443)
A4D 05D9C 4E000000 (00444)
A4E 05D9E 10000010 (00445)
      (00446) *
A4F 05DA0 00000000 (00447) *
A50 05DA2 02200038 (00448) LDONE:
A51 05DA4 08510853 (00449)
A52 05DA6 02F002F8 (00450)
A53 05DA8 08900800 (00451)
A54 05DAA 02A0020C (00452)
A55 05DAC 08900800 (00453)
A56 05DAE 0260027C (00454)
A57 05DB0 08900800 (00455)
A58 05DB2 0800089C (00456)
A59 05DB4 08500800 (00457)
A5A 05DB6 0800085C (00458)
A5B 05DB8 20300800 (00459)
A5C 05DBA 08000800 (00460)
A5D 05DBC 10000000 (00461)
      (00462) *
      (00463) *
      0000005E (00464)
      05DBE (00465)
      (00466) *
      (00467) *

MOV(EXT,M1)\MOV(EXT,M1) JG3(NEW),G2(NEW)
MOV(EXT,A3)\MOV(EXT,A3) JG3(NEW),G2(NEW)
MOV(EXT,EXD)\MOV(EXT,EXD) JG6=FB,G1
MOV(EXT,M0)\NOP JG1(NEW)
MOV(EXT,A1)\MOV(EXT,A1) JG1(NEW),G0(NEW)
MOV(EXT,M6)\MOV(EXT,M6)\MOV(EXT,M6) JG7(NEW),G4(NEW)
SUB(A0,A6)\NOP JF8-G7K8>F7
JUMP(LOOP)

FINISH LAST PART OF LAST LOOP, AND OUTPUT G'S
NOPAND(P,A0) J ,F0K1
R(A1)\MOV(EXT),ADD(A1,A0) JF8=GB,(G3)G0+P0K1
MOV(EXT,A3)\MOV(EXT,A3) JG3(NEW),G2(NEW)
MOV(EXT,EXD)\MOV(EXT,EXD),R(A7) JG0G7,(G1)G6
MOV(EXT,M0)\NOP JG7 OUT
R(A5)\MOV(EXT),R(A5) JG5,(G6 OUT),G4
MOV(EXT,M0)\NOP JG5 OUT
R(A3)\MOV(EXT),R(A3) JG3,(G4 OUT),G2
MOV(EXT,M0)\NOP JG3 OUT
NOPAND(P,A0) J ,G2 OUT
MOV(EXT,M0)\NOP JG1 OUT
NOPAND(P,A0)\NOP J ,G0 OUT
CLEAR(RA)\NOP JHALT
NOP
JUMP(0)

VLTSY$SZ=RA-VLTSY$SA
END VLTSY$SZ
EVEN

```

PAGE 13: (RHNJ)COCAL6>BBN16H.MSU.2, 29-Dec-80 14:30:40, Ed: WULF
APS3-V3200 APS PROGRAM FOR VLTSY

```

(00468) * APS3-V3200 APS PROGRAM FOR VLTSY
(00469) *
(00470) *
(00471) *
(00472) *
(00473) *
(00474) *
(00475) *
(00476) *
(00477) *
(00478) *
(00479) *
(00480) *
(00481) *
(00482) *
(00483) *
(00484) *
(00485) *
(00486) *
(00487) *
(00488) *
(00489) *
(00490) *
(00491) *
(00492) *
(00493) *
(00494) *
(00495) *
(00496) *
(00497) *
(00498) *
(00499) *
(00500) *
(00501) *
(00502) *
(00503) *
(00504) *
(00505) *
(00506) *
(00507) *
(00508) *
(00509) *
(00510) *
(00511) *
(00512) *
(00513) *
(00514) *
(00515) *
(00516) *
(00517) *
(00518) *
(00519) *
(00520) *

050BE 00005E2A (00481)
050C0 00000000 (00482)
050C2 0000 (00483)
050C3 0000 (00484)
050C4 00005E22 (00485)
050C5 00000000 (00486)
050C6 00201D40 (00495)
050C8 00230030 (00496)
050C9 00000000 (00497)
050CA 04401000 (00506)
050CB 06500000 (00507)
050CC 00600000 (00508)
050CD 0A200031 (00509)
050CE 0C00000F (00511)
050CF 0F200601 (00512)
050D0 100A900C (00514)
050D1 12000007 (00516)
050D2 14020004 (00517)
050D3 16100001 (00518)
050D4 18000000 (00519)
050D5 1A000000 (00520)

* APS3-V3200 APS PROGRAM FOR VLTSY
MAP-300 APS PROGRAM FOR VLTSY(U,V,W)
CODED BY KFIELD 3/79

INPUT STREAM: U(7),...,U(0),V(7),...,V(0),W(0),...,W(WBS-1),CP13
OUTPUT STREAM: Y(0),...,Y(WBS-1),U(7),...,U(0),LE03

HEADER BLOCK

EVEN V3200$1 JPTR TO CONSTR. INSTR. BLK.
ADDR 0 JPTR TO SCALAR BLOCK (NO SCALARS)
DATA 0 JNUMBER OF SCALARS
DATA V3200$Z JMODULE SIZE
ADDR V3200$A JPTR TO CHAIN ANCHOR
EVEN

REGIM APS(V3200)

INPUT PROGRAM

JSM(V3200$6,P2) JSET OUTPUT PC
SET(RO) JTURN ON OUTPUT GENERATION

INPUT PCM REGISTER USAGES:
RR0: VECTOR ELEMENT ADDRESSES
RR1: BUFFER SIZES
RR2: BUFFER SPACINGS

GENERATE "U" ADDRESSES
LOAD(BR0,13) JBR0 <= VECTOR U BASE ADDR
LOAD(BR1,MSS) JDUHNY LOAD
LOAD(BR2,MSS) JBR2 <= VECTOR U SPACING
SUBL(BR2,1) JBP2 <= SPACING-1
ADDL(BR0,7) JGEN ADDR OF U(7)
SUBL(BR2,1),JUNPP(B1) JADD (7 * SPACING)
ADD(BR0,193) JBR0 <= ADDR(U7)+SPACING
LOAD(BR1,7) JBR1 <= 7
SUBL(BR0,191,TF) JGEN ADDR (U(1))
SUBL(BR1,1),JUNPP(B2) JLOOP 8 TIMES

```

```

(00521) *
(00522) *
ABC 050DE 18402004 (00523)
APD 050F0 1A500000 (00524)
APE 050F2 1C600000 (00525)
APF 050F4 1E200001 (00526)
A10 050F6 20400010 (00527) *
A11 050F8 22201081 (00528) #3
A12 050FA 240AA00C (00529) *
A13 050FC 26500007 (00530) *
A14 050FE 282A0004 (00531) *
A15 050F0 2A191401 (00532) *
A16 050F2 2C403004 (00533) *
A17 050F4 2E500000 (00534) *
A18 050F6 30200000 (00535) *
A19 050F8 328A0006 (00536) *
A1A 050FA 34191001 (00537) *
A1B 050FC 36200003 (00538) *
A1C 050FE 38000020 (00539) *
A1D 050F0 3A300002 (00540) *
A1E 050F2 3C40000A (00541) *
A1F 050F4 3E500000 (00542) *
A20 050F6 40000000 (00543) *
A21 050F8 42700006 (00544) *
A22 050FA 44500000 (00545) *

GENERATE 'V' ADDRESSES
LOAD(BR0,C33)
LOAD(BR1,M$S)
LOAD(BR2,M$S)
SUBL(BR2,1)
ADDL(BR0,7)
SUBL(BR2,1),JUMPP(B3)
ADD(BR0,C103)
LOAD(BR1,7)
SUB(RN0,C103,TF)
SUBL(BR1,1),JUMPP(B4)

GENERATE 'W' ADDRESSES
LOAD(BR0,C33)
LOAD(BR1,M$S)
SUB(RN0,M$S)
ADD(BR0,C113,TF)
SUBL(BR1,1),JUMPP(B5)

WHEN DONE GENERATING INPUT ADDRESSES, CLEAR RI
CLEAR(RI)
NOP(0)

OUTPUT PROGRAM

OUTPUT PGM REG USAGES:
  RW0: VECTOR ELEMENT ADDRESSES
  RW1: BUFFER SIZES
  RW2: BUFFER SPACINGS
  RW3: SCRATCH REGISTER

TURN ON APU

GEN 'V' ADDRESSES
LOAD(BW0,C03)
LOAD(BW1,M$S)
SUB(BW0,M$S)
GET WBS-1 (GENERATE WBS V'S)
LOAD(BW3,C33)
LOAD(BW1,M$S)

JBR0 <= VECTOR V BASE ADDR
JBR1 <= WBS-1
JBR2 <= W BASE MINUS SPACING
JGEN ADDR(W(1))
JLOOP WBS TIMES

JBR0 <= VECTOR W BASE ADDR
JBR1 <= WBS-1
JBR2 <= W BASE MINUS SPACING
JGEN ADDR(W(1))
JLOOP WBS TIMES

JBR0 <= Y BASE ADDR
JBR1 <= Y BASE MINUS SPACING
JGEN ADDR(W(1))
JLOOP WBS TIMES

```


PAGE 15: (00MD)DCAL6>RBN16M.MS0.2, 29-Dec-80 14:30:40, E3: WOLF
AP53-V3200 APS PROGRAM FOR VLTSY

A23 05E0C 46700000 (00574) *	LOAD(BW3,MSS)	JUNMV LOAD
A24 05E0E 400A0006 (00575) *		
A25 05E10 40112401 (00576) 07	ADD(BW0,C03,TF)	JGEN ADDR(V(1))
	SUBL(BW1,1),JUMPP(07)	FLOOP WBS TIMES
	(00577) *	
	(00578) *	
	(00579) *	
	(00580) *	
A26 05E12 4C401004 (00581)	LOAD(BW0,C13)	JBW0 <= VECTOR U BASE ADDRESS
A27 05E14 4F500000 (00582)	LOAD(BW1,MSS)	JUNMV LOAD
A28 05E16 50600000 (00583)	LOAD(BW2,MSS)	JBW2 <= VECTOR U SPACING
A29 05E18 52210031 (00584) *	SUBL(BW2,1)	JBW2 <= SPACING-1
	(00585) *	
A2A 05E1A 5401003F (00586) 08	ADD(BW0,7)	JGEN ADDR OF U(7)
A2B 05E1C 56212401 (00587) *	SUBL(BW2,1),JUMPP(08)	JADD (7*SPACING)
	(00588) *	
A2C 05E1E 500A'00C (00589) *	ADD(BW0,C93)	JBW0 <= ADDR(U(7)) + SPACING
	(00590) *	
A2D 05E20 5A500007 (00591) *	LOAD(BW1,7)	JBW1 <= 7
A2E 05E22 5C020004 (00592) 09	SUB(BW0,C91,TF)	JGEN ADDR(U(1))
A2F 05E24 5E112EH1 (00593) *	SUBL(BW1,1),JUMPP(09)	FLOOP 8 TIMES
	(00594) *	
	(00595) *	
	(00596) *	
A30 05E26 60200030 (00597) *	CLEAR(00)	JHALT OUTPUT
A31 05E28 62000020 (00598) *	NOP(0)	
	(00599) *	
05E2A 00005E22 (00600) V3200SA=0C	END 0A-1	JASSIGN VALUE TO CHAIN ANCHOR
	(00601) *	JEND OF MODULE
	(00602) *	
	(00603) *	
	(00604) *	
05E2A 00000000 (00605) V3200ST DATA 14F'0.0'		
...		
00000000 (00606) V3200SZ=AL-V3200S		JMODULE SIZE
	(00607) *	

```

PAGE 17: (00ND)XDCALC>00016M.N50.2, 29-Dec-88 14:38:48, Ed: WOLF
APU3-VTDAT(Y,W) CONVERT REFL-M (K) TO LPC (A) COEFFICIENTS

A05 05E52 00C00020 (006611) MOV(TQA,M0) JINPUT K(M)
(00662) JINNER LOOP
A06 05E54 00F0002C (006633) MOV(TQA,M0) \ MOV(TQA,M4) JACM-13(M-L)
A07 05E56 00E00070 (006644) MOV(TQA,M4) \ MOV(TQA,M0) JACM-13(L)
A08 05E58 04000400 (006645) MUL(M0,M4) K(M)*ACM-13(L) \ K(M)*ACM-13(M-L)
A09 05E5A 0010001 (006666) MOV(P,A1)
A0A 05E5C 41004100 (006677) ADD(M0,A1) JACM-13(M-L) + K(M)*ACM-13(L) \
(006680) J JACM-13(L) + K(M)*ACM-13(M-L)
A0B 05E5E 009C009C (006699) MOV(R,OQ) JACM(M-L) \ ACM(L)
A0C 05E60 90160006 (00670) JUMPC(R2,PV1) JUMP IF NOT DONE THIS ITERATION
A0D 05E62 90100000 (00671) J3: JUMPC(R3,OQE)
A0E 05E64 00000000 (00672) NOP WAIT FOR THIS ITERATION'S RESULT TO BE
A0F 05E66 20172037 (00673) CLEAR(VI) BE WRITTEN BEFORE LETTING APS PROCEED
A10 05E68 901C0004 (00674) JUMPC(R1,E0) TO READ OPERANDS FOR NEXT ITERATION
A11 05E6A 20222032 (00675) CLEAR(RA) JUMP IF MORE ITERATIONS YET TO DO
A12 05E6C 10000000 (00676) JUMP(0)
(0000013 (00677) VKA0$82=BA-VKA0$5A
05E6E (00678) END

```

PAGE 16: [08N03]DCAL6>08N16M.NSO.2, 29-Dec-88 14:38:46, Ed: WOLF
AP03-VKTOA(V,U) CONVERT REFL-M (K) TO LPC (A) COEFFICIENTS

```

(00600) * AP03-VKTOA(V,U) CONVERT REFL-M (K) TO LPC (A) COEFFICIENTS
(00609) )
(00610) ) J. J. WOLF 6/30/88
(00611) ) BINDS TO APS3-VKTOA
(00612) )
(00613) ) THIS VKTUA IS DIFFERENT FROM THAT IN 08N300.NSO (9.6 KB/S CODE)
(00614) ) IN THAT (1) BOTH INPUT AND OUTPUT BUFFERS ARE FLT PT AND (2) THIS
(00615) ) IS WRITTEN TO HANDLE A GENERAL COEFFICIENT VECTOR (ANY LENGTH).
(00616) )
(00617) ) BUFFER V: OUTPUT, LPC COEFFS: A(1), A(2), ..., A(M). A(0), WHICH IS
(00618) ) 1.0, IS NOT OUTPUT.
(00619) ) BUFFER U: INPUT, REFLECTION COEFFS, K(1), K(2), ..., K(M)
(00620) ) NOTE: BUFFERS V AND U MUST BE COMPACT 32-BIT FLT PT.
(00621) )
(00622) ) THE LPC COEFFICIENTS ARE COMPUTED FROM THE RECURSION:
(00623) ) (1)  $ACM(N) = K(N)$ 
(00624) ) (2)  $ACM(L) = ACM-1(L) + K(M) \cdot ACM-1(M-L)$ 
(00625) )  $M=1,2,...,N$ 
(00626) )  $L=1,2,...,N-1$ 
(00627) ) WHERE C.J INDICATES ITERATION AND (.) INDICATES INDEX.
(00628) ) IN ORDER THAT THE COMPUTATION BE DONE IN PLACE, WE DO EQ. (2) IN
(00629) ) "SYMMETRIC" PAIRS:
(00630) ) (2A)  $ACM(M-L) = ACM-1(M-L) + K(M) \cdot ACM-1(L)$ 
(00631) ) (2B)  $ACM(L) = ACM-1(L) + K(M) \cdot ACM-1(M-L)$ 
(00632) )  $L=1,2,...,(M-1)/2$ 
(00633) )
(00634) ) APU REGISTER USAGE:
(00635) ) A(0): ACM-1(M-L) ACM-1(L)
(00636) ) A(1): P-A K-A
(00637) ) M(0): K(M)
(00638) ) M(4): ACM-1(L) ACM-1(M-L)
(00639) )
(00640) ) INPUT, OUTPUT STREAMS:
(00641) ) N=1 M=2 L=1
(00642) ) IQ= K(1)CM1, K(2),A(1),A(1)CM1, K(3),A(2),A(1)CM1,
(00643) ) QO= A(1), A(2),A(1),A(1), A(3),A(2),A(1),
(00644) ) M=4 L=1 L=2
(00645) ) K(4),A(3),A(1),A(2),A(2)CM1,...
(00646) ) A(4),A(3),A(1),A(2),A(2), ...C(0)
(00647) ) THE NUMBER OF ITERATIONS IS UBS.
(00648) )
(00649) ) EVEN
(00650) ) DATA VKAUSSA )START ADDRESS
(00651) ) DATA VKAUSSZ )SIZE
(00652) )
(00653) ) VKAU: BEGIN APU(VKAU)
(00654) ) VKAUSSA:
(00655) )
(00656) ) MUV(IQA,OQ) \ NOP
(00657) ) JUMP(CR10,UQE)
(00658) ) NOP
(00659) ) CLEAR(MI)
(00660) ) ITERATION LOOP
(00661) ) MUV(IQA,OQ) \ NOP
(00662) ) ACM(M)=K(M)
(00663) )
(00664) ) JAL1(1)=K(1). END OF M=1 ITERATION.
(00665) ) WAIT FOR M=1 RESULT TO BE WRITTEN
(00666) ) THEN LET APS PROCEED
(00667) )
(00668) ) JAL1(M)=K(M)
(00669) )
(00670) )
(00671) )
(00672) )
(00673) )
(00674) )
(00675) )
(00676) )
(00677) )
(00678) )
(00679) )
(00680) )
(00681) )
(00682) )
(00683) )
(00684) )
(00685) )
(00686) )
(00687) )
(00688) )
(00689) )
(00690) )
(00691) )
(00692) )
(00693) )
(00694) )
(00695) )
(00696) )
(00697) )
(00698) )
(00699) )
(00700) )

```

```

(00679) - APS3-VKTOA(Y,U)      APS PROGRAM FOR VKTDA
(00680) J J. WOLF, 7/2/80
(00681) J
(00682) JHEADER BLCK
(00683) J
(00684) EVEN
(00685) ADDR VKASSI
(00686) ADDR 0
(00687) DATA 0
(00688) DATA VKASSZ
(00689) ADDR VKASSA
(00690) EVEN
(00691) J
(00692) JINPUT PROGRAM
(00693) J BR0: COUNTS ITERATIONS FROM 2 TO UBS
(00694) J BR1: K(M), A(K)
(00695) J BR2: M1=2*(M-1)
(00696) J BR3: M' COUNTS FROM 2*(M-1) BY 2 TO 0 OR -2
(00697) J
(00698) VKASSI:
(00699) LOAD(BR1,C17,TF)
(00700) LOAD(BR0,M$)
(00701) LOAD(BR3,M$)
(00702) SUBL(BR0,1)
(00703) MOV8(BR0,BR0)
(00704) JSN(VKASSO,02)
(00705) SET(R0)
(00706) LOAD(BR2,0)
(00707) SET(M1)
(00708) NOP(0)
(00709) JITERATION LOOP. WE BEGIN WITH THE M=2 ITERATION.
(00710) M4:
(00711) ADDL(BR2,2)
(00712) LOAD(BR1,C17)
(00713) LOAD(BR3,M$)
(00714) ADDH(BR1,BR2,TF)
(00715) LOAD(BR1,C17)
(00716) LOAD(BR3,M$)
(00717) SUB(BR1,M$)
(00718) MOV8(BR3,BR2),JUMP(VK6)
(00719) JIMMER LOOP: L-1,2,...,(M-1)/2
(00720) M5:
(00721) ADDL(BR3,M$)
(00722) SUBL(BR3,M$)
(00723) JTHE NEXT 3 INSTRUCTIONS ARE UNINTERRUPTIBLE, ENSURING THAT ON THE FINAL
(00724) J LOOP OF THE ITERATION, M1 GETS SET 2 APU-CYCLES AFTER THE LAST READ
(00725) J ADDRESS IS GENERATED, SO IT'S SURE TO BE SET BEFORE THE APU CAN TEST
(00726) J IT.
(00727) SUB8(BR1,BR3,TF),JUMP(BA+1)
(00728) SUBL(BR3,2*M$),JUMPR(M5)
(00729) SET(M1)
(00730) NOP(0)
(00731) SUBL(BR0,1),JUMPP(M4)
(00732) J
(00733) J
(00734) J
(00735) J
(00736) J
(00737) J
(00738) J
(00739) J
(00740) J
(00741) J
(00742) J
(00743) J
(00744) J
(00745) J
(00746) J
(00747) J
(00748) J
(00749) J
(00750) J
(00751) J
(00752) J
(00753) J
(00754) J
(00755) J
(00756) J
(00757) J
(00758) J
(00759) J
(00760) J
(00761) J
(00762) J
(00763) J
(00764) J
(00765) J
(00766) J
(00767) J
(00768) J
(00769) J
(00770) J
(00771) J
(00772) J
(00773) J
(00774) J
(00775) J
(00776) J
(00777) J
(00778) J
(00779) J
(00780) J
(00781) J
(00782) J
(00783) J
(00784) J
(00785) J
(00786) J
(00787) J
(00788) J
(00789) J
(00790) J
(00791) J
(00792) J
(00793) J
(00794) J
(00795) J
(00796) J
(00797) J
(00798) J
(00799) J
(00800) J
(00801) J
(00802) J
(00803) J
(00804) J
(00805) J
(00806) J
(00807) J
(00808) J
(00809) J
(00810) J
(00811) J
(00812) J
(00813) J
(00814) J
(00815) J
(00816) J
(00817) J
(00818) J
(00819) J
(00820) J
(00821) J
(00822) J
(00823) J
(00824) J
(00825) J
(00826) J
(00827) J
(00828) J
(00829) J
(00830) J
(00831) J
(00832) J
(00833) J
(00834) J
(00835) J
(00836) J
(00837) J
(00838) J
(00839) J
(00840) J
(00841) J
(00842) J
(00843) J
(00844) J
(00845) J
(00846) J
(00847) J
(00848) J
(00849) J
(00850) J
(00851) J
(00852) J
(00853) J
(00854) J
(00855) J
(00856) J
(00857) J
(00858) J
(00859) J
(00860) J
(00861) J
(00862) J
(00863) J
(00864) J
(00865) J
(00866) J
(00867) J
(00868) J
(00869) J
(00870) J
(00871) J
(00872) J
(00873) J
(00874) J
(00875) J
(00876) J
(00877) J
(00878) J
(00879) J
(00880) J
(00881) J
(00882) J
(00883) J
(00884) J
(00885) J
(00886) J
(00887) J
(00888) J
(00889) J
(00890) J
(00891) J
(00892) J
(00893) J
(00894) J
(00895) J
(00896) J
(00897) J
(00898) J
(00899) J
(00900) J
(00901) J
(00902) J
(00903) J
(00904) J
(00905) J
(00906) J
(00907) J
(00908) J
(00909) J
(00910) J
(00911) J
(00912) J
(00913) J
(00914) J
(00915) J
(00916) J
(00917) J
(00918) J
(00919) J
(00920) J
(00921) J
(00922) J
(00923) J
(00924) J
(00925) J
(00926) J
(00927) J
(00928) J
(00929) J
(00930) J
(00931) J
(00932) J
(00933) J
(00934) J
(00935) J
(00936) J
(00937) J
(00938) J
(00939) J
(00940) J
(00941) J
(00942) J
(00943) J
(00944) J
(00945) J
(00946) J
(00947) J
(00948) J
(00949) J
(00950) J
(00951) J
(00952) J
(00953) J
(00954) J
(00955) J
(00956) J
(00957) J
(00958) J
(00959) J
(00960) J
(00961) J
(00962) J
(00963) J
(00964) J
(00965) J
(00966) J
(00967) J
(00968) J
(00969) J
(00970) J
(00971) J
(00972) J
(00973) J
(00974) J
(00975) J
(00976) J
(00977) J
(00978) J
(00979) J
(00980) J
(00981) J
(00982) J
(00983) J
(00984) J
(00985) J
(00986) J
(00987) J
(00988) J
(00989) J
(00990) J
(00991) J
(00992) J
(00993) J
(00994) J
(00995) J
(00996) J
(00997) J
(00998) J
(00999) J
(01000) J

```

[illegible]

PAGE 28: (BUND) <DCAL6>BUND16M.H50.2, 29-Dec-88 14:38:48, Eds WOLF
APU3-PITCH(V,A,U,B,C,D) COMPUTE AND CODE PITCH

(00768) * APU3-PITCH(V,A,U,B,C,D) COMPUTE AND CODE PITCH
(00769) * ALSO COMPUTE ONE-TAP COEFF
(00770) * AND OUTPUT 3 AUTOCORRELATION
(00771) * COEFF CENTERED AT THE PITCH LAG

(00772) * WHR 8/88

(00773) *
(00774) *
(00775) *
(00776) *
(00777) *
(00778) *
(00779) *

INPUT: U AUTOCORRELATION COEFFICIENTS(U(14),...,U(133))

1111 NOTE: BUFFER U MUST BE COMPACT-1111

SC CONSTANT = 14.8

OUTPUTS:

SA PITCH LAG (IN SAMPLES)

SB CODED PITCH(INTEGER)

SD ONE-TAP COEFFICIENT

Y 3 AUTOCORRELATION COEFFICIENTS

CENTERED AT THE PITCH LAG

-U(PITCH LAG-1)

-U(PITCH LAG)

-U(PITCH LAG+1)

PROCEDURE:

DO SMAX(U) TO GET U(MAX) AND LOC OF MAX(=PITCH-14)

COMPUTE AND OUTPUT CODED PITCH=INTEGER OF MAX=>SB

COMPUTE AND OUTPUT PITCH LAG=MAX+14=>SA

COMPUTE INTEGER PITCH LAG FOR ADDRESS MOD

MODIFY APS ADDRESS

COMPUTE AND OUTPUT ONE-TAP COEFF =>SD

OUTPUT 3 AUTOCORRELATION COEFFS

CENTERED AT PITCH LAG ; U=>Y

INPUT STREAM: U(14),...,U(133), (FW1), (2*-15), SC(CONST=14.8), (FW1),
, "MODIFIED INSTR", (FW1), U(8), U(PITCH LAG-1), U(PITCH LAG)
, U(PITCH LAG+1)

OUTPUT STREAM: SB(CODED PITCH), SA(PITCH LAG=MAX+14), "INTEGER
PITCH", "MODIFIED INSTR", SD(ONE TAP COEFF)
, 3 AUTOCORRELATION COEFF'S (Y(8), Y(1), Y(2)).

WHERE:

"INTEGER PITCH" IS USED BY APS OUTPUT PCM TO MODIFY

APS INPUT PCM

"MODIFIED INSTR" IS APS INSTR TO BE MODIFIED.

IT IS READ FROM BUS1, AND WRITTEN

TO APS PSEUDO MEMORY.

REGISTER USAGE:

SMAK SECTION:

ALL A REGISTERS ARE USED

LOC OF MAX STORED IN A3

(00780) *

```

(00021) *      NONE OF THE M REGISTERS ARE USED EXCEPT M7
(00022) *      VALUE OF COEFF AT MAX LOC STORED IN M7
(00023) *      OTHERWISE:
(00024) *
(00025) *      A0 U(0)(SDIV) \      M0 SDIV \
(00026) *      A1 SDIV,M(PITCH-1) \ W(PITCH) M1 2*-15\
(00027) *      A2 U(MAX)/U(0) \      M2 ---\---
(00028) *      A3 LOC U(MAX) \      M3 --- \---
(00029) *      A4 CONST=14 \      M4 PITCH LAG=MAX+14,SDIV \
(00030) *      A5 LAG2*-15 \      M5 LOC U(MAX),-1 \
(00031) *      A6 SDIV,W(PITCH+1) \ M6 W(0)(SDIV) \
(00032) *      A7 MAX2*-15 \      M7 U(MAX) \
(00033) *
(00034) *      PITU - APV INITIALIZATION
(00035) *
(00036) *      EVEN
(00037) *      DATA PITU$A
(00038) *      DATA PITU$Z
(00039) *
(00040) *      PITU$: BEGIN APV(PITU)
(00041) *
(00042) *      DO SMAX ON V ARRAY (TERMINATE INPUT ON (PVI))
(00043) *
(00044) *      PITU$A: NOP \ K(1)          \ --- \ R=1
(00045) *      MOV(IQA,A7) \ MOV(R,A4)      \ A7=U(14) \ A6=1
(00046) *      MOV(ZERO,A6) \ MOV(R,A6)      \ A6=0   \ A6=1
(00047) *      K(2)          \ R=2          \ R=2
(00048) *      NOP \ MOV(IQA,A7)          \ --- \ A7=U(15)
(00049) *      MOV(R,A5) \ MOV(15),R(A7)     \ AS=2   \ A5=2, R=U(15)
(00050) *      JUMPS(PITU$1,PVI)
(00051) *
(00052) *      PITU$1: MOV(IQA,A0) \ NOP      \ A0=UE
(00053) *      MOV(A4),MAX(A7,A0) \ MOV(R,A7) \ A4=LOC, R=MAX \ A7=MAX
(00054) *      CALLS(PITU$3,T2)          \ --- \ IF NEW MAX
(00055) *      NOP \ ADD(A5,A4)          \ --- \ R=LOC
(00056) *      JUMPS(PITU$4,PVI)          \ EXIT IF U05 IS 000
(00057) *      MOV(R,A7) \ MOV(IQA,A0)      \ A7=MAX \ A0=U0
(00058) *      CALLS(PITU$2,T1)          \ IF NEW MAX \ ---
(00059) *      ADD(A5,A4) \ MOV(A4),MAX(A7,A0) \ A4=LOC, R=MAX
(00060) *      JUMPC(PITU$0,PVI)          \ EXIT IF U05 IS EVEN
(00061) *
(00062) *      EXIT IF LOOP BOTTOM
(00063) *      R=MAX \ ---
(00064) *      MOV(R,A1) \ MOV(0,X10)      \ X10=MAX \ IF NEW MAX
(00065) *      CALLS(PITU$3,T2)          \ ---
(00066) *      MOV(X11,A2) \ NOP          \ A2=MAX \ ---
(00067) *      JUMPC(PITU$5)
(00068) *
(00069) *      SUBROUTINE FOR LM SIZE
(00070) *      PITU$2: R(A4) \ NOP          \ R=LOC \ ---
(00071) *      MOV(R,A6) \ NOP          \ A6=LOC \ ---
(00072) *      RETURN
(00073) *

```

PAGE 22: (00001)OCAL16)BOMICH.N50.2, 29-Dec-88 14:30:00, R01 WOLF
AP03-PITCH(Y,A,B,C,B)
COMPUTE AND CODE PITCH

```

(00074) *
A10 05F14 00000200 PITUS3: MOP \ R(A4)
A11 05F16 00000096 MOP \ MOV(R,A6)
A12 05F18 00001000 RETURN
(00075) *
A13 05F1A 00000200 PITUS4: MOP \ R(A7)
A14 05F1C 00000096 MOP \ MOV(R,A1) \ MOV(R,EXO)
A15 05F1E 00000015 CALLS(PITUS2,T1)
A16 05F20 00000000 MOV(EXT,A2) \ MOP
(00076) *
A17 05F22 00000200 PITUS5: MAX(A1,A2) \ R(A6)
A18 05F24 00000000 MOP(R,M7) \ MOP
A19 05F26 00000025 JUMPS(PITUS6,T1)
A20 05F28 00000000 R(A6) \ MOP
A21 05F2A 00000000 MOP(R,A3) \ MOP
A22 05F2C 00000027 JUMP(PITUS7)
A23 05F2E 00000098 MOP \ MOV(R,EXO)
A24 05F30 00000000 MOV(EXT,A3) \ MOP
(00077) *
A25 05F32 00000200 PITUS7: CLEAR(M1)
A26 05F34 00000000 MOV(LOI,M1) \ MOP
A27 05F36 00000000 R(A3) \ MOP
A28 05F38 00000000 MOP(R,M5) \ MOP
A29 05F3A 00000000 MOP(M1,M5) \ MOP
A30 05F3C 00000000 MOV(P,A7) \ MOP
A31 05F3E 00000000 ALIGN(A7) \ MOP
(00078) *
A32 05F40 00000000 MOV(R,OQ) \ MOP
(00079) *
A33 05F42 00000000 ADD 14 TO MAX LOC TO GET PITCH LAG (FLT POINT VALUE)
A34 05F44 00000000 MOV(LOI,A4) \ MOP
A35 05F46 00000000 ADD(A3,A4) \ MOP
(00080) *
A36 05F48 00000000 MOV(R,OQ) \ MOP
(00081) *
A37 05F4A 00000000 GENERATE INTEGER PITCH LAG FOR ADDRESS MODIFICATION
A38 05F4C 00000000 (MULT BY (2**15), THEN ALIGN)
A39 05F4E 00000000 MOV(R,M4) \ MOP
A40 05F50 00000000 MUL(M1,M4) \ MOP
A41 05F52 00000000 MOV(P,A5) \ MOP
A42 05F54 00000000 ALIGN(A5) \ MOP
(00082) *
A43 05F56 00000000 MOV(R,M4) \ MOP
A44 05F58 00000000 MUL(M1,M4) \ MOP
A45 05F5A 00000000 MOV(P,A5) \ MOP
A46 05F5C 00000000 ALIGN(A5) \ MOP
(00083) *
A47 05F5E 00000000 SEND TO BUS1 TO MODIFY BUS1 COPY OF APS INSTR.

```


PAGE 231 (000010C0A1000010M.N00.2, 29-Dec-00 14:30:40, Ed: WOLF
AP03-PITCH(A,B,C,D) COMPUTE AND CODE PITCH

```

A36 05F50 00C0000 (00927) * MOV(R,00) \ NOP      J 00 <= INTEGER PITCH
    (00928) *
    (00929) * WAIT TIL BUS1 COPY IS MODIFIED, THEN READ
    (00930) * IT AND WRITE TO APS PSEUDO MEMORY.
    (00931) * (ALSO CLEAR WE SO APS INPUT PCN WILL CONTINUE.)
    (00932) * (IT WAS WAITING FOR WRITE TO COMPLETE.)
    (00933) *
A37 05F52 9010037 (00934) * PITUSW1: JUMPC(PITUSW1,0QE)  JLOOP WHILE OUTPUT QUEUE NOT EMPTY
A38 05F54 0000000 (00935) * NOP                      JDELAY 1 FOR MEMORY WRITE
A39 05F56 20372037 (00936) * CLEAR(W1)                JHUM LET APS PROCEED
    (00937) *
A3A 05F58 00C0000 (00938) * MOV(10A,00) \ NOP      J THEN READ MODIFIED APS INSTR
    (00939) *
    (00940) *
    (00941) * WAIT TIL APS COPY IS WRITTEN , THEN PROCEED APS INPUT PCN AGAIN
    (00942) *
A3B 05F5A 9010030 (00943) * PITUSW2: JUMPC(PITUSW2,0QE)  JLOOP WHILE OUTPUT QUEUE NOT EMPTY
A3C 05F5C 0000000 (00944) * NOP                      JDELAY 1 AS ABOVE
    (00945) *
    (00946) *
    (00947) *
    (00948) * SET BP DECS FOR DIVIDE: 0(MAX)/0(0)
    (00949) * M7 ALREADY = 0(MAX)
    (00950) * GET 0(0) FROM IO
    (00951) *
    (00952) * MOV(10A,M6) \ NOP      M6 = 0(0) \ ---
A3E 05F60 00C7000 (00953) * MOV(10A,AB) \ NOP      AB = 0(0) \ ---
A3F 05F62 00F0000 (00954) *
    (00955) * DO SOIV: DIVIDE 0/C
    (00956) * EXPECTS (IN LEFT ADM) 0 IN M7
    (00957) * C IN M6
    (00958) * C IN AB
    (00959) *
    (00960) * PITUS0: K(2) \ NOP      R=2
A40 05F64 1610000 (00961) * MOV(R,A6) \ NOP      A6=2
A41 05F66 00C0000 (00962) * RCP(A9) \ NOP      R1=1/C+DEL(=F0)(= 1/C + 6)
A42 05F68 2E00000 (00963) * MOV(R,M6) \ NOP      M6=F0
A43 05F6A 0000000 (00964) * MUL(M6,M6) \ NOP      P1=F0+C
A44 05F6C 0410000 (00965) * MOV(P,A1) \ NOP      A1=F0+C
A45 05F6E 0001000 (00966) * SUB(A6,A1) \ NOP      R2=2-F0+C (= 1 - C6)
A46 05F70 49C0000 (00967) * MOV(R,M4) \ NOP      M4=R2
A47 05F72 00C0000 (00968) * MUL(M6,M4) \ NOP      P2=F0*R2(=P1)(= 1/C - (6+2)C )
A48 05F74 0400000 (00969) * MOV(P,M6) \ NOP      M6=F1
A49 05F76 0010000 (00970) * MUL(M6,M6) \ NOP      P3=P1+C
A4A 05F78 0410000 (00971) * MOV(P,A1) \ NOP      A1=P3
A4B 05F7A 0001000 (00972) * SUB(A6,A1) \ NOP      R3=2-P1+C
A4C 05F7C 49C0000 (00973) * MOV(R,M4) \ NOP      M4=R3
A4D 05F7E 00C0000 (00974) * MUL(M6,M4) \ NOP      P4=P1*(2-P1+C)(=P2)(= 1/C - (6+4)C+3) (=1/C)
A4E 05F80 0400000 (00975) * MOV(P,M6) \ NOP      M6=F2
A4F 05F82 0001000 (00976) * MUL(M6,M7) \ NOP      P5=P2+0 (=R/C)
A50 05F84 0400000 (00977) * MOV(P,A2) \ NOP      A2 = TAP \ ---
A51 05F86 0007000 (00978) *
    (00979) * SOIV COMPLETE - NEGATE ONE-TAP COEFF AND OUTPUT

```

163

PAGE 26: (00001)<0CA16>000164.M50.2, 29-Dec-88 14:38:48, ED: WOLF
IP53-PTC00(V,A,U,B,C,D)


```

(01159) *
(01160) * GEN D SCALAR ADDR
(01161) *
(01162) * MOVW(BW3,BW3,TF)
(01163) *
(01164) * GENERATE AUTOCORRELATION CORRP ADDR
(01165) *
(01166) * LOAD(BW3,C03,TF)
(01167) * LOAD(BW3,M$5)
(01168) * ADD(BW3,M$5,TF)
(01169) * ADD(BW3,C03,TF)
(01170) *
(01171) * CLEAR(R0)
(01172) * NOP(0)
(01173) *
(01174) * PITS$A=BC
(01175) *
(01176) * END
(01177) *
(01178) * PITS$I DATA 6F'0.0'
...
(01179) * PITS$2=BL-PITS$

```

PAGE 29: [0000]C0CA16>00016M-MSD-2, 29-Dec-80 14:30:40, Ed: WOLF
 MWLQ6(Y,A,U,V,W) WEINER-LEVINSON MATRIX SOL'D & P=6 CODING/QUANT.

MWLQ6(Y,A,U,V,W) WEINER-LEVINSON MATRIX SOL'D & P=6 CODING/QUANT.

MWLQ6-APU PROGRAM

THE CHANGES FROM MWLQ IN 00016M-MSD ARE:

1. CODE/QUANTIZE 6 COEFFICIENTS INSTEAD OF 8.
2. THE QUANTIZED E-VALUES ARE REGULAR FLY PT, NOT 16-BIT FIXED.
3. THEREFORE THE THRESH/VALUE TABLES ARE FLOATING. SEPARATE, NOT INTERLEAVED, THRESH/VALUE TABLES ARE USED. THE VALUE TABLE IS THE K-DECODING TABLE.

WEINER LEVINSON DURHAM INVERSE (MWLF) FOLLOWED BY 6TH ORDER REFLECTION COEFFICIENT CODING AND QUANTIZATION (EQWAN)

MATHEMATICS

SEE J. MARKOW, LINEAR PREDICTION REVIEW
 PROC. IEEE, VOL 63, PPS66, APR 1975

BUFFER DEFINITIONS FOR THE MWLF PART:

V(K), AUTOCORRELATION COEFFICIENTS, $V(K)=E(K)$
 V(K) MUST BE COMPACT 32 BIT FLOATING POINT
 VSIZE NOT USED IN COMPUTATION, BUT
 VSIZE > USIZE FOR VALID RESULTS.

U(K), A(K) COEFFICIENTS
 U(K) MUST BE COMPACT 32 BIT FLOATING POINT
 U(K)=A(K-1), 16.A(0)=1 NOT IN BUFFER

V(K), PARTIAL CORRELATIONS AND ERROR
 V(K) MUST BE COMPACT 32 BIT FLOATING POINT
 VSIZE=2*USIZE
 V(K)=P(K-1), 0<K<USIZE
 V(K*USIZE)=E(K-1), USIZE<K<VSIZE

A, STABILITY TEST PARAMETER
 IF P(N)> CONTENTS OF (A), THEN
 FOR J>0
 A(N+J)=P(N+J)=0, E(N+J)=E(N)

FOR THE EQWAN PART, OUTPUT BUFFERS:
 THE W(K) BUFFER IS OVERWRITTEN BY THE QUANTIZED REFLECTION
 (PARTIAL CORRELATION) COEFFICIENTS.

W(K) = 6 CODED REFLECTION COEFFICIENTS. W IS A COMPACT LONG (16 BIT) FIXED POINT BUFFER.

MWLQ-APU INITIALIZATION

PAGE 301 (888D3C0C16)B0H16M, NS0.2, 29-Dec-88 14:30:40, Ed: WOLF
 WMLQ(V,A,U,V,U) WEINER-LEVINSON MATRIX SOL-M & P-6 CODING/QUANT.

		EVEN	DATA	WMLFSA	WMLFSSZ	START ON WORD BOUNDARY	START ADDRESS	SIZE
06014 0000	(01233)							
06015 0065	(01234)							
	(01235)							
	(01236)							
	(01237)							
	(01238)							
	(01239)							
	(01240)							
A00 06016 00P70000	(01241)	WMLFSA	MOV(IQA,A7)\NOP					
A01 06018 00C000C	(01242)		MOV(ZERO,M4)					
A02 0601A 00C0000	(01243)		MOV(IQ,M5)\NOP					
A03 0601C 00F50000	(01244)		MOV(IQA,A5)\NOP					
	(01245)							
	(01246)							
	(01247)							
	(01248)							
	(01249)							
	(01250)							
A04 0601E 2EA02E00	(01251)		RCP(A5)					
A05 06020 00P20000	(01252)		MOV(IQA,A2)\NOP					
A06 06022 00000000	(01253)		MOV(ZERO,M0)					
A07 06024 04000400	(01254)		MUL(M0,M4)					
A08 06026 02490249	(01255)		MOV(M1), R(A2)					
A09 06028 900A004F	(01256)		JUMPC(WMLFSA,AF2)					
	(01257)							
APA 0602A 90400010	(01258)		JUMPC(WMLFSE,AF3),SET					
	(01259)							
A0B 0602C 20202020	(01260)		CLEAR(AF3)					
	(01261)							
A0C 0602E 08EA0000	(01262)		MOV(IQA,M2)\NOP					
A0D 06030 00EF0000	(01263)		MOV(IQA,M6)\NOP					
A0E 06032 05500550	(01264)		MOV(A0), MUL(M2,M6)					
A0F 06034 42124212	(01265)		MOV(A2), ADD(A0,A2)					
	(01266)							
A10 06036 08E00000	(01267)	WMLFSE	MOV(IQA,M3)\NOP					
A11 06038 00EF0000	(01268)		MOV(IQA,M7)\NOP					
A12 0603A 05P105P1	(01269)		MOV(A1), MUL(M3,M7)					
A13 0603C 42242232	(01270)		MOV(A2), ADD(A1,A2)					
	(01271)							
A14 0603E 9016000C	(01272)		JUMPC(02,FV1)					
	(01273)							
A15 06040 00000000	(01274)		MOV(P,A0)					
A16 06042 42124212	(01275)		MOV(A2), ADD(A0,A2)					
	(01276)							
A17 06044 20372037	(01277)	WMLFRE	CLEAR(M1)					
	(01278)							
	(01279)							
	(01280)							
	(01281)							
	(01282)							
	(01283)							
	(01284)							
	(01285)							

(01200)	00000	00000	MUL(M1,M5)	E* RCP(E)
(01201)	00000	00000	MOV(M2), K(2)	M2=S(M)
(01202)	00000	00000	MOV(P,A0)	A0=E* RCP(E)
(01203)	00000	00000	MOV(A1), SUB(A1,A0)	A1=2
(01204)	00000	00000	MOV(M6), K(1)	M6=2-E* RCP(E)
(01205)	00000	00000	MUL(M1,M6)	
(01206)	00000	00000	MOV(P,M1)	M1=A4=EIR,
(01207)	00000	00000	MOV(A4), MUL(M1,M5)	EIR APPROX(1/E) TO 2*(-11)
(01208)	00000	00000	MOV(P,A0)	
(01209)	00000	00000	MOV(A0), SUB(A6,A0)	A6=1
(01210)	00000	00000	MOV(R,M6)	M6=1-EIR* E
(01211)	00000	00000	MUL(M1,M6)	
(01212)	00000	00000	MOV(P,A0)	A0=EIR(1-E* EIR)
(01213)	00000	00000	ADD(A0,A4)	
(01214)	00000	00000	MOV(R,M6)	R=1/(S(M-1), FULL PRECISION
(01215)	00000	00000	MUL(M2,M6)	
(01216)	00000	00000	MOV(P,A4)	
(01217)	00000	00000	R(A4)	
(01218)	00000	00000	MOV(P,M0)	M0=M4-A4=P(M)
(01219)	00000	00000	MOV(P,M4)	Q= P(M) (FIXED) = A(M,M)
(01220)	00000	00000	MOV(R,Q)\MOP	
(01221)	00000	00000		
(01222)	00000	00000		
(01223)	00000	00000		
(01224)	00000	00000		
(01225)	00000	00000		
(01226)	00000	00000		
(01227)	00000	00000		
(01228)	00000	00000		
(01229)	00000	00000		
(01230)	00000	00000		
(01231)	00000	00000		
(01232)	00000	00000		
(01233)	00000	00000		
(01234)	00000	00000		
(01235)	00000	00000		
(01236)	00000	00000		
(01237)	00000	00000		
(01238)	00000	00000		
(01239)	00000	00000		
(01240)	00000	00000		
(01241)	00000	00000		
(01242)	00000	00000		
(01243)	00000	00000		
(01244)	00000	00000		
(01245)	00000	00000		
(01246)	00000	00000		
(01247)	00000	00000		
(01248)	00000	00000		
(01249)	00000	00000		
(01250)	00000	00000		
(01251)	00000	00000		
(01252)	00000	00000		
(01253)	00000	00000		
(01254)	00000	00000		
(01255)	00000	00000		
(01256)	00000	00000		
(01257)	00000	00000		
(01258)	00000	00000		
(01259)	00000	00000		
(01260)	00000	00000		
(01261)	00000	00000		
(01262)	00000	00000		
(01263)	00000	00000		
(01264)	00000	00000		
(01265)	00000	00000		
(01266)	00000	00000		
(01267)	00000	00000		
(01268)	00000	00000		
(01269)	00000	00000		
(01270)	00000	00000		
(01271)	00000	00000		
(01272)	00000	00000		
(01273)	00000	00000		
(01274)	00000	00000		
(01275)	00000	00000		
(01276)	00000	00000		
(01277)	00000	00000		
(01278)	00000	00000		
(01279)	00000	00000		
(01280)	00000	00000		
(01281)	00000	00000		
(01282)	00000	00000		
(01283)	00000	00000		
(01284)	00000	00000		
(01285)	00000	00000		
(01286)	00000	00000		
(01287)	00000	0		

PAGE 33: CMMND3<DCAL16>BWM16M.NSO.2, 29-Dec-88 14:38:48, E4: WOLF
WMLQ6(Y,L,U,V,M) WEINER-LEVINSON MATRIX SOL'N & P-6 CODING/QUANT.

```

A37 06004 00000000 (01339) NOP
A38 06006 9016002F (01340) JUMPC(03,FH1)
A39 06008 20372037 (01341) *
A39 06008 20372037 (01342) WMLFAE CLEAR(VI)
A3A 0600A 04110411 (01343) MOV(A1), MUL(M0,M4)
A3B 0600C 433C0000 (01344) MOV(OQ), ADD(A1,A3)\NOP
(01345) *
(01346) *
(01347) * WMLF-APU CALCULATION OF E(M)
(01348) * E(M)=CL-P(M)P(M)E(M-1)
(01349) *
(01350) * REGISTER CONTENTS AT START
(01351) *
(01352) * M0=M4=A4=P(M), M5=A5=E(M-1), A6=1, A7=TEST
(01353) * P=P(M)P(M),R=P(M)=A(M,M)
(01354) * DATA SEQUENCE
(01355) * OTPT P(M), E(M)
A3C 0600E 00000000 (01356) AB=P(M)P(M)
A3C 0600E 40DC0000 (01357) OQ=A(M,M)
A3C 0600E 020A020A (01358) M2=1-P(M)P(M)
A3D 0600A 05200520 (01359) E(M-1)=CL-P(M)P(M)
A3E 06006 74FC0000 (01360) MOV(OQ), MAAABS(A7,A4)\NOP
A3F 06008 00000000 (01361) MOV(R,BULL)
A40 0600A 00C00000 (01362) MON(P,OQ)\NOP
A41 0600C 20092009 (01363) SETTC(AP1)
A42 0600E 00050005 (01364) MOV(P,A5)
A43 0600A 00A000A0 (01365) MON(P,M5)
A44 0600C 20402040 (01366) SET(AF0)
A45 0600A 91100040 (01367) JUMPS(WMLFD,P1)
A46 0600A 901E0004 (01368) *
A47 0600A 901E0004 (01369) *
A48 0600A 901E0004 (01370) *
A49 0600A 901E0004 (01371) *
A49 0600A 901E0004 (01372) * WMLF-APU ABORT ROUTINE
A4A 0600A 901E0004 (01373) * P REGISTER=E(M)
A4B 0600C 00C00000 (01374) *
A4C 0600E 901C0049 (01375) *
A4C 0600E 901C0049 (01376) #4
A4D 0600A 901C0000 (01377) MOV(ZERO,OQ)\NOP
A4E 0600C 00C00000 (01378) MOV(ZERO,OQ)\NOP
A4E 0600C 00C00000 (01379) MOV(P,OQ)\NOP
A4E 0600C 00C00000 (01380) JUMPC(04,EO)
A4E 0600C 20322032 (01381) WMLFD CLEAR(R1)
A4E 0600C 10000053 (01382) JUMP(RQUAN)
A4E 0600C 10000053 (01383) *
A4F 0600A 9016004F (01384) *
A4F 0600A 9016004F (01385) JUMPC(WMLFRM,FH1)
A50 0600C 10000017 (01386) JUMP(WMLFRE)
A51 0600B 00E00000 (01387) WMLFAV MOV(IGA,MULL)\NOP
A52 0600A 10000039 (01388) JUMP(WMLFAE)
A52 0600A 10000039 (01389) *
A52 0600A 10000039 (01390) *
A52 0600A 10000039 (01391) *

```

A(M+J)=B
 P(M+J)=B
 E(M+J)=E(M)

JAPU HALTS HERE AT END OF WMLD,
 ; THEN GOES TO EQUAN WHEN RESTARTED
 ; BY THE APS PROGRAM

LET APS/OTPT GO
 GO BACK UNLESS DONE

PAGE 33: (R0WD1)<DC116>B0N1CH.MSD.2, 29-Dec-88 14130140, ED: WOLF
MULQC(V,A,W,V,W) WEINER-LEVINSON MATRIX SOL-W & P-6 CODING/QUANT.

```

(01392) )
(01393) )QUANT: QUANTIZE AND CODE 6 REFLECTION COEFFICIENTS
(01394) ) J. WOLF 7/9/88
(01395) )
(01396) )INPUT AND OUTPUT STREAMS:
(01397) )IQR: 2**15,
(01398) ) FOR EACH COEFF: K(1), (CTH(K(N),DVL(K(N)), N=1,...,LENGTH(IW1)
(01399) ) APS SETS APS ON LAST K(1) (EO WILL BE TWO LATER).
(01400) )QOI: (FLY PT VALUE(1), FID PT CODE(1), I=1,...,6)
(01401) )
(01402) )TABLE LENGTHS FOR K(1) ARE 64, 32, 16, 16, 16, 16
(01403) )APU REGISTER USAGE: CODE CTR
(01404) ) A0: K(1) 2**15
(01405) ) A1: CTH(K(N)
(01406) ) A2: DVL(K(N)
(01407) )
(01408) )QUANT: NOP \ MOV(IQR,A1)
(01409) )LOOP FOR EACH NEW COEFFICIENT TO BE QUANTIZED/CODED
(01410) )I: MOV(IQR,A0) \ MOV(ZERO,A0)
(01411) )NOP \ R(A0)
(01412) )INNER LOOP
(01413) )I2: MOV(IQR,A1) \ NOP
(01414) )SUB(AB,A1) \ MOV(AB,A0),ADD(AB,A1)
(01415) )MOV(IQR,A2) \ NOP
(01416) )MOV(R,NOLL) \ NOP
(01417) )JUMPS(RQSDONE,FWI)
(01418) )JUMPC(R2,I1)
(01419) )FOUND THE RIGHT QUANTIZATION BIN. NOW READ IN THE REST OF TABLE
(01420) )I3: MOV(IQR,NOLL) \ NOP
(01421) )NOP
(01422) )JUMPC(R3,FWI)
(01423) )RQSDONE: CLEAR(FWI)
(01424) )R(I2) \ ALIGN(AB)
(01425) )MOV(R,OQ) \ MOV(R,OQ)
(01426) )JUMPC(I1,AF0)
(01427) )CLEAR(RA)
(01428) )JUMP(0)
(01429) )
(01430) )MMLF$SZ=8A-MWLP$SA
(01431) )
(01432) )
(01433) )
(01434) )
(01435) )
(01436) )
(01437) )
(01438) )
(01439) )
(01440) )
(01441) )
(01442) )
(01443) )
(01444) )
(01445) )
(01446) )
(01447) )
(01448) )
(01449) )
(01450) )
(01451) )
(01452) )
(01453) )
(01454) )
(01455) )
(01456) )
(01457) )
(01458) )
(01459) )
(01460) )
(01461) )
(01462) )
(01463) )
(01464) )
(01465) )
(01466) )
(01467) )
(01468) )
(01469) )
(01470) )
(01471) )
(01472) )
(01473) )
(01474) )
(01475) )
(01476) )
(01477) )
(01478) )
(01479) )
(01480) )
(01481) )
(01482) )
(01483) )
(01484) )
(01485) )
(01486) )
(01487) )
(01488) )
(01489) )
(01490) )
(01491) )
(01492) )
(01493) )
(01494) )
(01495) )
(01496) )
(01497) )
(01498) )
(01499) )
(01500) )
(01501) )
(01502) )
(01503) )
(01504) )
(01505) )
(01506) )
(01507) )
(01508) )
(01509) )
(01510) )
(01511) )
(01512) )
(01513) )
(01514) )
(01515) )
(01516) )
(01517) )
(01518) )
(01519) )
(01520) )
(01521) )
(01522) )
(01523) )
(01524) )
(01525) )
(01526) )
(01527) )
(01528) )
(01529) )
(01530) )
(01531) )
(01532) )
(01533) )
(01534) )
(01535) )
(01536) )
(01537) )
(01538) )
(01539) )
(01540) )
(01541) )
(01542) )
(01543) )
(01544) )
(01545) )
(01546) )
(01547) )
(01548) )
(01549) )
(01550) )
(01551) )
(01552) )
(01553) )
(01554) )
(01555) )
(01556) )
(01557) )
(01558) )
(01559) )
(01560) )
(01561) )
(01562) )
(01563) )
(01564) )
(01565) )
(01566) )
(01567) )
(01568) )
(01569) )
(01570) )
(01571) )
(01572) )
(01573) )
(01574) )
(01575) )
(01576) )
(01577) )
(01578) )
(01579) )
(01580) )
(01581) )
(01582) )
(01583) )
(01584) )
(01585) )
(01586) )
(01587) )
(01588) )
(01589) )
(01590) )
(01591) )
(01592) )
(01593) )
(01594) )
(01595) )
(01596) )
(01597) )
(01598) )
(01599) )
(01600) )
(01601) )
(01602) )
(01603) )
(01604) )
(01605) )
(01606) )
(01607) )
(01608) )
(01609) )
(01610) )
(01611) )
(01612) )
(01613) )
(01614) )
(01615) )
(01616) )
(01617) )
(01618) )
(01619) )
(01620) )
(01621) )
(01622) )
(01623) )
(01624) )
(01625) )
(01626) )
(01627) )
(01628) )
(01629) )
(01630) )
(01631) )
(01632) )
(01633) )
(01634) )
(01635) )
(01636) )
(01637) )
(01638) )
(01639) )
(01640) )
(01641) )
(01642) )
(01643) )
(01644) )
(01645) )
(01646) )
(01647) )
(01648) )
(01649) )
(01650) )
(01651) )
(01652) )
(01653) )
(01654) )
(01655) )
(01656) )
(01657) )
(01658) )
(01659) )
(01660) )
(01661) )
(01662) )
(01663) )
(01664) )
(01665) )
(01666) )
(01667) )
(01668) )
(01669) )
(01670) )
(01671) )
(01672) )
(01673) )
(01674) )
(01675) )
(01676) )
(01677) )
(01678) )
(01679) )
(01680) )
(01681) )
(01682) )
(01683) )
(01684) )
(01685) )
(01686) )
(01687) )
(01688) )
(01689) )
(01690) )
(01691) )
(01692) )
(01693) )
(01694) )
(01695) )
(01696) )
(01697) )
(01698) )
(01699) )
(01700) )
(01701) )
(01702) )
(01703) )
(01704) )
(01705) )
(01706) )
(01707) )
(01708) )
(01709) )
(01710) )
(01711) )
(01712) )
(01713) )
(01714) )
(01715) )
(01716) )
(01717) )
(01718) )
(01719) )
(01720) )
(01721) )
(01722) )
(01723) )
(01724) )
(01725) )
(01726) )
(01727) )
(01728) )
(01729) )
(01730) )
(01731) )
(01732) )
(01733) )
(01734) )
(01735) )
(01736) )
(01737) )
(01738) )
(01739) )
(01740) )
(01741) )
(01742) )
(01743) )
(01744) )
(01745) )
(01746) )
(01747) )
(01748) )
(01749) )
(01750) )
(01751) )
(01752) )
(01753) )
(01754) )
(01755) )
(01756) )
(01757) )
(01758) )
(01759) )
(01760) )
(01761) )
(01762) )
(01763) )
(01764) )
(01765) )
(01766) )
(01767) )
(01768) )
(01769) )
(01770) )
(01771) )
(01772) )
(01773) )
(01774) )
(01775) )
(01776) )
(01777) )
(01778) )
(01779) )
(01780) )
(01781) )
(01782) )
(01783) )
(01784) )
(01785) )
(01786) )
(01787) )
(01788) )
(01789) )
(01790) )
(01791) )
(01792) )
(01793) )
(01794) )
(01795) )
(01796) )
(01797) )
(01798) )
(01799) )
(01800) )
(01801) )
(01802) )
(01803) )
(01804) )
(01805) )
(01806) )
(01807) )
(01808) )
(01809) )
(01810) )
(01811) )
(01812) )
(01813) )
(01814) )
(01815) )
(01816) )
(01817) )
(01818) )
(01819) )
(01820) )
(01821) )
(01822) )
(01823) )
(01824) )
(01825) )
(01826) )
(01827) )
(01828) )
(01829) )
(01830) )
(01831) )
(01832) )
(01833) )
(01834) )
(01835) )
(01836) )
(01837) )
(01838) )
(01839) )
(01840) )
(01841) )
(01842) )
(01843) )
(01844) )
(01845) )
(01846) )
(01847) )
(01848) )
(01849) )
(01850) )
(01851) )
(01852) )
(01853) )
(01854) )
(01855) )
(01856) )
(01857) )
(01858) )
(01859) )
(01860) )
(01861) )
(01862) )
(01863) )
(01864) )
(01865) )
(01866) )
(01867) )
(01868) )
(01869) )
(01870) )
(01871) )
(01872) )
(01873) )
(01874) )
(01875) )
(01876) )
(01877) )
(01878) )
(01879) )
(01880) )
(01881) )
(01882) )
(01883) )
(01884) )
(01885) )
(01886) )
(01887) )
(01888) )
(01889) )
(01890) )
(01891) )
(01892) )
(01893) )
(01894) )
(01895) )
(01896) )
(01897) )
(01898) )
(01899) )
(01900) )
(01901) )
(01902) )
(01903) )
(01904) )
(01905) )
(01906) )
(01907) )
(01908) )
(01909) )
(01910) )
(01911) )
(01912) )
(01913) )
(01914) )
(01915) )
(01916) )
(01917) )
(01918) )
(01919) )
(01920) )
(01921) )
(01922) )
(01923) )
(01924) )
(01925) )
(01926) )
(01927) )
(01928) )
(01929) )
(01930) )
(01931) )
(01932) )
(01933) )
(01934) )
(01935) )
(01936) )
(01937) )
(01938) )
(01939) )
(01940) )
(01941) )
(01942) )
(01943) )
(01944) )
(01945) )
(01946) )
(01947) )
(01948) )
(01949) )
(01950) )
(01951) )
(01952) )
(01953) )
(01954) )
(01955) )
(01956) )
(01957) )
(01958) )
(01959) )
(01960) )
(01961) )
(01962) )
(01963) )
(01964) )
(01965) )
(01966) )
(01967) )
(01968) )
(01969) )
(01970) )
(01971) )
(01972) )
(01973) )
(01974) )
(01975) )
(01976) )
(01977) )
(01978) )
(01979) )
(01980) )
(01981) )
(01982) )
(01983) )
(01984) )
(01985) )
(01986) )
(01987) )
(01988) )
(01989) )
(01990) )
(01991) )
(01992) )
(01993) )
(01994) )
(01995) )
(01996) )
(01997) )
(01998) )
(01999) )
(02000) )

```


PAGE 361 (00003)DCAL16>00016H.MSO.2, 29-Dec-88 14:30:40, Ed: WOLF
MWLQ6-APS PROGRAM

```

A34 06150 00500000 (01539)
A35 06152 61200000 (01540)
A36 06154 61200001 (01541)
A37 06156 61200002 (01542)
A38 06158 70010020 (01543)
A39 0615A 70000020 (01544)
A3A 0615C 700030A0 (01545)
A3B 0615E 702030C0 (01546)
A3C 06160 700040A9 (01547)
A3D 06162 70200031 (01548)
A3E 06164 70000020 (01549)
A3F 06166 70300010 (01550)
A40 06168 00200010 (01551)
A41 0616A 02300022 (01552)
A42 0616C 04300031 (01553)
A43 0616E 06401020 (01554)
A44 06170 00700000 (01555)
A45 06172 00200000 (01556)
A46 06174 00014000 (01557)
A47 06176 000A0002 (01558)
A48 06178 00210022 (01559)
A49 0617A 020A0002 (01560)
A4A 0617C 90100020 (01561)
A4B 0617E 9030407A (01562)
A4C 06180 90200030 (01563)
A4D 06182 90000020 (01564)
A4E 06184 90207040 (01565)
A4F 06186 90300030 (01566)
A50 06188 00106640 (01567)

LOAD(BW1,MSS)
SUB(BW2,MSS)
ADD(BW1,1)
ADDB(BW2,BW0,TP)
ADDB(BW2,BW1,TP)
NOP(0)
JUMPC(B5,AP0)
JUMP(BA+1,AP0),CLEAR
JUMPC(MWLOC,AP1)

CLEAR(R1)
NOP(0)
MOV8(BR3,BW0)
MOV8(BR2,BW0)
SUB8(BR3,BW1)
SUB8(BR2,BW1)
SWAP(BR3,1)

LOAD(BW0,C13)
LOAD(BW3,MSS)
SUB(BW0,MSS)
ADDB(BW0,BR2),JUMP(MWLOC)
ADD(BW0,2,TP)
SUB8(BW2,BW1)
ADD(BW2,2,TP)
ADDB(BW2,BW1,TP)
ADDB(BR3,2),JUMP(B6)

CLEAR(R0)
NOP(0)

JAPS PROGRAM FOR QUANTIZATION AND CODING. THIS IS STARTED
BY THE MWLQ6SSM SPECIAL SUPPORT MODULE AFTER MWLQ FINISHES.
J.J.WOLF, 7/8/88

INPUT AND OUTPUT STREAMS:
2*-15,
FOR EACH COEFF: K(1), (CTHET(K),DVLK(K), N=1,...,LENGTH(MW1)
APS SETS AFB ON LAST K(1) (EO WILL BE TOO LATE).
(FLT PT VALUE(1), FID PT CODE(1), 1=1,...,6)

INPUT PROGRAM
BR0: V(1) = UNQUANTIZED K(1)
BR1: THRESH(N)
BR2: VALUE(N)
BR3: TABLE LENGTH COUNTER

JSH(KQ$0,P2)
SET(R0)
JSH(KQ$SUB,P1)

;SET AND START OUTPUT PROGRAM
;SET UP P1 TO USE KQ$SUB AS A SUBROUTINE

```

HWL96-APS PROGRAM

176

```

00000100 (01648) *
          (01649) *
          MULPSA=0C
          (01650)
          (01651) *
          END
00102 (01652)
          (01653) *
          (01654) *
          (01655) *
          (01656) MULPSI DATA 0P"0.0"
          ...
          (01657) *
          (01658) MULPS2=0L-MULPSAPS
          (01659)

```



```

(01678) * MEWL(V,I,U,V,W)
(01679) * CORRELATION MATRIX EQUATION, WEINER-LEVINSON SOLUTION
(01680) *
(01681) * SEPTEMBER 14, 1979
(01682) *
(01683) * BUFFER USAGE
(01684) *
(01685) * V(K), OUTPUT VECTOR
(01686) * W(K), INPUT VECTOR
(01687) * V(K)=R(K)-R(K), CORRELATION MATRIX ELEMENTS
(01688) * W(K), WORKING BUFFER
(01689) * (I), (I-1), INTEGER TABLE OUTPUTS
(01690) * (N); THRESHOLD, EARLY TERMINATION TEST.
(01691) *
(01692) * FUNCTION W(K)=SUM J=0,N-13 OF R(K-J)*V(J)
(01693) * (I)=Y2-1-N,
(01694) * (I+1)=E(N)/R(N), RELATIVE ENERGY, "SPIKE SOLUTION"
(01695) * IF COMPLETED SUCCESSFULLY, N=Y2-1, I.E. (I)=0
(01696) *
(01697) * ALGORITHM
(01698) * DETERMINES SOLUTION BY FORWARD INTERATION
(01699) * MAKES USE OF THE SPIKE CASE, I.E.
(01700) * -R(K+1)=SUM FOR J=0,N-13 OF R(K-J)*A(N-1,J)
(01701) *
(01702) * INITIALIZATION: E(-1)=R(0)
(01703) * RECURSION FORMULA
(01704) *
(01705) * B(N)=R(N+1)+SUM J=0,N-13 OF R(N-J)*A(N-1,J)
(01706) * C(N)=U(N)-SUM J=0,N-13 OF R(N-J)*V(N-1,J)
(01707) *
(01708) * KB(N), REFLECTANCE, =R(N)/E(N-1)
(01709) * KC(N)=C(N)/E(N-1)
(01710) * E(N), ENERGY, SPIKE CASE, = E(N-1)-B(N)*KB(N)
(01711) *
(01712) * V(N,J)=V(N-1,J)+KC(N)*A(N-1,N-1-J) V(N,N)=EC(N)
(01713) * A(N,J)=A(N-1,J)-KB(N)*A(N-1,N-1-J) A(N,N)=E(N)
(01714) *
(01715) * THE EXTERNAL VALUES
(01716) * A(N,-1)=1, A(N,J)=V(N,J)=0, J>N
(01717) * ARE CONSISTENT WITH THE RECURSION FORMULA
(01718) *
(01719) * THE ITERATION NORMALLY TERMINATES WITH N=Y2-1
(01720) * EARLY TERMINATION OCCURS IF (0)*MAC (R(N)) > E(N-1)
(01721) * IN SUCH CASE UNUSED VALUES OF A,K0, AND Y ARE SET TO ZERO
(01722) *
(01723) * BUFFER USAGE
(01724) *
(01725) * V(K)=V(N,K)
(01726) * W(K)=A(N,K), 0<=K<N
(01727) * W(N+E)=KB(E), 0<=E<N
(01728) * NORMAL TERMINATION, N=Y2-1
(01729) * EARLY TERMINATION, UNUSED VALUES ZERO FILLED
(01730) * (I)=Y2-1-N, N=Y2-1, NORMAL TERMINATION
(01731) * (I+1)=E(N)/R(N)=E(N)/E(-1), FRACTIONAL ENERGY

```

PAGE 41: (0000) <0C16>00H16M.H50.2, 29-Dec-88 14:38:48, Ed: WOLF
NEWL(Y,I,W,D,V,U)

181

PAGE 43: (BMM030C0A16)B0016M.MS0.2, 29-Dec-88 14130140, Ed: WOLF
MEWL - APB PROGRAM

```

A17 06234 04000000 (01704) MOV(A0),MWL(M1,M5) \ NOP JAB=A(M-1,J)*E(M-J)
      (01705) JUMPC(B2,FV1) LOOP UNLESS DONE
A18 06236 90160010 (01706) CLRZ(A0)
A19 06238 20372037 (01707) MEWL3:
      (01708) MOV(A4),ADD(A0,A5) \ NOP J44=CSUM
      (01709) MOV(P,A1) \ NOP J41=Y(M-1,N-1)*R(1)
      (01710) MOV(A5),SUB(A4,A1) \ NOP J45=B(M)
      (01711) MOV(A4),R(A6) \ NOP J44=C(M), SET REGC=E(M-1)
      (01712)
      (01713)
      (01714)
      (01715)
      (01716)
      (01717) REFLECTANCE CALCULATION AND EARLY TERMINATION TEST
      (01718) FUNCTION
      (01719) IF (0)*ABS B(M) > E(M-1), EARLY TERMINATION, SET AP2
      (01720)
      (01721) A7=A7-1, DECREMENT COUNT
      (01722)
      (01723)
      (01724)
      (01725)
      (01726)
      (01727)
      (01728)
      (01729)
      (01730)
      (01731)
      (01732)
      (01733)
      (01734)
      (01735)
      (01736)
      (01737)
      (01738)
      (01739)
      (01740)
      (01741)
      (01742)
      (01743)
      (01744)
      (01745)
      (01746)
      (01747)
      (01748)
      (01749)
      (01750)
      (01751)
      (01752)
      (01753)
      (01754)
      (01755)
      (01756)
      (01757)
      (01758)
      (01759)
      (01760)
      (01761)
      (01762)
      (01763)
      (01764)
      (01765)
      (01766)
      (01767)
      (01768)
      (01769)
      (01770)
      (01771)
      (01772)
      (01773)
      (01774)
      (01775)
      (01776)
      (01777)
      (01778)
      (01779)
      (01780)
      (01781)
      (01782)
      (01783)
      (01784)
      (01785)
      (01786)
      (01787)
      (01788)
      (01789)
      (01790)
      (01791)
      (01792)
      (01793)
      (01794)
      (01795)
      (01796)
      (01797)
      (01798)
      (01799)
      (01800)
      (01801)
      (01802)
      (01803)
      (01804)
      (01805)
      (01806)
      (01807)
      (01808)
      (01809)
      (01810)
      (01811)
      (01812)
      (01813)
      (01814)
      (01815)
      (01816)
      (01817)
      (01818)
      (01819)
      (01820)
      (01821)
      (01822)
      (01823)
      (01824)
      (01825)
      (01826)
      (01827)
      (01828)
      (01829)
      (01830)
      (01831)
      (01832)
      (01833)
      (01834)
      (01835)
      (01836)

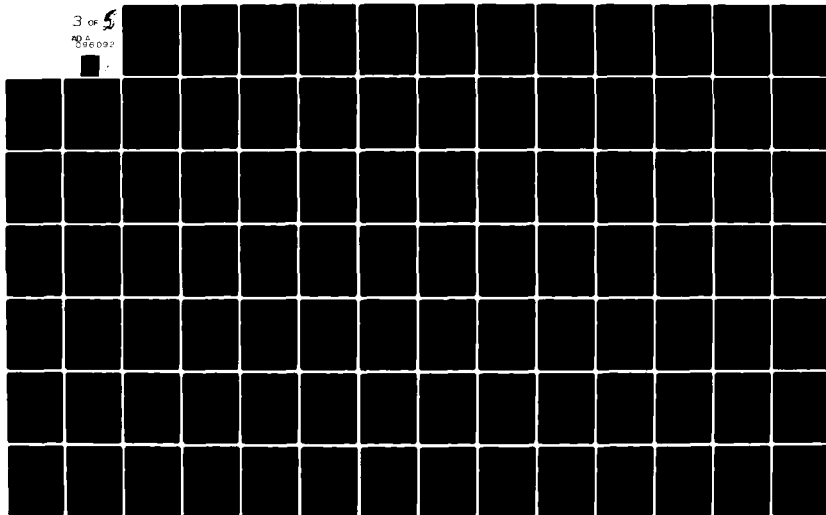
```


AD-A096 092

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 J J WOLF, K D FIELD, W H RUSSELL
DCA100-79-C-0037
NL

UNCLASSIFIED

3 OF 5
NO 5
096092



```

(01898) ) A(M,N)=K(N) Y(M,N)=K(N),
(01899) ) WERE EXECUTED DURING REFLECTANCE CALCULATION
(01900) ) A(M,J)=A(M-1,J)-K(N)*A(M-1,N-1-J)
(01901) ) A(M,N-1-J)=A(M-1,N-1-J)-K(N)*A(M-1,J)
(01902) ) Y(M,J)=Y(M-1,J)+K(N)*A(M-1,N-1-J)
(01903) ) Y(M,N-1-J)=Y(M-1,N-1-J)+K(N)*A(M-1,J)
(01904) ) AS INDICATED, RECURSION WORKS WITH ENDS TOWARDS MIDDLE
(01905) ) FOR N ODD, RECURSION IS DUPLICATED ON J=(N+1)/2 VALUE
(01906) ) SEQUENCING
(01907) ) A(M-1,N-1),A(M-1,N), JUMP TO FIRST Y INPUT IN LOOP
(01908) ) OUT A(M,N-1),A(M-1,N-1-J),OUT Y(M),A(M-1,J),
(01909) ) OUT Y(M,N-1),Y(M-1,J),OUT A(M,N),Y(M-1,N-1-J), FWI
(01910) ) OUT A(M,N-1-N/2), OUT Y(M,N/2), OUT Y(M,N-1-N/2)
(01911) ) REGISTER CONTENTS
(01912) ) M4=K(N), M5=K(N)
(01913) ) A6=E(N), A7=YCZ-N-1
(01914) ) MOV(IQ,M8) \ NOP
(01915) ) MUL(M8,M4) \ NOP
(01916) ) MOV(IQ,A8) \ NOP
(01917) ) MOV(IQ,M1) \ NOP
(01918) ) MOV(A5),MUL(M1,M4) \ NOP
(01919) ) MOV(IQ,A1) \ NOP
(01920) ) SUB(A1,A5) \ NOP
(01921) ) JUMP(NEWL6)
(01922) ) MOV(OQ),ADD(A2,A5) \ NOP
(01923) ) MOV(IQ,M8) \ NOP
(01924) ) MOV(A4),MUL(M8,M4) \ NOP
(01925) ) MOV(IQ,A8) \ NOP
(01926) ) MOV(OQ),ADD(A3,A4) \ NOP
(01927) ) MOV(IQ,M1) \ NOP
(01928) ) MOV(A5),MUL(M1,M4) \ NOP
(01929) ) MOV(IQ,A1) \ NOP
(01930) ) MOV(OQ),SUB(A6,A4) \ NOP
(01931) ) MOV(IQ,A3) \ NOP
(01932) ) MOV(A5),MUL(M1,M5) \ NOP
(01933) ) JUMP(C15,FWI)
(01934) ) CLEAR(VI)
(01935) ) MOV(OQ),ADD(A2,A5) \ NOP
(01936) ) MOV(P,A4) \ NOP
(01937) ) JQ=A(N,M-1-N/2)
(01938) )
(01939) )
(01940) )
(01941) )
(01942) )

```


188

PAGE 581 (0000)C0C16>00016M.MSO.2, 29-Dec-88 14:30:40, Ed: WOLF
NEWL - APS PROGRAM

```

(02135) ,
A3P 06394 7F410013      MOVW(BW0,BR1,TP)      ;BW0=A(N-N/2), OUTPUT
A40 06396 80010015      MOVW(BW0,BR2,TP)      ;BW0=Y(N/2-1), OUTPUT
A41 06398 02010017      MOVW(BW0,BR3,TP)      ;BW0=Y(N-N/2), OUTPUT
A42 0639A 04000A60      JUMP(NEWL15)
(02141) ,
(02142) ,
(02143) ,
(02144) ,
(02145) ,
(02146) ,
(02147) ,
(02148) ,
(02149) ,
(02150) ,
A43 0639C 06910000      NEWL16:00(BR1,N,TP)      ;BR0=R(0), INPUT
A44 0639E 0852630A      LOAD(BR1,IT(1),L)      ;BR1=1 ADDRESS
A45 063A0 00010013      MOVW(BW0,BR1,TP)      ;OUTPUT (1)=YZ-1-N
(02154) ,
A46 063A2 0C70001C      NEWL17:LOAD(BR3,2*YZ)      ;BR3=2*YZ
A47 063A4 0E390026      SUBW(BR3,BW3)          ;BR3=2*(YZ-N-1)
(02157) ,
(02158) ,
(02159) ,
A48 063A6 9009002E      ADDW(BR0,BW3)          ;BR0=A(N-1)
A49 063A8 9229002E      ADDW(BR2,BW3)          ;BR2=Y(N-1)
A4A 063AA 94110011      MOVW(BW1,BR0)          ;BW1=A(N-1)
A4B 063AC 96310015      MOVW(BW3,BR2)          ;BW3=Y(N-1)
A4C 063AE 99015079      ADDL(BW0,1,TP),JUMP(NEWL19) ;OUT (1+1)=E(N)/R(N)
(02166) ,
A4D 063B0 9A91003A      ADDL(BW1,2,TP)          ;ZERO FILL A(N-1+J)
A4E 063B2 9CA1003A      ADDL(BW2,2,TP)          ;ZERO FILL R0(N-1+J)
A4F 063B4 9EB1003A      ADDL(BW3,2,TP)          ;ZERO FILL Y(N-1+J)
A50 063B6 A0394002      NEWL19:SUBL(BR3,2),JUMPP(R10)
(02171) ,
A51 063B8 A2200031      CLEAR(R1)
A52 063BA A4000020      NOP(0)
(02174) ,
(02175) ,
(02176) ,
(02177) ,
(02178) ,
(02179) ,
063BC 00000000      NEWL5C=BC
063BC 000000A6      END
063BC 000000A6      NEWL5Z=BL-NEWL5
063BC 000000A6      NEWL11=BL
063BC 000000A6      DATA $0001

```

```

(02180) * SBMSMEVL - SPECIAL SUPPORT ROUTINE FOR MEVL
(02181) ;
(02182) ;
063BD C63408E0 SBMSMEVL PUSHMIL R3,AFDT$ORC(R2)      STACK APU MODULE BUS ORIGIN
063BF 9071FFFE MOVIR R7,-4$
063C1 C607FFFE PUSHMIL R3,0-W$(R3)                    STACK APU MODULE ST ADDR AND SZ
063C3 C63408E2 PUSHMIL R3,AFDT$ORC+W$(R2)              STACK APS MODULE BUS ORIGIN
063C5 C63408E6 PUSHMIL R3,AFDT$ORC+2*W$(R2)              STACK CSPU SUPPORT MODULE
063C7 2771     DECR R7,1
(02188) ;
(02189) ;
063C9 C407FFFC EVEN
063CA 9047FFFD PUSHMIL R3,0-2*W$(R3)                    APS MODULE SIZE
063CB 9047FFFD MOVIR R4,-3*W$(R3)                      GET SPECIAL SUPPORT SEMAPHORE
063CC 7040FF70 MOVIR R4,R4,MSK$LOVT
063CE 3C40     LRS R4,8
063CF 3430     PUSHMIL R3,R4
063D0 4052     MOVIR R5,R1
063D1 2651     INCR R5,W$
063D2 704100FF MOVIR R4,R5,MSK$RBYT
063D4 3630     PUSHMIL R3,R4
063D5 C6306792 PUSHMIL R3,CLR$C0G1
063D7 2632     INCR R3,2
063D8 C630678A PUSHMIL R3,ZERO
063DA 9F306011 SUBIR R3,AP$LL0C
(02203) ;
(02204) ;
063DC F0420002 * SCALAR BINDING
063DE 914000FF MOVIR R4,2*W$(R1)
063E0 3A41     ANDIR R4,MSK$RBYT
063E1 9C400302 LLS R4,1
063E3 E0406351 ADDIR R4,SVTS
063E5 F0420001 MOVIR R4,NEWL$+W$-NEWL$+11+W$ ;STORED APS INSTR IN MAIN MEM
063E7 9A4000FF INTEGER TABLE BINDING (SCALAR A)
063E9 9C400502 MOVIR R4,W$(R1)
063EB E040639F ANDIR R4,MSK$RBYT
063ED F0620001 ADDIR R4,TSVTS
063EF 506CFF00 MOVIR R4,NEWL$+W$-NEWL$+16+3*W$ ;STORED APS INSTR IN MAIN MEM
063F1 3C67     MOVIR R6,W$(R1)
063F2 C04C0502 YBASE BINDING
063F4 F0706330 MOVIR R6,R6,MSK$LOVT
063F6 564FFFF0 LRS R6,7
063F8 04406330 EVEN
063FA F04C0605 MOVIR R4,ACT$BA(R6)
063FC 2641     MOVIR R7,NEWL$+W$-NEWL$+7
063FD E0406330 INCR R4,R7,SVTS
063FF 04406330 MOVIR R4,NEWL$+W$-NEWL$+7
(02227) ;
(02228) ;
063FA F04C0605 * VZ BINDING
063FC 2641     MOVIR R4,W$+BCT$AD(R6)
063FD E0406330 INCR R4,1
063FF 3A41     MOVIR R4,NEWL$+W$-NEWL$+11+W$
(02232) ;
(02233) ;
063FA F04C0605 ;R4=BUFFER SIZE-1
063FC 2641     ;R4=BUFFER SIZE
063FD E0406330 ;R4=R4-2
063FF 3A41

```

PAGE 52: (00NDJ)DCALC080816M.NSO.2, 29-Dec-00 14130140, Ed: WOLF
SOM\$MENL - SPECIAL SUPPORT ROUTINE FOR MENL

06400 00406325 (02233)	MOVBN R4,MENL\$+W\$*MENLA4+H\$		
06402 00406357 (02234)	MOVBN R4,MENL\$+W\$*MENLA13+H\$		
06404 00406313 (02235)	MOVBN R4,MENL\$+W\$*MENLA17+H\$		
06406 00620002 (02237)	MOVNR R6,2*H\$(R1)	JGET U ID	
06408 006CFF00 (02238)	MOVNR R6,R6,MSK\$LOVT		
0640A 3C67 (02239)	LRS R6,7		
0640B 0000 (02240)	EVEN		
0640C C04C0502 (02241)	MOVNRL R4,BCISBA(R6)		
0640E F070632A (02242)	MOVNR R7,MENL\$+W\$*MENLAB	JGET 17-BIT VBASE ADDR	
06410 564EFFF0 (02243)	IORER R4,R7,\$FFFF		
06412 0440633A (02244)	MOVNRL R4,MENL\$+W\$*MENLAB		
06414 F0620003 (02247)	MOVNR R6,3*H\$(R1)	JGET VBASE ID	
06416 506CFF00 (02248)	MOVNR R6,R6,MSK\$LOVT		
06418 3C67 (02249)	LRS R6,7		
06419 0000 (02250)	EVEN		
0641A C04C0502 (02251)	MOVNRL R4,BCISBA(R6)		
0641C F0706320 (02252)	MOVNR R7,MENL\$+W\$*MENLA2		
0641E 564EFFF0 (02253)	IORER R4,R7,\$FFFF		
06420 04406320 (02254)	MOVNRL R4,MENL\$+W\$*MENLA2		
06422 C04C0502 (02255)	MOVNRL R4,BCISBA(R6)		
06424 F070632E (02256)	MOVNR R7,MENL\$+W\$*MENLA6		
06426 564EFFF0 (02258)	IORER R4,R7,\$FFFF	JGET W ID	
06428 0440632E (02259)	MOVNRL R4,MENL\$+W\$*MENLA6	JGET PROPER BYTE	
0642A F0620004 (02261)	MOVNR R6,4*H\$(R1)		
0642C 506CFF00 (02262)	MOVNR R6,R6,MSK\$LOVT		
0642E 3C67 (02264)	LRS R6,7		
0642F 0000 (02265)	EVEN		
06430 C04C0502 (02266)	MOVNRL R4,BCISBA(R6)	JGET 17-BIT VBASE	
06432 F0706322 (02267)	MOVNR R7,MENL\$+W\$*MENLA3		
06434 564EFFF0 (02268)	IORER R4,R7,\$FFFF		
06436 04406322 (02269)	MOVNRL R4,MENL\$+W\$*MENLA3		
06438 C04C0502 (02270)	MOVNRL R4,BCISBA(R6)		
0643A F070632A (02271)	MOVNR R7,MENL\$+W\$*MENLA5		
0643C 564EFFF0 (02273)	IORER R4,R7,\$FFFF		
0641E 0440632A (02274)	MOVNRL R4,MENL\$+W\$*MENLA5		
06440 00000F63 (022A3)	JMP AP\$BND00		

```

(02284) * APU3 - STAB(Y,A,U,V)
(02285) * WNR 0/00
(02286) *
(02287) * ROUTINE TO CHECK STABILITY OF
(02288) * PITCH PREDICTOR COEFFICIENTS
(02289) * ALSO CODES AND QUANTIZES COEFF'S
(02290) *
(02291) * THE INPUT COEFFICIENTS ARE : V(0),V(1),V(2)
(02292) * THEY ARE LINEARLY TRANSFORMED AS:
(02293) *
(02294) * T1=V(0)+V(1)+V(2)
(02295) * T2=V(0)-2*V(1)+V(2)
(02296) * T3=V(0)-V(2)
(02297) *
(02298) * THEN , IF (T1<0 .OR.
(02299) * T2<0 .OR.
(02300) * T3<0 .OR.
(02301) * T2>0 .OR.
(02302) * T3>0 )
(02303) * THE FILTER IS DECLARED UNSTABLE
(02304) * AND THE ONE-TAP FILTER IS USED AS
(02305) * C1=0.
(02306) * C2=SA , SA=ONE-TAP COEFF
(02307) * C3=0.
(02308) * OTHERWISE THE
(02309) * FILTER IS DECLARED STABLE AND
(02310) * C1=V(0)
(02311) * C2=V(1)
(02312) * C3=V(2)
(02313) *
(02314) * CODE AND QUANTIZE C1,C2,C3
(02315) * SET Y-BUFFER OF QUANTIZED VALUES
(02316) * SET D-BUFFER OF CODED VALUES
(02317) *
(02318) * APU REGISTER USAGE
(02319) * LEFT RIGHT
(02320) * A0 V(0):C1 V(0):C1
(02321) * A1 V(1):C2 V(1):C2
(02322) * A2 V(2):C3 V(2):C3
(02323) * A3 T1 T2
(02324) * A4 T3
(02325) * A5 +1:C1,C2,C3 +1:-2:CODE
(02326) * A6 CTHC*(N) 2**15
(02327) * A7 SA:VLC*(N) SA:
(02328) *
(02329) * INPUT AND OUTPUT SEQUENCE
(02330) * IQ: V(0),V(1),V(2),SA,2**15
(02331) * ,CTHC*(N),VLC*(N), N=1,...,LENGTH(W1)
(02332) *
(02333) * UQ: (FLT PT QUANTIZED VALUE(1),FIX PT CODE(1),I=1,2,3)
(02334) *
(02335) *
(02336) * STAB-APU INITIALIZATION

```


PAGE 55: (0000)DCAL01000016M.WSD.2, 29-Dec-88 14:30:48, Ed: WOLF
AP03 - STAR(V,A,U,V)

```

A19 06476 00120012 (02390)      MOV(ZERO,A2)      JAZ=C3=0.0
A1A 06478 00910091 (02391)      MOV(R,A1)        JAI=C2=SA=1-TAP COEFF..
A1B 0647A 00000076 (02392)      *START OF QUANTIZATION / CODING
A1C 0647C 02000000 (02393)      STAB$OT: NOP\MOV(IQA,A6)  JHOP\2=0-15 FOR FIXED PT COUNTING
A1D 0647E 30000024 (02394)      * CALL SUBROUTINE CQ$SB FOR EACH COEFF
A1E 06480 02200000 (02395)      * 1ST TAP INPUT
A1F 06482 30000024 (02396)      R(A0)\NOP      JR=AP=CIL\NOP
A20 06484 02400000 (02397)      CALL(CQ$SB)    JCALL QUANTIZATION SUBROUTINE
A21 06486 30000024 (02398)      * 2ND TAP INPUT
A22 06488 20322032 (02399)      R(A1)\NOP      JR=AI=C2 \ NOP
A23 0648A 10000000 (02400)      CALL(CQ$SB)    JCALL QUANTIZATION SUBROUTINE
A24 0648C 00950015 (02401)      * 3RD TAP INPUT
A25 0648E 000002A0 (02402)      R(A2)\NOP      JR=A2=C3 \ NOP
A26 06490 00760000 (02403)      CALL(CQ$SB)    JCALL QUANTIZATION SUBROUTINE
A27 06492 4E046B5 (02404)      CLEAR(RA)      JHALT APV
A28 06494 00770000 (02405)      JUMP(R)         JUMP(R)
A29 06496 00000000 (02406)      * * CQ$SB - QUANTIZATION SUBROUTINE
A2A 06498 00760000 (02407)      * PUT COEFF IN A5 AND INIT CODE
A2B 0649A 901F0026 (02408)      CQ$SB: MOV(R,A5)\MOV(ZERO,A5) JINPUT COEFF \ INIT CODE=0
A2C 0649C 00E000E0 (02409)      * INNER LOOP - FIND QUANT VALUE AND CODE
A2D 0649E 00000000 (02410)      MOV(IQA,A6)\NOP JHOP\R=A5=ZERO
A2E 064A0 9016002C (02411)      SUB(A5,A6)\MOV(A5),ADD(A5,A6) JINPUT BOUNDARY - CTH(N)\NOP
A2F 064A2 20322037 (02412)      MOV(IQA,A7)\NOP J(C-CTH(N))\UPDATE CODE, CODE+1
A30 064A4 02003A00 (02413)      JUMP(R,NULL)\NOP JINPUT QUANTIZED VALUE DVL(N)\NOP
A31 064A6 009C009C (02414)      JUMP(START$OT,FWI) JENSURE T01 SET
A32 064A8 00000000 (02415)      JUMP(C#6,T1)    JHOP IF DONE TABLE
A33 064AA 00000000 (02416)      * FOUND THE RIGHT QUANTIZATION BIN. READ IN REST OF TABLE.
A34 064AC 00E000E0 (02417)      P7: MOV(IQA,NULL) JINPUT REST OF TABLE
A35 064AE 00000000 (02418)      JUMP(C#7,FWI)   JTO ALLOW FWI TO SET
A36 064B0 9016002C (02419)      * TABLE FINISHED - OUTPUT DATA
A37 064B2 20322037 (02420)      STAB$OT: CLEAR(WI) J LET APS INPUT PRG PROCEED
A38 064B4 02003A00 (02421)      MOV(R,DQ)\MOV(R,DQ) JQUANT VALUE\FIX CODE VALUE
A39 064B6 009C009C (02422)      RETURN          JOUTPUT QUANT AND CODED VALUES
A3A 064B8 00000000 (02423)      * STAB$SZ=PA-STAR$SA JCOMPUTE SIZE
A3B 064BA 00000033 (02424)      * END
A3C 064BC 00000000 (02425)      *
A3D 064BE 00000000 (02426)      *
A3E 064C0 00000000 (02427)      *
A3F 064C2 00000000 (02428)      *
A40 064C4 00000000 (02429)      *
A41 064C6 00000000 (02430)      *
A42 064C8 00000000 (02431)      *
A43 064CA 00000000 (02432)      *
A44 064CC 00000000 (02433)      *
A45 064CE 00000000 (02434)      *
A46 064D0 00000000 (02435)      *
A47 064D2 00000000 (02436)      *
A48 064D4 00000000 (02437)      *
A49 064D6 00000000 (02438)      *
A50 064D8 00000000 (02439)      *
A51 064DA 00000000 (02440)      *
A52 064DC 00000000 (02441)      *
A53 064DE 00000000 (02442)      *
A54 064E0 00000000 (02443)      *
A55 064E2 00000000 (02444)      *
A56 064E4 00000000 (02445)      *
A57 064E6 00000000 (02446)      *
A58 064E8 00000000 (02447)      *
A59 064EA 00000000 (02448)      *
A60 064EC 00000000 (02449)      *
A61 064EE 00000000 (02450)      *
A62 064F0 00000000 (02451)      *
A63 064F2 00000000 (02452)      *
A64 064F4 00000000 (02453)      *
A65 064F6 00000000 (02454)      *
A66 064F8 00000000 (02455)      *
A67 064FA 00000000 (02456)      *
A68 064FC 00000000 (02457)      *
A69 064FE 00000000 (02458)      *
A70 06500 00000000 (02459)      *
A71 06502 00000000 (02460)      *
A72 06504 00000000 (02461)      *
A73 06506 00000000 (02462)      *
A74 06508 00000000 (02463)      *
A75 0650A 00000000 (02464)      *
A76 0650C 00000000 (02465)      *
A77 0650E 00000000 (02466)      *
A78 06510 00000000 (02467)      *
A79 06512 00000000 (02468)      *
A80 06514 00000000 (02469)      *
A81 06516 00000000 (02470)      *
A82 06518 00000000 (02471)      *
A83 0651A 00000000 (02472)      *
A84 0651C 00000000 (02473)      *
A85 0651E 00000000 (02474)      *
A86 06520 00000000 (02475)      *
A87 06522 00000000 (02476)      *
A88 06524 00000000 (02477)      *
A89 06526 00000000 (02478)      *
A90 06528 00000000 (02479)      *
A91 0652A 00000000 (02480)      *
A92 0652C 00000000 (02481)      *
A93 0652E 00000000 (02482)      *
A94 06530 00000000 (02483)      *
A95 06532 00000000 (02484)      *
A96 06534 00000000 (02485)      *
A97 06536 00000000 (02486)      *
A98 06538 00000000 (02487)      *
A99 0653A 00000000 (02488)      *
A00 0653C 00000000 (02489)      *
A01 0653E 00000000 (02490)      *
A02 06540 00000000 (02491)      *
A03 06542 00000000 (02492)      *
A04 06544 00000000 (02493)      *
A05 06546 00000000 (02494)      *
A06 06548 00000000 (02495)      *
A07 0654A 00000000 (02496)      *
A08 0654C 00000000 (02497)      *
A09 0654E 00000000 (02498)      *
A10 06550 00000000 (02499)      *
A11 06552 00000000 (02500)      *
A12 06554 00000000 (02501)      *
A13 06556 00000000 (02502)      *
A14 06558 00000000 (02503)      *
A15 0655A 00000000 (02504)      *
A16 0655C 00000000 (02505)      *
A17 0655E 00000000 (02506)      *
A18 06560 00000000 (02507)      *
A19 06562 00000000 (02508)      *
A20 06564 00000000 (02509)      *
A21 06566 00000000 (02510)      *
A22 06568 00000000 (02511)      *
A23 0656A 00000000 (02512)      *
A24 0656C 00000000 (02513)      *
A25 0656E 00000000 (02514)      *
A26 06570 00000000 (02515)      *
A27 06572 00000000 (02516)      *
A28 06574 00000000 (02517)      *
A29 06576 00000000 (02518)      *
A30 06578 00000000 (02519)      *
A31 0657A 00000000 (02520)      *
A32 0657C 00000000 (02521)      *
A33 0657E 00000000 (02522)      *
A34 06580 00000000 (02523)      *
A35 06582 00000000 (02524)      *
A36 06584 00000000 (02525)      *
A37 06586 00000000 (02526)      *
A38 06588 00000000 (02527)      *
A39 0658A 00000000 (02528)      *
A40 0658C 00000000 (02529)      *
A41 0658E 00000000 (02530)      *
A42 06590 00000000 (02531)      *
A43 06592 00000000 (02532)      *
A44 06594 00000000 (02533)      *
A45 06596 00000000 (02534)      *
A46 06598 00000000 (02535)      *
A47 0659A 00000000 (02536)      *
A48 0659C 00000000 (02537)      *
A49 0659E 00000000 (02538)      *
A50 065A0 00000000 (02539)      *
A51 065A2 00000000 (02540)      *
A52 065A4 00000000 (02541)      *
A53 065A6 00000000 (02542)      *
A54 065A8 00000000 (02543)      *
A55 065AA 00000000 (02544)      *
A56 065AC 00000000 (02545)      *
A57 065AE 00000000 (02546)      *
A58 065B0 00000000 (02547)      *
A59 065B2 00000000 (02548)      *
A60 065B4 00000000 (02549)      *
A61 065B6 00000000 (02550)      *
A62 065B8 00000000 (02551)      *
A63 065BA 00000000 (02552)      *
A64 065BC 00000000 (02553)      *
A65 065BE 00000000 (02554)      *
A66 065C0 00000000 (02555)      *
A67 065C2 00000000 (02556)      *
A68 065C4 00000000 (02557)      *
A69 065C6 00000000 (02558)      *
A70 065C8 00000000 (02559)      *
A71 065CA 00000000 (02560)      *
A72 065CC 00000000 (02561)      *
A73 065CE 00000000 (02562)      *
A74 065D0 00000000 (02563)      *
A75 065D2 00000000 (02564)      *
A76 065D4 00000000 (02565)      *
A77 065D6 00000000 (02566)      *
A78 065D8 00000000 (02567)      *
A79 065DA 00000000 (02568)      *
A80 065DC 00000000 (02569)      *
A81 065DE 00000000 (02570)      *
A82 065E0 00000000 (02571)      *
A83 065E2 00000000 (02572)      *
A84 065E4 00000000 (02573)      *
A85 065E6 00000000 (02574)      *
A86 065E8 00000000 (02575)      *
A87 065EA 00000000 (02576)      *
A88 065EC 00000000 (02577)      *
A89 065EE 00000000 (02578)      *
A90 065F0 00000000 (02579)      *
A91 065F2 00000000 (02580)      *
A92 065F4 00000000 (02581)      *
A93 065F6 00000000 (02582)      *
A94 065F8 00000000 (02583)      *
A95 065FA 00000000 (02584)      *
A96 065FC 00000000 (02585)      *
A97 065FE 00000000 (02586)      *
A98 06600 00000000 (02587)      *
A99 06602 00000000 (02588)      *
A00 06604 00000000 (02589)      *
A01 06606 00000000 (02590)      *
A02 06608 00000000 (02591)      *
A03 0660A 00000000 (02592)      *
A04 0660C 00000000 (02593)      *
A05 0660E 00000000 (02594)      *
A06 06610 00000000 (02595)      *
A07 06612 00000000 (02596)      *
A08 06614 00000000 (02597)      *
A09 06616 00000000 (02598)      *
A10 06618 00000000 (02599)      *
A11 0661A 00000000 (02600)      *
A12 0661C 00000000 (02601)      *
A13 0661E 00000000 (02602)      *
A14 06620 00000000 (02603)      *
A15 06622 00000000 (02604)      *
A16 06624 00000000 (02605)      *
A17 06626 00000000 (02606)      *
A18 06628 00000000 (02607)      *
A19 0662A 00000000 (02608)      *
A20 0662C 00000000 (02609)      *
A21 0662E 00000000 (02610)      *
A22 06630 00000000 (02611)      *
A23 06632 00000000 (02612)      *
A24 06634 00000000 (02613)      *
A25 06636 00000000 (02614)      *
A26 06638 00000000 (02615)      *
A27 0663A 00000000 (02616)      *
A28 0663C 00000000 (02617)      *
A29 0663E 00000000 (02618)      *
A30 06640 00000000 (02619)      *
A31 06642 00000000 (02620)      *
A32 06644 00000000 (02621)      *
A33 06646 00000000 (02622)      *
A34 06648 00000000 (02623)      *
A35 0664A 00000000 (02624)      *
A36 0664C 00000000 (02625)      *
A37 0664E 00000000 (02626)      *
A38 06650 00000000 (02627)      *
A39 06652 00000000 (02628)      *
A40 06654 00000000 (02629)      *
A41 06656 00000000 (02630)      *
A42 06658 00000000 (02631)      *
A43 0665A 00000000 (02632)      *
A44 0665C 00000000 (02633)      *
A45 0665E 00000000 (02634)      *
A46 06660 00000000 (02635)      *
A47 06662 00000000 (02636)      *
A48 06664 00000000 (02637)      *
A49 06666 00000000 (02638)      *
A50 06668 00000000 (02639)      *
A51 0666A 00000000 (02640)      *
A52 0666C 00000000 (02641)      *
A53 0666E 00000000 (02642)      *
A54 06670 00000000 (02643)      *
A55 06672 00000000 (02644)      *
A56 06674 00000000 (02645)      *
A57 06676 00000000 (02646)      *
A58 06678 00000000 (02647)      *
A59 0667A 00000000 (02648)      *
A60 0667C 00000000 (02649)      *
A61 0667E 00000000 (02650)      *
A62 06680 00000000 (02651)      *
A63 06682 00000000 (02652)      *
A64 06684 00000000 (02653)      *
A65 06686 00000000 (02654)      *
A66 06688 00000000 (02655)      *
A67 0668A 00000000 (02656)      *
A68 0668C 00000000 (02657)      *
A69 0668E 00000000 (02658)      *
A70 06690 00000000 (02659)      *
A71 06692 00000000 (02660)      *
A72 06694 00000000 (02661)      *
A73 06696 00000000 (02662)      *
A74 06698 00000000 (02663)      *
A75 0669A 00000000 (02664)      *
A76 0669C 00000000 (02665)      *
A77 0669E 00000000 (02666)      *
A78 06700 00000000 (02667)      *
A79 06702 00000000 (02668)      *
A80 06704 00000000 (02669)      *
A81 06706 00000000 (02670)      *
A82 06708 00000000 (02671)      *
A83 0670A 00000000 (02672)      *
A84 0670C 00000000 (02673)      *
A85 0670E 00000000 (02674)      *
A86 06710 00000000 (02675)      *
A87 06712 00000000 (02676)      *
A88 06714 00000000 (02677)      *
A89 06716 00000000 (02678)      *
A90 06718 00000000 (02679)      *
A91 0671A 00000000 (02680)      *
A92 0671C 00000000 (02681)      *
A93 0671E 00000000 (02682)      *
A94 06720 00000000 (02683)      *
A95 06722 00000000 (02684)      *
A96 06724 00000000 (02685)      *
A97 06726 00000000 (02686)      *
A98 06728 00000000 (02687)      *
A99 0672A 00000000 (02688)      *
A00 0672C 00000000 (02689)      *
A01 0672E 00000000 (02690)      *
A02 06730 00000000 (02691)      *
A03 06732 00000000 (02692)      *
A04 06734 00000000 (02693)      *
A05 06736 00000000 (02694)      *
A06 06738 00000000 (02695)      *
A07 0673A 00000000 (02696)      *
A08 0673C 00000000 (02697)      *
A09 0673E 00000000 (02698)      *
A10 06740 00000000 (02699)      *
A11 06742 00000000 (02700)      *
A12 06744 00000000 (02701)      *
A13 06746 00000000 (02702)      *
A14 06748 00000000 (02703)      *
A15 0674A 00000000 (02704)      *
A16 0674C 00000000 (02705)      *
A17 0674E 00000000 (02706)      *
A18 06750 00000000 (02707)      *
A19 06752 00000000 (02708)      *
A20 06754 00000000 (02709)      *
A21 06756 00000000 (02710)      *
A22 06758 00000000 (02711)      *
A23 0675A 00000000 (02712)      *
A24 0675C 00000000 (02713)      *
A25 0675E 00000000 (02714)      *
A26 06760 00000000 (02715)      *
A27 06762 00000000 (02716)      *
A28 06764 00000000 (02717)      *
A29 06766 00000000 (02718)      *
A30 06768 00000000 (02719)      *
A31 0676A 00000000 (02720)      *
A32 0676C 00000000 (02721)      *
A33 0676E 00000000 (02722)      *
A34 06770 00000000 (02723)      *
A35 06772 00000000 (02724)      *
A36 06774 00000000 (02725)      *
A37 06776 00000000 (02726)      *
A38 06778 00000000 (02727)      *
A39 0677A 00000000 (02728)      *
A40 0677C 00000000 (02729)      *
A41 0677E 00000000 (02730)      *
A42 06780 00000000 (02731)      *
A43 06782 00000000 (02732)      *
A44 06784 00000000 (02733)      *
A45 06786 00000000 (02734)      *
A46 06788 00000000 (02735)      *
A47 0678A 00000000 (02736)      *
A48 0678C 00000000 (02737)      *
A49 0678E 00000000 (02738)      *
A50 06790 00000000 (02739)      *
A51 06792 00000000 (02740)      *
A52 06794 00000000 (02741)      *
A53 06796 00000000 (02742)      *
A54 06798 00000000 (02743)      *
A55 0679A 00000000 (02744)      *
A56 0679C 00000000 (02745)      *
A57 0679E 00000000 (02746)      *
A58 06800 00000000 (02747)      *
A59 06802 00000000 (02748)      *
A60 06804 00000000 (02749)      *
A61 06806 00000000 (02750)      *
A62 06808 00000000 (02751)      *
A63 0680A 00000000 (02752)      *
A64 0680C 00000000 (02753)      *
A65 0680E 00000000 (02754)      *
A66 06810 00000000 (02755)      *
A67 06812 00000000 (02756)      *
A68 06814 00000000 (02757)      *
A69 06816 00000000 (02758)      *
A70 06818 00000000 (02759)      *
A71 0681A 00000000 (02760)      *
A72 0681C 00000000 (02761)      *
A73 0681E 00000000 (02762)      *
A74 06820 00000000 (02763)      *
A75 06822 00000000 (02764)      *
A76 06824 00000000 (02765)      *
A77 06826 00000000 (02766)      *
A78 06828 00000000 (02767)      *
A79 0682A 00000000 (02768)      *
A80 0682C 00000000 (02769)      *
A81 0682E 00000000 (02770)      *
A82 06830 00000000 (02771)      *
A83 06832 00000000 (02772)      *
A84 06834 00000000 (02773)      *
A85 06836 00000000 (02774)      *
A86 06838 00000000 (02775)      *
A87 0683A 00000000 (02776)      *
A88 0683C 00000000 (02777)      *
A89 0683E 00000000 (02778)      *
A90 06840 00000000 (02779)      *
A91 06842 00000000 (02780)      *
A92 06844 00000000 (02781)      *
A93 06846 00000000 (02782)      *
A94 06848 00000000 (02783)      *
A95 0684A 00000000 (02784)      *
A96 0684C 00000000 (02785)      *
A97 0684E 00000000 (02786)      *
A98 06850 00000000 (02787)      *
A99 06852 00000000 (02788)      *
A00 06854 00000000 (02789)      *
A01 06856 00000000 (02790)      *
A02 06858 00000000 (02791)      *
A03 0685A 00000000 (02792)      *
A04 0685C 00000000 (02793)      *
A05 0685E 00000000 (02794)      *
A06 06860 00000000 (02795)      *
A07 06862 00000000 (02796)      *
A08 06864 00000000 (02797)      *
A09 06866 00000000 (02798)      *
A10 06868 00000000 (02799)      *
A11 0686A 00000000 (02800)      *
A12 0686C 00000000 (02801)      *
A13 0686E 00000000 (02802)      *
A14 06870 00000000 (02803)      *
A15 06872 00000000 (02804)      *
A16 06874 00000000 (02805)      *
A17 06876 00000000 (02806)      *
A18 06878 00000000 (02807)      *
A19 0687A 00000000 (02808)      *
A20 0687C 00000000 (02809)      *
A21 0687E 00000000 (02810)      *
A22 06880 00000000 (02811)      *
A23 06882 00000000 (02812
```


196

PAGE 59: [RNDJ]DCAL6>DMMIGN.HSO.2, 29-Dec-88 14:38:40, Ed: WOLF
APU3-INVPF(V,A,U,V) 3-TAP PITCH INVERSE FILTER

```

A00 0651A 00200000 (02592) INVPUS: BEGIN APU(INVPF)
A01 0651C 00EC0000 (02593) INVPUS$A: MOV(IQA,M0) \ NOP
A02 0651E 04000000 (02594) MOV(IQA,M4) \ NOP
A03 06520 00800000 (02595) MUL(M0,M4) \ NOP
A04 06522 3A000000 (02596) MOV(P,A0) \ NOP
A05 06524 009C0000 (02597) ALIGN(A0) \ NOP
A06 06526 90100000 (02598) MOV(R,DQ) \ NOP
A07 06528 00000000 (02599) WHERE WE WAIT ON OQE, PROCEED APS, READ/WRITE THE INSTRUCTION, WAIT
A08 0652A 20372037 (02600) 3 AND PROCEED THE APS AGAIN, AND THEN GET DOWN TO THE COMPUTATION.
A09 0652C 00FC0000 (02601) JUMPC(B1,ONE)
A10 0652E 90100000 (02602) NOP
A11 06530 00000000 (02603) CLEAR(VI)
A12 06532 20372037 (02604) MOV(IQA,OQ) \ NOP
A13 06534 00EC0000 (02605) JUMPC(B2,OQE)
A14 06536 00ED0000 (02606) NOP
A15 06538 00000000 (02607) CLEAR(VI)
A16 0653A 00000000 (02608) MOV(IQA,M4) \ MOV(IQA,M6)
A17 0653C 00000000 (02609) MOV(IQA,M5) \ MOV(IQA,M4)
A18 0653E 00000000 (02610) MOV(IQA,M6) \ MOV(IQA,M5)
A19 06540 00000000 (02611) JMAIN LOOP
A20 06542 00000000 (02612) MOV(IQA,M0)
A21 06544 00000000 (02613) MOV(A1),MUL(M0,M4)
A22 06546 00000000 (02614) MOV(A0),ADD(A0,A1)
A23 06548 00000000 (02615) MOV(IQA,A2) \ NOP
A24 0654A 00000000 (02616) NOP \ MOV(IQA,A2)
A25 0654C 00000000 (02617) MOV(R,DQ)
A26 0654E 00000000 (02618) MOV(IQA,M1)
A27 06550 00000000 (02619) MOV(A1),MUL(M1,M5)
A28 06552 00000000 (02620) ADD(A2,A1)
A29 06554 00000000 (02621) MOV(IQA,M2) \ NOP
A30 06556 00000000 (02622) NOP \ MOV(IQA,M2)
A31 06558 00000000 (02623) MOV(A1),MUL(M2,M6)
A32 0655A 00000000 (02624) MOV(A0),ADD(A0,A1)
A33 0655C 00000000 (02625) JUMPC(B3,E0)
A34 0655E 00000000 (02626) CLEAR(A0)
A35 06560 00000000 (02627) JUMP(0)
A36 06562 00000000 (02628) INVPUS$Z-B1-INVPUS$A
A37 06564 00000000 (02629) END
A38 06566 00000000 (02630)
A39 06568 00000000 (02631)
A40 0656A 00000000 (02632)
A41 0656C 00000000 (02633)
A42 0656E 00000000 (02634)
A43 06570 00000000 (02635)
A44 06572 00000000 (02636)
A45 06574 00000000 (02637)
A46 06576 00000000 (02638)
A47 06578 00000000 (02639)
A48 0657A 00000000 (02640)
A49 0657C 00000000 (02641)
A50 0657E 00000000 (02642)
A51 06580 00000000 (02643)
A52 06582 00000000 (02644)
A53 06584 00000000 (02645)
A54 06586 00000000 (02646)
A55 06588 00000000 (02647)
A56 0658A 00000000 (02648)
A57 0658C 00000000 (02649)
A58 0658E 00000000 (02650)
A59 06590 00000000 (02651)
A60 06592 00000000 (02652)
A61 06594 00000000 (02653)
A62 06596 00000000 (02654)
A63 06598 00000000 (02655)
A64 0659A 00000000 (02656)
A65 0659C 00000000 (02657)
A66 0659E 00000000 (02658)
A67 065A0 00000000 (02659)
A68 065A2 00000000 (02660)
A69 065A4 00000000 (02661)
A70 065A6 00000000 (02662)
A71 065A8 00000000 (02663)
A72 065AA 00000000 (02664)
A73 065AC 00000000 (02665)
A74 065AE 00000000 (02666)
A75 065B0 00000000 (02667)
A76 065B2 00000000 (02668)
A77 065B4 00000000 (02669)
A78 065B6 00000000 (02670)
A79 065B8 00000000 (02671)
A80 065BA 00000000 (02672)
A81 065BC 00000000 (02673)
A82 065BE 00000000 (02674)
A83 065C0 00000000 (02675)
A84 065C2 00000000 (02676)
A85 065C4 00000000 (02677)
A86 065C6 00000000 (02678)
A87 065C8 00000000 (02679)
A88 065CA 00000000 (02680)
A89 065CC 00000000 (02681)
A90 065CE 00000000 (02682)
A91 065D0 00000000 (02683)
A92 065D2 00000000 (02684)
A93 065D4 00000000 (02685)
A94 065D6 00000000 (02686)
A95 065D8 00000000 (02687)
A96 065DA 00000000 (02688)
A97 065DC 00000000 (02689)
A98 065DE 00000000 (02690)
A99 065E0 00000000 (02691)
A100 065E2 00000000 (02692)
A101 065E4 00000000 (02693)
A102 065E6 00000000 (02694)
A103 065E8 00000000 (02695)
A104 065EA 00000000 (02696)
A105 065EC 00000000 (02697)
A106 065EE 00000000 (02698)
A107 065F0 00000000 (02699)
A108 065F2 00000000 (02700)
A109 065F4 00000000 (02701)
A110 065F6 00000000 (02702)
A111 065F8 00000000 (02703)
A112 065FA 00000000 (02704)
A113 065FC 00000000 (02705)
A114 065FE 00000000 (02706)
A115 06600 00000000 (02707)
A116 06602 00000000 (02708)
A117 06604 00000000 (02709)
A118 06606 00000000 (02710)
A119 06608 00000000 (02711)
A120 0660A 00000000 (02712)
A121 0660C 00000000 (02713)
A122 0660E 00000000 (02714)
A123 06610 00000000 (02715)
A124 06612 00000000 (02716)
A125 06614 00000000 (02717)
A126 06616 00000000 (02718)
A127 06618 00000000 (02719)
A128 0661A 00000000 (02720)
A129 0661C 00000000 (02721)
A130 0661E 00000000 (02722)
A131 06620 00000000 (02723)
A132 06622 00000000 (02724)
A133 06624 00000000 (02725)
A134 06626 00000000 (02726)
A135 06628 00000000 (02727)
A136 0662A 00000000 (02728)
A137 0662C 00000000 (02729)
A138 0662E 00000000 (02730)
A139 06630 00000000 (02731)
A140 06632 00000000 (02732)
A141 06634 00000000 (02733)
A142 06636 00000000 (02734)
A143 06638 00000000 (02735)
A144 0663A 00000000 (02736)
A145 0663C 00000000 (02737)
A146 0663E 00000000 (02738)
A147 06640 00000000 (02739)
A148 06642 00000000 (02740)
A149 06644 00000000 (02741)
A150 06646 00000000 (02742)
A151 06648 00000000 (02743)
A152 0664A 00000000 (02744)
A153 0664C 00000000 (02745)
A154 0664E 00000000 (02746)
A155 06650 00000000 (02747)
A156 06652 00000000 (02748)
A157 06654 00000000 (02749)
A158 06656 00000000 (02750)
A159 06658 00000000 (02751)
A160 0665A 00000000 (02752)
A161 0665C 00000000 (02753)
A162 0665E 00000000 (02754)
A163 06660 00000000 (02755)
A164 06662 00000000 (02756)
A165 06664 00000000 (02757)
A166 06666 00000000 (02758)
A167 06668 00000000 (02759)
A168 0666A 00000000 (02760)
A169 0666C 00000000 (02761)
A170 0666E 00000000 (02762)
A171 06670 00000000 (02763)
A172 06672 00000000 (02764)
A173 06674 00000000 (02765)
A174 06676 00000000 (02766)
A175 06678 00000000 (02767)
A176 0667A 00000000 (02768)
A177 0667C 00000000 (02769)
A178 0667E 00000000 (02770)
A179 06680 00000000 (02771)
A180 06682 00000000 (02772)
A181 06684 00000000 (02773)
A182 06686 00000000 (02774)
A183 06688 00000000 (02775)
A184 0668A 00000000 (02776)
A185 0668C 00000000 (02777)
A186 0668E 00000000 (02778)
A187 06690 00000000 (02779)
A188 06692 00000000 (02780)
A189 06694 00000000 (02781)
A190 06696 00000000 (02782)
A191 06698 00000000 (02783)
A192 0669A 00000000 (02784)
A193 0669C 00000000 (02785)
A194 0669E 00000000 (02786)
A195 066A0 00000000 (02787)
A196 066A2 00000000 (02788)
A197 066A4 00000000 (02789)
A198 066A6 00000000 (02790)
A199 066A8 00000000 (02791)
A200 066AA 00000000 (02792)
A201 066AC 00000000 (02793)
A202 066AE 00000000 (02794)
A203 066B0 00000000 (02795)
A204 066B2 00000000 (02796)
A205 066B4 00000000 (02797)
A206 066B6 00000000 (02798)
A207 066B8 00000000 (02799)
A208 066BA 00000000 (02800)
A209 066BC 00000000 (02801)
A210 066BE 00000000 (02802)
A211 066C0 00000000 (02803)
A212 066C2 00000000 (02804)
A213 066C4 00000000 (02805)
A214 066C6 00000000 (02806)
A215 066C8 00000000 (02807)
A216 066CA 00000000 (02808)
A217 066CC 00000000 (02809)
A218 066CE 00000000 (02810)
A219 066D0 00000000 (02811)
A220 066D2 00000000 (02812)
A221 066D4 00000000 (02813)
A222 066D6 00000000 (02814)
A223 066D8 00000000 (02815)
A224 066DA 00000000 (02816)
A225 066DC 00000000 (02817)
A226 066DE 00000000 (02818)
A227 066E0 00000000 (02819)
A228 066E2 00000000 (02820)
A229 066E4 00000000 (02821)
A230 066E6 00000000 (02822)
A231 066E8 00000000 (02823)
A232 066EA 00000000 (02824)
A233 066EC 00000000 (02825)
A234 066EE 00000000 (02826)
A235 066F0 00000000 (02827)
A236 066F2 00000000 (02828)
A237 066F4 00000000 (02829)
A238 066F6 00000000 (02830)
A239 066F8 00000000 (02831)
A240 066FA 00000000 (02832)
A241 066FC 00000000 (02833)
A242 066FE 00000000 (02834)
A243 06700 00000000 (02835)
A244 06702 00000000 (02836)
A245 06704 00000000 (02837)
A246 06706 00000000 (02838)
A247 06708 00000000 (02839)
A248 0670A 00000000 (02840)
A249 0670C 00000000 (02841)
A250 0670E 00000000 (02842)
A251 06710 00000000 (02843)
A252 06712 00000000 (02844)
A253 06714 00000000 (02845)
A254 06716 00000000 (02846)
A255 06718 00000000 (02847)
A256 0671A 00000000 (02848)
A257 0671C 00000000 (02849)
A258 0671E 00000000 (02850)
A259 06720 00000000 (02851)
A260 06722 00000000 (02852)
A261 06724 00000000 (02853)
A262 06726 00000000 (02854)
A263 06728 00000000 (02855)
A264 0672A 00000000 (02856)
A265 0672C 00000000 (02857)
A266 0672E 00000000 (02858)
A267 06730 00000000 (02859)
A268 06732 00000000 (02860)
A269 06734 00000000 (02861)
A270 06736 00000000 (02862)
A271 06738 00000000 (02863)
A272 0673A 00000000 (02864)
A273 0673C 00000000 (02865)
A274 0673E 00000000 (02866)
A275 06740 00000000 (02867)
A276 06742 00000000 (02868)
A277 06744 00000000 (02869)
A278 06746 00000000 (02870)
A279 06748 00000000 (02871)
A280 0674A 00000000 (02872)
A281 0674C 00000000 (02873)
A282 0674E 00000000 (02874)
A283 06750 00000000 (02875)
A284 06752 00000000 (02876)
A285 06754 00000000 (02877)
A286 06756 00000000 (02878)
A287 06758 00000000 (02879)
A288 0675A 00000000 (02880)
A289 0675C 00000000 (02881)
A290 0675E 00000000 (02882)
A291 06760 00000000 (02883)
A292 06762 00000000 (02884)
A293 06764 00000000 (02885)
A294 06766 00000000 (02886)
A295 06768 00000000 (02887)
A296 0676A 00000000 (02888)
A297 0676C 00000000 (02889)
A298 0676E 00000000 (02890)
A299 06770 00000000 (02891)
A300 06772 00000000 (02892)
A301 06774 00000000 (02893)
A302 06776 00000000 (02894)
A303 06778 00000000 (02895)
A304 0677A 00000000 (02896)
A305 0677C 00000000 (02897)
A306 0677E 00000000 (02898)
A307 06780 00000000 (02899)
A308 06782 00000000 (02900)
A309 06784 00000000 (02901)
A310 06786 00000000 (02902)
A311 06788 00000000 (02903)
A312 0678A 00000000 (02904)
A313 0678C 00000000 (02905)
A314 0678E 00000000 (02906)
A315 06790 00000000 (02907)
A316 06792 00000000 (02908)
A317 06794 00000000 (02909)
A318 06796 00000000 (02910)
A319 06798 00000000 (02911)
A320 0679A 00000000 (02912)
A321 0679C 00000000 (02913)
A322 0679E 00000000 (02914)
A323 067A0 00000000 (02915)
A324 067A2 00000000 (02916)
A325 067A4 00000000 (02917)
A326 067A6 00000000 (02918)
A327 067A8 00000000 (02919)
A328 067AA 00000000 (02920)
A329 067AC 00000000 (02921)
A330 067AE 00000000 (02922)
A331 067B0 00000000 (02923)
A332 067B2 00000000 (02924)
A333 067B4 00000000 (02925)
A334 067B6 00000000 (02926)
A335 067B8 00000000 (02927)
A336 067BA 00000000 (02928)
A337 067BC 00000000 (02929)
A338 067BE 00000000 (02930)
A339 067C0 00000000 (02931)
A340 067C2 00000000 (02932)
A341 067C4 00000000 (02933)
A342 067C6 00000000 (02934)
A343 067C8 00000000 (02935)
A344 067CA 00000000 (02936)
A345 067CC 00000000 (02937)
A346 067CE 00000000 (02938)
A347 067D0 00000000 (02939)
A348 067D2 00000000 (02940)
A349 067D4 00000000 (02941)
A350 067D6 00000000 (02942)
A351 067D8 00000000 (02943)
A352 067DA 00000000 (02944)
A353 067DC 00000000 (02945)
A354 067DE 00000000 (02946)
A355 067E0 00000000 (02947)
A356 067E2 00000000 (02948)
A357 067E4 00000000 (02949)
A358 067E6 00000000 (02950)
A359 067E8 00000000 (02951)
A360 067EA 00000000 (02952)
A361 067EC 00000000 (02953)
A362 067EE 00000000 (02954)
A363 067F0 00000000 (02955)
A364 067F2 00000000 (02956)
A365 067F4 00000000 (02957)
A366 067F6 00000000 (02958)
A367 067F8 00000000 (02959)
A368 067FA 00000000 (02960)
A369 067FC 00000000 (02961)
A370 067FE 00000000 (02962)
A371 06800 00000000 (02963)
A372 06802 00000000 (02964)
A373 06804 00000000 (02965)
A374 06806 00000000 (02966)
A375 06808 00000000 (02967)
A376 0680A 00000000 (02968)
A377 0680C 00000000 (02969)
A378 0680E 00000000 (02970)
A379 06810 00000000 (02971)
A380 06812 00000000 (02972)
A381 06814 00000000 (02973)
A382 06816 00000000 (02974)
A383 06818 00000000 (02975)
A384 0681A 00000000 (02976)
A385 0681C 00000000 (02977)
A386 0681E 00000000 (02978)
A387 06820 00000000 (02979)
A388 06822 00000000 (02980)
A389 06824 00000000 (02981)
A390 06826 00000000 (02982)
A391 06828 00000000 (02983)
A392 0682A 00000000 (02984)
A393 0682C 00000000 (02985)
A394 0682E 00000000 (02986)
A395 06830 00000000 (02987)
A396 06832 00000000 (02988)
A397 06834 00000000 (02989)
A398 06836 00000000 (02990)
A399 06838 00000000 (02991)
A400 0683A 00000000 (02992)
A401 0683C 00000000 (02993)
A402 0683E 00000000 (02994)
A403 06840 00000000 (02995)
A404 06842 00000000 (02996)
A405 06844 00000000 (02997)
A406 06846 00000000 (02998)
A407 06848 00000000 (02999)
A408 0684A 00000000 (03000)
A409 0684C 00000000 (03001)
A410 0684E 00000000 (03002)
A411 06850 00000000 (03003)
A412 06852 00000000 (03004)
A413 06854 00000000 (03005)
A414 06856 00000000 (03006)
A415 06858 00000000 (03007)
A416 0685A 00000000 (03008)
A417 0685C 00000000 (03009)
A418 0685E 00000000 (03010)
A419 06860 00000000 (03011)
A420 06862 00000000 (03012)
A421 06864 00000000 (03013)
A422 06866 00000000 (03014)
A423 06868 00000000 (03015)
A424 0686A 00000000 (03016)
A425 0686C 00000000 (03017)
A426 0686E 00000000 (03018)
A427 06870 00000000 (03019)
A428 06872 00000000 (03020)
A429 06874 00000000 (03021)
A430 06876 00000000 (03022)
A431 06878 00000000 (03023)
A432 0687A 00000000 (03024)
A433 0687C 00000000 (03025)
A434 0687E 00000000 (03026)
A435 06880 00000000 (03027)
A436 06882 00000000 (03028)
A437 06884 00000000 (03029)
A438 06886 00000000 (03030)
A439 06888 00000000 (03031)
A440 0688A 00000000 (03032)
A441 0688C 00000000 (03033)
A442 0688E 00000000 (03034)
A443 06890 00000000 (03035)
A444 06892 00000000 (03036)
A445 06894 00000000 (03037)
A446 06896 00000000 (03038)
A447 06898 00000000 (03039)
A
```

```

(02630) * APS3-INPUT(Y,A,U,V) APS PROGRAM FOR INPUT
(02631) *
(02632) * J. WOLF 7/1/88
(02633) *
(02634) * THIS APS PROGRAM MODIFIES ONE INSTRUCTION IN ITSELF BY:
(02635) * 0. READING PITCH FROM SCALAR A, AND FIXING IT
(02636) * 1. WRITING IT INTO THE RN OF THE BUS-1 COPY OF THE INSTRUCTION
(02637) * 2. WAITING ON ONE TO BE SURE IT'S THERE (PLUS 1 NOP THAT STORER
(02638) * SAVES IS NECESSARY)
(02639) * 3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
(02640) * 4. WRITING IT TO APSHEN IN PSEUDOMEMORY
(02641) * 5. WAITING FOR ONE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
(02642) * PROCEED.
(02643) *
(02644) * HEADER BLOCK
(02645) *
(02646) *
(02647) *
(02648) *
(02649) *
(02650) *
(02651) *
(02652) *
(02653) *
(02654) * INPUT PROGRAM
(02655) * BR0: V(1), THEN U(M-N+1)
(02656) * BR1: 0(M)
(02657) * BR2: LOOP COUNTER, STARTS AT VBS-1
(02658) * BR3: SCRATCH
(02659) *
(02660) * INPSI: BEGIN APS(INVPS)
(02661) * JSH(INVPS0,P2)
(02662) * SET(R0)
(02663) * LOAD(BR0,SVTSUM(1),TF) JGEN (2**15) ADR
(02664) * INVPSS: LOAD(BR0,M$(1),TF) JGEN SA ADDRESS (PITCH)
(02665) * SET(W1) JWAIT FOR APO TO SIGNAL WRITE COMPLETE
(02666) * NOP
(02667) * LOAD(BR0,INVPSSM(1),TF) JLOAD MODIFIED INSTR THRU APU INTO APS MEMORY
(02668) * SET(W1) JWAIT AGAIN FOR WRITE TO COMPLETE
(02669) *
(02670) *
(02671) *
(02672) *
(02673) *
(02674) *
(02675) *
(02676) *
(02677) *
(02678) *
(02679) *
(02680) *
(02681) *
(02682) *
(02683) *
(02684) *
(02685) *
(02686) *
(02687) *
(02688) *
(02689) *
(02690) *
(02691) *
(02692) *
(02693) *
(02694) *
(02695) *
(02696) *
(02697) *
(02698) *
(02699) *
(02700) *
(02701) *
(02702) *
(02703) *
(02704) *
(02705) *
(02706) *
(02707) *
(02708) *
(02709) *
(02710) *
(02711) *
(02712) *
(02713) *
(02714) *
(02715) *
(02716) *
(02717) *
(02718) *
(02719) *
(02720) *
(02721) *
(02722) *
(02723) *
(02724) *
(02725) *
(02726) *
(02727) *
(02728) *
(02729) *
(02730) *
(02731) *
(02732) *
(02733) *
(02734) *
(02735) *
(02736) *
(02737) *
(02738) *
(02739) *
(02740) *
(02741) *
(02742) *
(02743) *
(02744) *
(02745) *
(02746) *
(02747) *
(02748) *
(02749) *
(02750) *
(02751) *
(02752) *
(02753) *
(02754) *
(02755) *
(02756) *
(02757) *
(02758) *
(02759) *
(02760) *
(02761) *
(02762) *
(02763) *
(02764) *
(02765) *
(02766) *
(02767) *
(02768) *
(02769) *
(02770) *
(02771) *
(02772) *
(02773) *
(02774) *
(02775) *
(02776) *
(02777) *
(02778) *
(02779) *
(02780) *
(02781) *
(02782) *
(02783) *
(02784) *
(02785) *
(02786) *
(02787) *
(02788) *
(02789) *
(02790) *
(02791) *
(02792) *
(02793) *
(02794) *
(02795) *
(02796) *
(02797) *
(02798) *
(02799) *
(02800) *
(02801) *
(02802) *
(02803) *
(02804) *
(02805) *
(02806) *
(02807) *
(02808) *
(02809) *
(02810) *
(02811) *
(02812) *
(02813) *
(02814) *
(02815) *
(02816) *
(02817) *
(02818) *
(02819) *
(02820) *
(02821) *
(02822) *
(02823) *
(02824) *
(02825) *
(02826) *
(02827) *
(02828) *
(02829) *
(02830) *
(02831) *
(02832) *
(02833) *
(02834) *
(02835) *
(02836) *
(02837) *
(02838) *
(02839) *
(02840) *
(02841) *
(02842) *
(02843) *
(02844) *
(02845) *
(02846) *
(02847) *
(02848) *
(02849) *
(02850) *
(02851) *
(02852) *
(02853) *
(02854) *
(02855) *
(02856) *
(02857) *
(02858) *
(02859) *
(02860) *
(02861) *
(02862) *
(02863) *
(02864) *
(02865) *
(02866) *
(02867) *
(02868) *
(02869) *
(02870) *
(02871) *
(02872) *
(02873) *
(02874) *
(02875) *
(02876) *
(02877) *
(02878) *
(02879) *
(02880) *
(02881) *
(02882) *
(02883) *
(02884) *
(02885) *
(02886) *
(02887) *
(02888) *
(02889) *
(02890) *
(02891) *
(02892) *
(02893) *
(02894) *
(02895) *
(02896) *
(02897) *
(02898) *
(02899) *
(02900) *
(02901) *
(02902) *
(02903) *
(02904) *
(02905) *
(02906) *
(02907) *
(02908) *
(02909) *
(02910) *
(02911) *
(02912) *
(02913) *
(02914) *
(02915) *
(02916) *
(02917) *
(02918) *
(02919) *
(02920) *
(02921) *
(02922) *
(02923) *
(02924) *
(02925) *
(02926) *
(02927) *
(02928) *
(02929) *
(02930) *
(02931) *
(02932) *
(02933) *
(02934) *
(02935) *
(02936) *
(02937) *
(02938) *
(02939) *
(02940) *
(02941) *
(02942) *
(02943) *
(02944) *
(02945) *
(02946) *
(02947) *
(02948) *
(02949) *
(02950) *
(02951) *
(02952) *
(02953) *
(02954) *
(02955) *
(02956) *
(02957) *
(02958) *
(02959) *
(02960) *
(02961) *
(02962) *
(02963) *
(02964) *
(02965) *
(02966) *
(02967) *
(02968) *
(02969) *
(02970) *
(02971) *
(02972) *
(02973) *
(02974) *
(02975) *
(02976) *
(02977) *
(02978) *
(02979) *
(02980) *
(02981) *
(02982) *
(02983) *
(02984) *
(02985) *
(02986) *
(02987) *
(02988) *
(02989) *
(02990) *
(02991) *
(02992) *
(02993) *
(02994) *
(02995) *
(02996) *
(02997) *
(02998) *
(02999) *
(03000) *

```

```

PAGE 611 (00ND)(<DC116>00NIGN.MSO.2, 29-Dec-88 14:30:40, E4: WOLF
APSJ-INVP(Y,A,B,V) APS PROGRAM FOR INVP

```

A13 06500 26090025 (02603)	SUB0(BR0,BR2)	POINT TO V(B-N)
A14 0650A 2050100E (02604)	LOAD(BR1,C13)	SUBA
A15 0650C 2A700000 (02605)	LOAD(BR3,M\$)	(UNUSED) VBS-1
A16 0650E 2C120000 (02606)	SUB(BR1,M\$)	DU INCR
A17 06590 2F1A0100 (02607)	ADD(BR1,216*W\$)	OFFSET FOR 216 PAST VALUES
A18 06592 30700000 (02608)	LOAD(BR3,C03)	(UNUSED) YBA
A19 06594 37600000 (02609)	LOAD(BR2,M\$)	VBS-1 FOR LOOP COUNTER
A1A 06596 34700000 (02690)	LOAD(BR3,M\$)	(UNUSED) YINC
A1B 06590 3609003A (02691)	MAIN LOOP	
A1C 0659A 3099003A (02692)	ADDL(BR0,W\$,TF)	DU(M-N+1) (REMEMBER BUFFER U IS COMPACT)
A1D 0659C 3A99003A (02693)	ADDL(BR1,W\$,TF)	DU(M)
A1E 0659E 3C090032 (02694)	ADDL(BR1,W\$,TF)	DU(M+1)
A1F 065A0 3F090032 (02695)	SUBL(BR0,W\$,TF)	DU(M-N)
A20 065A2 4009003E (02697)	SUBL(BR0,W\$,TF)	DU(M-N-1)
A21 065A4 47291002 (02698)	ADDL(BR0,3-W\$,TF)	DU(M-N+2)
A22 065A6 44200031 (02699)	SUBL(BR2,2),JUMPP(B4)	WE'RE DOING 2 POINTS AT ONCE
A23 065A8 46000020 (02700)	CLEAR(R1)	
	NOPI(0)	
	(02701)	
	(02702)	
	JUMPTUT PROGRAM	
	DU0: V(M)	
	DU1: LOOP COUNTER (STARTS AT VBS-1)	
	DU3: SCRATCH	
	(02706)	
	(02707)	
A24 065AA 40300032 (02708)	INVP0: SET(RA)	START APV
A25 065AC 40726505 (02709)	GENERATE BUS1 ADR FOR THE INSTRUCTION TO BE MODIFIED (RIGHT HALF)	
	LOAD(BW3,INVP\$M\$S(1),TE)	
A26 065AE 4CF3EFC0 (02710)	GENERATE PSEUDOMEMORY ADR FOR LOADING THE APS MEMORY	
	LOAD(BW3,AP\$M\$S(1),TF)	OUTPUT
	NOW GET ON WITH THE "ORDINARY"	JUMPTUT 2 DUMMIES
A27 06500 4FF20794 (02714)	LOAD(BW3,OW\$S(1),TF)	
A28 06502 50F20794 (02715)	LOAD(BW3,OW\$S(1),TF)	YBA
A29 06504 52400022 (02716)	LOAD(BW0,C03)	VBS-1 FOR LOOP COUNTER
A2A 06506 54500000 (02717)	LOAD(BW1,M\$)	DU INCR
A2B 06508 56020000 (02718)	SUB(BW0,M\$)	
	JUMPTUT LOOP	
A2C 0650A 500A0006 (02720)	ADD(BW0,C03,TF)	
A2D 0650C 5A112C01 (02721)	SUBL(BW1,1),JUMPP(B5)	
A2E 0650E 5C200030 (02722)	CLFAR(RO)	
A2F 065C0 5F000020 (02723)	NOPI(0)	
065C2 (02724)	END	
000F450A (02725)	INVP\$A=0C	ASSIGN VALUE TO CHAIN ANCHOR
065C2 00000000 (02726)	INVP\$S: DATA 7F'0'0'	BLOCK FOR CONSTRUCTED INSTRUCTIONS
...		
0000006E (02727)	INVP\$S2=0L-INVP\$1	COMPUTE BUS1 MODULE SIZE

(02720) - AP3-ADPEAK(U) MONITOR PEAK ABS VAL OF INPUT SPEECH
(02729) ; J WOLF, 10/30/80

1027307 ; BINDS TO APS3-ADPEAK

PERFORMS A RUNNING MAXIMUM-ABSOLUTE-VALUE SCAN OF A BUFFER OF SPEECH DATA INPUT FROM THE ADAM, TO PERMIT MONITORING OF THE (PSR) LEVEL OF THE INPUT SIGNAL. THE RUNNING PEAK IS MAINTAINED AS AN INTEGER (VALUE = 0-2047), FOR EASE OF DISPLAY TO THE USER. THE A/D SIGNAL VALUES RANGE FROM -1 TO +1, SO THERE MUST BE BOTH A MAGNITUDE CHANGE (OF 2ⁿ-1) AND AN INTEGER/FLTPT CHANGE WHEN THIS RUNNING MAX IS BROUGHT IN AND WRITTEN OUT AGAIN.

```

      ... , INPUT SEQUENCE:
      (02741) , PREVIOUS MAX (INTEGER), U(0), ..., U(UBS-1), CFI)
      (02742) ,

```

```

(02744) , NEW MAX (INTEGER), [ED]

```

2 ELEMENTS PER 7 APU CYCLES = -.35 USEC/ELEMENT
1 MEM REF PER ELEMENT = .5 USEC (DOS 1) PER ELEMENT

(02749) ; LEFT
RIGHT

002730	11	1000	IMAXI(000)
002751	11	1000	IMAXI(000)

(02752) JUKING LOU CLEANOP
(02753) ; A2; IMAXI (EVEN)
(02754) ; A2; IMAXI (EVEN)

(02754)	: A3:	MAXI (OVERALL)
(02755)	: A4:	MAXI / 16

NAME
(2560)
(95120)

(02758) 1141 108KUS\$A
 (03759) 1141 108KUS\$A

(02760) ,
033761) 088H. HECIA 10U/C 0882W

02762) ADPKU\$A: MOV(IQA,A1)

NOV(A1), MORH(A1)

(92766) M1: NOP \ MOV(IGA, 18)

(02769) JUMPS(EXMID,FI)

(02770) MAXABS(A0,A1) \ 1

```

(02772) EXMID: MOV(R,A1) \ MOV(

```

```

(02774)
MAX(A1,A2) \ MOP
MAX(CAT,ME) \ MOP

```

(02776) MDV(A4), ALIGN(A4)

```

(02777)
(02778)
      MOV(R12,R0) \ MUF
      CLEAR(RA)

```

10PKUS52=01-10PKUSA

JAPU	START	ADDRESS
JAPU	MODULE	SIZE
0000	0000	0000
0001	0001	0001
0002	0002	0002
0003	0003	0003
0004	0004	0004
0005	0005	0005
0006	0006	0006
0007	0007	0007
0008	0008	0008
0009	0009	0009
0010	0010	0010
0011	0011	0011
0012	0012	0012
0013	0013	0013
0014	0014	0014
0015	0015	0015
0016	0016	0016
0017	0017	0017
0018	0018	0018
0019	0019	0019
0020	0020	0020
0021	0021	0021
0022	0022	0022
0023	0023	0023
0024	0024	0024
0025	0025	0025
0026	0026	0026
0027	0027	0027
0028	0028	0028
0029	0029	0029
0030	0030	0030
0031	0031	0031
0032	0032	0032
0033	0033	0033
0034	0034	0034
0035	0035	0035
0036	0036	0036
0037	0037	0037
0038	0038	0038
0039	0039	0039
0040	0040	0040
0041	0041	0041
0042	0042	0042
0043	0043	0043
0044	0044	0044
0045	0045	0045
0046	0046	0046
0047	0047	0047
0048	0048	0048
0049	0049	0049
0050	0050	0050
0051	0051	0051
0052	0052	0052
0053	0053	0053
0054	0054	0054
0055	0055	0055
0056	0056	0056
0057	0057	0057
0058	0058	0058
0059	0059	0059
0060	0060	0060
0061	0061	0061
0062	0062	0062
0063	0063	0063
0064	0064	0064
0065	0065	0065
0066	0066	0066
0067	0067	0067
0068	0068	0068
0069	0069	0069
0070	0070	0070
0071	0071	0071
0072	0072	0072
0073	0073	0073
0074	0074	0074
0075	0075	0075
0076	0076	0076
0077	0077	0077
0078	0078	0078
0079	0079	0079
0080	0080	0080
0081	0081	0081
0082	0082	0082
0083	0083	0083
0084	0084	0084
0085	0085	0085
0086	0086	0086
0087	0087	0087
0088	0088	0088
0089	0089	0089
0090	0090	0090
0091	0091	0091
0092	0092	0092
0093	0093	0093
0094	0094	0094
0095	0095	0095
0096	0096	0096
0097	0097	0097
0098	0098	0098
0099	0099	0099

```

JFETCH PREVIOUS MAX (INTEGER)
IMULT BY (2**15)*(2**11)
JAND NORMALIZE FOR GOOD MEASURE
JII = PREVIOUS MAX (FLT PT)
J-- \ U(EVEN)
JSTORE MAX(ODD) \ MAKE NEW MAX(EVEN)
JLEAP IF ODD NO. OF ELEMENTS IN U
JFETCH U(ODD) \ --
JMAKE NEW MAX(ODD) \ STORE MAX(EVEN)
JLOOP IF MORE DATA IN U
JSTORE MAX(ODD) \ BRING OVER MAX(EVEN)
JSTORE MAX(EVEN)
JRESOLVE MAX(U(ODD),U(EVEN))
IMULT NEW MAX BY (2**15)*(2**11)
JAND CONVERT TO INTEGER
JAND OUTPUT IT
JAND HALT

```

06500 0000
06501 0012

A00	05D2	00F100F1
A01	05D4	22D22220
A02	06D6	32333231
A03	05D0	00910A91
A04	05D0	000000F0
A05	05D0	00971100
A06	06D6	91100A00
A07	05F0	00P00000
A08	05F2	71000091
A09	05F4	90F00000
A10	05F6	00910A90
A11	05F8	00520000
A12	05E4	62200000
A13	05E6	2A700000
A14	05E8	3A9A0000
A15	05E0	009C0000
A16	05F2	20322032
A17	05F4	10000000
A18	05F6	00000012

94596 (02701)

PAGE 64: (END)(<0C016>00016M.MSD.2, 29-00c-00 14130:40, Ed: WOLF
APS3-ADPEAK

```

(02782) APS3-ADPEAK
(02783) J. WOLF, 10/30/68
(02784) B1005 TO APS3-ADPEAK
(02785)
(02786) JWE MAINTAIN THE RUNNING ABS. MAX. AS AN INTEGER IN THE INTEGER
(02787) SCALAR TABLE, FOR CONVENIENCE IN DISPLAYING ITS VALUE TO THE USER.
(02788) BECAUSE SNAP STANDARD BINDING DOESN'T ALLOW FOR INTEGER SCALARS
(02789) AS ARGUMENTS TO ARRAY FUNCTIONS, WE SIMPLY "HARD WIRE" THE
(02790) INTEGER SCALAR OFFSET OF TADPK TO THIS APS PROGRAM.
(02791) TADPK=07
00000057 (02792) TADPK=07
(02793)
(02794) JHEADER BLOCK
(02795)
(02796) EVEN
(02797) ADDR ADPKAS1
(02798) ADDR 0
(02799) DATA 0
(02800) DATA ADPKAS2
(02801) ADDR ADPKAS3
(02802) EVEN
(02803) JINPUT PROGRAM:
(02804) B001 U(0)
(02805) B011 LOOP COUNTER
(02806)
(02807) ADPKAS1
(02808) ADPKAS1: BEGIN APS(ADPKAS1)
(02809) JSM(ADPKAS1,P2)
(02810) SET(R0)
(02811) LOAD(R0R,ISUT$TADPK(1),TE) JGEN ADR FOR TADPK
(02812) R10: LOAD(R0R,C13) JLOAD 0 BASE ADR
(02813) LOAD(R0R,MS) JBASE-SPACING
(02814) SUB(R0R,MS) JGEN U(0)
(02815) R1: ADD(R0R,C9,TF) JLOOP TIL U(0) ELEMENTS ARE DONE
(02816) SOBL(R0R,1),JUMPP(R1)
(02817) CLEAR(R1)
(02818) NOP(0)
(02819)
(02820) JOUTPUT PROGRAM: WRITE NEW MAX BACK IN THE INTEGER SCALAR TABLE
(02821)
(02822) ADPKAS1: SET(RA)
(02823) LOAD(R0R,ISUT$TADPK(1),TE)
(02824) CLEAR(R0)
(02825) NOP(0)
(02826)
(02827) ADPKAS1=EC
(02828) END
00000060 (02829) ADPKAS1: DATA 2F'0.0'
...
00000070 (02830) ADPKAS2 = 0L - ADPKA

```

JPTR TO CONSTRUCTED INSTRUCTION BLOCK
JNO (REAL) SCALARS USED
JAPS MODULE SIZE
JPTR TO CHAIN ANCHOR

JSET AND START OUTPUT PROGRAM

JCHAIN ANCHOR

JSTORAGE FOR CONSTRUCTED INSTRUCTIONS

```

PAGE 65: (BAND)C0C116>R0116H.M50.2, 29-Dec-88 14:38:48, Ed: WOLF
APDMP(V) -- DEBUGGING FUNCTION THAT DUMPS THE APU REGISTERS

(02031) * APDMP(V) -- DEBUGGING FUNCTION THAT DUMPS THE APU REGISTERS
(02032) * J. J. WOLF, 17 JULY 88
(02033) *
(02034) * BUFFER USAGE:
(02035) * V: COMPACT BUFFER, LENGTH 44 FULL WORDS, RECEIVES REGISTER CONTENTS
(02036) *
(02037) * APU PROGRAMS EMPTIES ALL REGISTERS INTO THE OQ
(02038) *
(02039) * EVEN
(02040) * DATA DUMPU$A
(02041) * DATA DUMPU$Z
(02042) *
(02043) * DUMP: BEGIN APU(DUMPU)
(02044) * DUMPU$A:
(02045) * ALL OUTPUTS ARE LEFT, THEN RIGHT
(02046) * OUTPUT R, A0-A7
(02047) * MOV(OQ),R(A0)
(02048) * MOV(OQ),R(A1)
(02049) * MOV(OQ),R(A2)
(02050) * MOV(OQ),R(A3)
(02051) * MOV(OQ),R(A4)
(02052) * MOV(OQ),R(A5)
(02053) * MOV(OQ),R(A6)
(02054) * MOV(OQ),R(A7)
(02055) * MOV(R,OQ)
(02056) * OUTPUT P, THEN 90 M0-M7, M1-M7, M2-M7, M3-M7 SO WE CAN (TRY TO)
(02057) * RECOVER THE CONTENTS OF M7, WHICH WE NEED TO CLOBBER
(02058) * MOV(OQ),MUL(M0,M7)
(02059) * MOV(OQ),MUL(M1,M7)
(02060) * MOV(OQ),MUL(M2,M7)
(02061) * MOV(OQ),MUL(M3,M7)
(02062) * MOV(P,OQ)
(02063) * NOW PUT 1.0 INTO M7 AND MULTIPLY TO OUTPUT M0-M3
(02064) * K(+1)
(02065) * MOV(R,M7)
(02066) * MUL(M0,M7)
(02067) * MOV(OQ),MUL(M1,M7)
(02068) * MOV(OQ),MUL(M2,M7)
(02069) * MOV(OQ),MUL(M3,M7)
(02070) * PUT 1.0 INTO M0 AND MULTIPLY TO OUTPUT M4-M6
(02071) * MOV(R,M0)
(02072) * MUL(M0,M4)
(02073) * MOV(OQ),MUL(M0,M5)
(02074) * MOV(OQ),MUL(M0,M6)
(02075) * MOV(P,OQ)
(02076) * FINALLY, OUTPUT EXI
(02077) * MOV(EXT,OQ)
(02078) * CLEAR(RA)
(02079) * JUMP(0)
(02080) * DUMPU$Z=RA-DUMPU$A
(02081) *
(02082) * END
(02083) * APS MODULE FOR APDMP

0661E 0000
0661F 001C

06620 021C021C
06621 023C023C
06622 025C025C
06623 027C027C
06624 029C029C
06625 02BC02BC
06626 02DC02DC
06627 02FC02FC
06628 030C030C
06629 032C032C
06630 034C034C
06631 036C036C
06632 038C038C
06633 03AC03AC
06634 03CC03CC
06635 03EC03EC
06636 03FC03FC
06637 041C041C
06638 043C043C
06639 045C045C
06640 047C047C
06641 049C049C
06642 04BC04BC
06643 04DC04DC
06644 04FC04FC
06645 051C051C
06646 053C053C
06647 055C055C
06648 057C057C
06649 059C059C
06650 05BC05BC
06651 05DC05DC
06652 05FC05FC
06653 061C061C
06654 063C063C
06655 065C065C
06656 067C067C
06657 069C069C
06658 06BC06BC
06659 06DC06DC
06660 06FC06FC
06661 071C071C
06662 073C073C
06663 075C075C
06664 077C077C
06665 079C079C
06666 07BC07BC
06667 07DC07DC
06668 07FC07FC
06669 081C081C
06670 083C083C
06671 085C085C
06672 087C087C
06673 089C089C
06674 08BC08BC
06675 08DC08DC
06676 08FC08FC
06677 091C091C
06678 093C093C
06679 095C095C
06680 097C097C
06681 099C099C
06682 09BC09BC
06683 09DC09DC
06684 09FC09FC
06685 0A1C0A1C
06686 0A3C0A3C
06687 0A5C0A5C
06688 0A7C0A7C
06689 0A9C0A9C
06690 0ABC0ABC
06691 0ADC0ADC
06692 0AFC0AFC
06693 0B1C0B1C
06694 0B3C0B3C
06695 0B5C0B5C
06696 0B7C0B7C
06697 0B9C0B9C
06698 0BBC0BBC
06699 0BDC0BDC
06700 0BFC0BFC
06701 0C1C0C1C
06702 0C3C0C3C
06703 0C5C0C5C
06704 0C7C0C7C
06705 0C9C0C9C
06706 0CBC0CBC
06707 0CDC0CDC
06708 0CFC0CFC
06709 0D1C0D1C
06710 0D3C0D3C
06711 0D5C0D5C
06712 0D7C0D7C
06713 0D9C0D9C
06714 0DBC0DBC
06715 0DDC0DDC
06716 0DFC0DFC
06717 0E1C0E1C
06718 0E3C0E3C
06719 0E5C0E5C
06720 0E7C0E7C
06721 0E9C0E9C
06722 0EBC0EBC
06723 0EDC0EDC
06724 0EFC0EFC
06725 0F1C0F1C
06726 0F3C0F3C
06727 0F5C0F5C
06728 0F7C0F7C
06729 0F9C0F9C
06730 0FBC0FBC
06731 0FDC0FDC
06732 0FFC0FFC
06733 101C101C
06734 103C103C
06735 105C105C
06736 107C107C
06737 109C109C
06738 10BC10BC
06739 10DC10DC
06740 10FC10FC
06741 111C111C
06742 113C113C
06743 115C115C
06744 117C117C
06745 119C119C
06746 11BC11BC
06747 11DC11DC
06748 11FC11FC
06749 121C121C
06750 123C123C
06751 125C125C
06752 127C127C
06753 129C129C
06754 12BC12BC
06755 12DC12DC
06756 12FC12FC
06757 131C131C
06758 133C133C
06759 135C135C
06760 137C137C
06761 139C139C
06762 13BC13BC
06763 13DC13DC
06764 13FC13FC
06765 141C141C
06766 143C143C
06767 145C145C
06768 147C147C
06769 149C149C
06770 14BC14BC
06771 14DC14DC
06772 14FC14FC
06773 151C151C
06774 153C153C
06775 155C155C
06776 157C157C
06777 159C159C
06778 15BC15BC
06779 15DC15DC
06780 15FC15FC
06781 161C161C
06782 163C163C
06783 165C165C
06784 167C167C
06785 169C169C
06786 16BC16BC
06787 16DC16DC
06788 16FC16FC
06789 171C171C
06790 173C173C
06791 175C175C
06792 177C177C
06793 179C179C
06794 17BC17BC
06795 17DC17DC
06796 17FC17FC
06797 181C181C
06798 183C183C
06799 185C185C
06800 187C187C
06801 189C189C
06802 18BC18BC
06803 18DC18DC
06804 18FC18FC
06805 191C191C
06806 193C193C
06807 195C195C
06808 197C197C
06809 199C199C
06810 19BC19BC
06811 19DC19DC
06812 19FC19FC
06813 1A1C1A1C
06814 1A3C1A3C
06815 1A5C1A5C
06816 1A7C1A7C
06817 1A9C1A9C
06818 1ABC1ABC
06819 1ADC1ADC
06820 1AFC1AFC
06821 1B1C1B1C
06822 1B3C1B3C
06823 1B5C1B5C
06824 1B7C1B7C
06825 1B9C1B9C
06826 1BBC1BBC
06827 1BDC1BDC
06828 1BFC1BFC
06829 1C1C1C1C
06830 1C3C1C3C
06831 1C5C1C5C
06832 1C7C1C7C
06833 1C9C1C9C
06834 1CBC1CBC
06835 1CDC1CDC
06836 1CFC1CFC
06837 1D1C1D1C
06838 1D3C1D3C
06839 1D5C1D5C
06840 1D7C1D7C
06841 1D9C1D9C
06842 1DBC1DBC
06843 1DDC1DDC
06844 1DFC1DFC
06845 1E1C1E1C
06846 1E3C1E3C
06847 1E5C1E5C
06848 1E7C1E7C
06849 1E9C1E9C
06850 1EBC1EBC
06851 1EDC1EDC
06852 1EFC1EFC
06853 1F1C1F1C
06854 1F3C1F3C
06855 1F5C1F5C
06856 1F7C1F7C
06857 1F9C1F9C
06858 1FBC1FBC
06859 1FDC1FDC
06860 1FFC1FFC
06861 201C201C
06862 203C203C
06863 205C205C
06864 207C207C
06865 209C209C
06866 20BC20BC
06867 20DC20DC
06868 20FC20FC
06869 211C211C
06870 213C213C
06871 215C215C
06872 217C217C
06873 219C219C
06874 21BC21BC
06875 21DC21DC
06876 21FC21FC
06877 221C221C
06878 223C223C
06879 225C225C
06880 227C227C
06881 229C229C
06882 22BC22BC
06883 22DC22DC
06884 22FC22FC
06885 231C231C
06886 233C233C
06887 235C235C
06888 237C237C
06889 239C239C
06890 23BC23BC
06891 23DC23DC
06892 23FC23FC
06893 241C241C
06894 243C243C
06895 245C245C
06896 247C247C
06897 249C249C
06898 24BC24BC
06899 24DC24DC
06900 24FC24FC
06901 251C251C
06902 253C253C
06903 255C255C
06904 257C257C
06905 259C259C
06906 25BC25BC
06907 25DC25DC
06908 25FC25FC
06909 261C261C
06910 263C263C
06911 265C265C
06912 267C267C
06913 269C269C
06914 26BC26BC
06915 26DC26DC
06916 26FC26FC
06917 271C271C
06918 273C273C
06919 275C275C
06920 277C277C
06921 279C279C
06922 27BC27BC
06923 27DC27DC
06924 27FC27FC
06925 281C281C
06926 283C283C
06927 285C285C
06928 287C287C
06929 289C289C
06930 28BC28BC
06931 28DC28DC
06932 28FC28FC
06933 291C291C
06934 293C293C
06935 295C295C
06936 297C297C
06937 299C299C
06938 29BC29BC
06939 29DC29DC
06940 29FC29FC
06941 2A1C2A1C
06942 2A3C2A3C
06943 2A5C2A5C
06944 2A7C2A7C
06945 2A9C2A9C
06946 2ABC2ABC
06947 2ADC2ADC
06948 2AFC2AFC
06949 2B1C2B1C
06950 2B3C2B3C
06951 2B5C2B5C
06952 2B7C2B7C
06953 2B9C2B9C
06954 2BBC2BBC
06955 2BDC2BDC
06956 2BFC2BFC
06957 2C1C2C1C
06958 2C3C2C3C
06959 2C5C2C5C
06960 2C7C2C7C
06961 2C9C2C9C
06962 2CBC2CBC
06963 2CDC2CDC
06964 2CFC2CFC
06965 2D1C2D1C
06966 2D3C2D3C
06967 2D5C2D5C
06968 2D7C2D7C
06969 2D9C2D9C
06970 2DBC2DBC
06971 2DDC2DDC
06972 2DFC2DFC
06973 2E1C2E1C
06974 2E3C2E3C
06975 2E5C2E5C
06976 2E7C2E7C
06977 2E9C2E9C
06978 2EBC2EBC
06979 2EDC2EDC
06980 2EFC2EFC
06981 2F1C2F1C
06982 2F3C2F3C
06983 2F5C2F5C
06984 2F7C2F7C
06985 2F9C2F9C
06986 2FBC2FBC
06987 2FDC2FDC
06988 2FFC2FFC
06989 301C301C
06990 303C303C
06991 305C305C
06992 307C307C
06993 309C309C
06994 30BC30BC
06995 30DC30DC
06996 30FC30FC
06997 311C311C
06998 313C313C
06999 315C315C
07000 317C317C
07001 319C319C
07002 31BC31BC
07003 31DC31DC
07004 31FC31FC
07005 321C321C
07006 323C323C
07007 325C325C
07008 327C327C
07009 329C329C
07010 32BC32BC
07011 32DC32DC
07012 32FC32FC
07013 331C331C
07014 333C333C
07015 335C335C
07016 337C337C
07017 339C339C
07018 33BC33BC
07019 33DC33DC
07020 33FC33FC
07021 341C341C
07022 343C343C
07023 345C345C
07024 347C347C
07025 349C349C
07026 34BC34BC
07027 34DC34DC
07028 34FC34FC
07029 351C351C
07030 353C353C
07031 355C355C
07032 357C357C
07033 359C359C
07034 35BC35BC
07035 35DC35DC
07036 35FC35FC
07037 361C361C
07038 363C363C
07039 365C365C
07040 367C367C
07041 369C369C
07042 36BC36BC
07043 36DC36DC
07044 36FC36FC
07045 371C371C
07046 373C373C
07047 375C375C
07048 377C377C
07049 379C379C
07050 37BC37BC
07051 37DC37DC
07052 37FC37FC
07053 381C381C
07054 383C383C
07055 385C385C
07056 387C387C
07057 389C389C
07058 38BC38BC
07059 38DC38DC
07060 38FC38FC
07061 391C391C
07062 393C393C
07063 395C395C
07064 397C397C
07065 399C399C
07066 39BC39BC
07067 39DC39DC
07068 39FC39FC
07069 3A1C3A1C
07070 3A3C3A3C
07071 3A5C3A5C
07072 3A7C3A7C
07073 3A9C3A9C
07074 3ABC3ABC
07075 3ADC3ADC
07076 3AFC3AFC
07077 3B1C3B1C
07078 3B3C3B3C
07079 3B5C3B5C
07080 3B7C3B7C
07081 3B9C3B9C
07082 3BBC3BBC
07083 3BDC3BDC
07084 3BFC3BFC
07085 3C1C3C1C
07086 3C3C3C3C
07087 3C5C3C5C
07088 3C7C3C7C
07089 3C9C3C9C
07090 3CBC3CBC
07091 3CDC3CDC
07092 3CFC3CFC
07093 3D1C3D1C
07094 3D3C3D3C
07095 3D5C3D5C
07096 3D7C3D7C
07097 3D9C3D9C
07098 3DBC3DBC
07099 3DDC3DDC
07100 3DFC3DFC
07101 3E1C3E1C
07102 3E3C3E3C
07103 3E5C3E5C
07104 3E7C3E7C
07105 3E9C3E9C
07106 3EBC3EBC
07107 3EDC3EDC
07108 3EFC3EFC
07109 3F1C3F1C
07110 3F3C3F3C
07111 3F5C3F5C
07112 3F7C3F7C
07113 3F9C3F9C
07114 3FBC3FBC
07115 3FDC3FDC
07116 3FFC3FFC
07117 401C401C
07118 403C403C
07119 405C405C
07120 407C407C
07121 409C409C
07122 40BC40BC
07123 40DC40DC
07124 40FC40FC
07125 411C411C
07126 413C413C
07127 415C415C
07128 417C417C
07129 419C419C
07130 41BC41BC
07131 41DC41DC
07132 41FC41FC
07133 421C421C
07134 423C423C
07135 425C425C
07136 427C427C
07137 429C429C
07138 42BC42BC
07139 42DC42DC
07140 42FC42FC
07141 431C431C
07142 433C433C
07143 435C435C
07144 437C437C
07145 439C439C
07146 43BC43BC
07147 43DC43DC
07148 43FC43FC
07149 441C441C
07150 443C443C
07151 445C445C
07152 447C447C
07153 449C449C
07154 44BC44BC
07155 44DC44DC
07156 44FC44FC
07157 451C451C
07158 453C453C
07159 455C455C
07160 457C457C
07161 459C459C
07162 45BC45BC
07163 45DC45DC
07164 45FC45FC
07165 461C461C
07166 463C463C
07167 465C465C
07168 467C467C
07169 469C469C
07170 46BC46BC
07171 46DC46DC
07172 46FC46FC
07173 471C471C
07174 473C473C
07175 475C475C
07176 477C477C
07177 479C479C
07178 47BC47BC
07179 47DC47DC
07180 47FC47FC
07181 481C481C
07182 483C483C
07183 485C485C
07184 487C487C
07185 489C489C
07186 48BC48BC
07187 48DC48DC
07188 48FC48FC
07189 491C491C
07190 493C493C
07191 495C495C
07192 497C497C
07193 499C499C
07194 49BC49BC
07195 49DC49DC
07196 49FC49FC
07197 4A1C4A1C
07198 4A3C4A3C
07199 4A5C4A5C
07200 4A7C4A7C
07201 4A9C4A9C
07202 4ABC4ABC
07203 4ADC4ADC
07204 4AFC4AFC
07205 4B1C4B1C
07206 4B3C4B3C
07207 4B5C4B5C
07208 4B7C4B7C
07209 4B9C4B9C
07210 4BBC4BBC
07211 4BDC4BDC
07212 4BFC4BFC
07213 4C1C4C1C
07214 4C3C4C3C
07215 4C5C4C5C
07216 4C7C4C7C
07217 4C9C4C9C
07218 4CBC4CBC
07219 4CDC4CDC
07220 4CFC4CFC
07221 4D1C4D1C
07222 4D3C4D3C
07223 4D5C4D5C
07224 4D7C4D7C
07225 4D9C4D9C
07226 4DBC4DBC
07227 4DDC4DDC
07228 4DFC4DFC
07229 4E1C4E1C
07230 4E3C4E3C
07231 4E5C4E5C
07232 4E7C4E7C
07233 4E9C4E9C
07234 4EBC4EBC
07235 4EDC4EDC
07236 4EFC4EFC
07237 4F1C4F1C
07238 4F3C4F3C
07239 4F5C4F5C
07240 4F7C4F7C
07241 4F9C4F9C
07242 4FBC4FBC
07243 4FDC4FDC
07244 4FFC4FFC
07245 501C501C
07246 503C503C
07247 505C505C
07248 507C507C
07249 509C509C
07250 50BC50BC
07251 50DC50DC
07252 50FC50FC
07253 511C511C
07254 513C513C
07255 515C515C
07256 517C517C
07257 519C519C
07258 51BC51BC
07259 51DC51DC
07260 51FC51FC
07261 521C521C
07262 523C523C
07263 525C525C
07264 527C527C
07265 529C529C
07266 52BC52BC
07267 52DC52DC
07268 52FC52FC
07269 531C531C
07270 533C533C
07271 535C535C
07272 537C537C
07273 539C539C
07274 53BC53BC
07275 53DC53DC
07276 53FC53FC
07277 541C541C
07278 543C543C
07279 545C545C
07280 547C547C
07281 549C549C
07282 54BC54BC
07283 54DC54DC
07284 54FC54FC
07285 551C551C
07286 553C553C
07287 555C555C
07288 557C557C
07289 559C559C
07290 55BC55BC
07291 55DC55DC
07292 55FC55FC
07293 561C561C
07294 563C563C
07295 565C565C
07296 567C567C
07297 569C569C
07298 56BC56BC
07299 56DC56DC
07300 56FC56FC
07301 571C571C
07302 573C573C
07303 575C575C
07304 577C577C
07305 579C579C
07306 57BC57BC
07307 57DC57DC
07308 57FC57FC
07309 581C581C
07310 583C583C
07311 585C585C
07312 587C587C
07313 589C589C
07314 58BC58BC
07315 58DC58DC
07316 58FC58FC
07317 591C591C
07318 593C593C
07319 595C595C
07320 597C597C
07321 599C599C
07322 59BC59BC
07323 59DC59DC
07324 59FC59FC
07325 5A1C5A1C
07326 5A3C5A3C
07327 5A5C5A5C
07328 5A7C5A7C
07329 5A9C5A9C
07330 5ABC5ABC
07331 5ADC5ADC
07332 5AFC5AFC
07333 5B1C5B1C
07334 5B3C5B3C
07335 5B5C5B5C
07336 5B7C5B7C
07337 5B9C5B9C
07338 5BBC5BBC
07339 5BDC5BDC
07340 5BFC5BFC
07341 5C1C5C1C
07342 5C3C5C3C
07343 5C5C5C5C
07344 5C7C5C7C
07345 5C9C5C9C
07346 5CBC5CBC
07347 5CDC5CDC
07348 5CFC5CFC
07349 5D1C5D1C
07350 5D3C5D3C
07351 5D5C5D5C
07352 5D7C5D7C
07353 5D9C5D9C
07354 5DBC5DBC
07355 5DDC5DDC
07356 5DFC5DFC
07357 5E1C5E1C
07358 5E3C5E3C
07359 5E5C5E5C
07360 5E7C5E7C
07361 5E9C5E9C
07362 5EBC5EBC
07363 5EDC5EDC
07364 5EFC5EFC
07365 5F1C5F1C
07366 5F3C5F3C
07367 5F5C5F5C
07368 5F7C5F7C
07369 5F9C5F9C
07370 5FBC5FBC
07371 5FDC5FDC
07372 5FFC5FFC
07373 601C601C
07374 603C603C
07375 605C605C
07376 607C607C
07377 609C609C
07378 60BC60BC
07379 60DC60DC
07380 60FC60FC
07381 611C611C
07382 613C613C
07383 615C615C
07384 617C617C
07385 619C619C
07386 61BC61BC
07387 61DC61DC
07388 61FC61FC
07389 621C621C
07390 623C623C
07391 625C625C
07392 627C627C
07393 629C629C
07394 62BC62BC
07395 62DC62DC
07396 62FC62FC
07397 631C631C
07398 633C633C
07399 635C635C
07400 637C637C
07401 639C639C
07402 63BC63BC
07403 63DC63DC
07404 63FC63FC
07405 641C641C
07406 643C643C
07407 645C645C
07408 647C647C
07409 649C649C
07410 64BC64BC
07411 64DC64DC
07412 64FC64FC
07413 651C651C
07414 653C653C
07415 655C655C
07416 657C657C
07417 659C659C
07418 65BC65BC
07419 65DC65DC
07420 65FC65FC
0
```

PAGE 661 (00ND) (00AL16) 00N16H.MSD.2, 29-DEC-80 14130:40, E0: WOLF
 APUMP(V) -- DEBUGGING FUNCTION THAT DUMPS THE APU REGISTERS

```

(02084) )
(02085) )NO INPUT STREAMI
(02086) ) NO: V(0),V(1),...V(43)
(02087) )
(02088) )
(02089) )PTR TO CONST. INSTR. BLOCK
(02090) )PTR TO SCALAR BLOCK
(02091) )NUMBER OF SCALARS
(02092) )MODULE SIZE IN HALFWORDS
(02093) )PTR TO CHAIN ANCHOR
(02094) )
(02095) )
(02096) )APS INPUT PROGRAM
(02097) )
(02098) )DUMPS: BEGIN APS(DUMPS)
(02099) )DUMPSI:
(02100) )JSH(P2,DUMPSO)
(02101) )SET(RO)
(02102) )CLEAR(R1)
(02103) )NOP(0)
(02104) )
(02105) )APS OUTPUT PROGRAM
(02106) )
(02107) )DUMPSO: SET(RA)
(02108) )LOAD(0W0,(0))
(02109) )LOAD(0W1,M$)
(02110) )SUB(0W0,M$)
(02111) )LOAD(0W1,45-1)
(02112) )ADD(0W0,(0),TF)
(02113) )SUBL(0W1,1),JUNPP(01)
(02114) )CLEAR(RO)
(02115) )NOP(0)
(02116) )END
(02117) )DUMPSA=PC
(02118) )DUMPSI: DATA 2F'0.0'
...
0000001E (02119) DUMPSZ=JL-DUMPS
  
```

)SET AND START OUTPUT PCN
)WE DO NO INPUT
)START APU
)YBA = OUTPUT BUFFER
)(UNUSED)
)PREPARE TO INCR
)THERE WILL BE 44 OUTPUTS
)OUTPUT LOOP
)ASSIGN VALUE TO CHAIN ANCHOR
)BLOCK FOR CONSTRUCTED INSTRUCTIONS
)MODULE SIZE

```

(02920) * APUS - HFC(U,V)
(02921) * UMR 0/00
(02922) *
(02923) * ROUTINE TO APPLY HFC(MICH FREQUENCY CORRECTION)
(02924) * TO THE AUTOCORRELATION COEFF.
(02925) *
(02926) *
(02927) *
(02928) *
(02929) *
(02930) *
(02931) *
(02932) *
(02933) *
(02934) *
(02935) *
(02936) *
(02937) *
(02938) *
(02939) *
(02940) *
(02941) *
(02942) *
(02943) *
(02944) *
(02945) *
(02946) *
(02947) *
(02948) *
(02949) *
(02950) *
(02951) *
(02952) *
(02953) *
(02954) *
(02955) *
(02956) *
(02957) *
(02958) *
(02959) *
(02960) *
(02961) *
(02962) *
(02963) *
(02964) *
(02965) *
(02966) *
(02967) *
(02968) *
(02969) *
(02970) *
(02971) *
(02972) *

      = AUTOCORRELATION COEFFICIENTS
      = CONSTANT FACTORS
      = LAMDA*U(1), I=0,1,2
      LAMDA=8.035
      MU(0)=0.375, MU(1)=-0.25, MU(2)=0.0625

      = MPC'D COEFFICIENTS
      = U(1) + LAMDA*MU(1)*E2, I=0,1,2
      WHERE E2=MINIMUM MEAN-SQUARED PREDICTION ERROR
      = U(1), I=3,4,5,6

PROCEDURE
E2 IS COMPUTED FIRST IN THE FOLLOWING FOUR STEPS
  COMPUTE E1=U(1)/U(0)
  COMPUTE E1=(1-E1**2)*U(0)
  COMPUTE E2=U(2)-E1*U(1)/E1
  COMPUTE E2=E1*(1-E2**2)
  THEN THE Y(I)'S ARE COMPUTED AS ABOVE

APU REGISTER USAGE (ADMI ONLY)
  A0 SDIV(U0,E1),U(1)      M0 SDIV(1/U(0),1/E1
  A1 SDIV(SCRATCH)         M1 1-E1**2
  A2 U(2)                  M2 K1,1-E2**2
  A3 E1**2                  M3 K2,E2
  A4 E1*U(1)                M4 SDIV(SCRATCH)
  A5 E2**2,E2*V(1)         M5 U(1),K2
  A6 SDIV(2)                M6 SDIV(U(0),E1)
  A7 1                      M7 K1,U(2)-E1*U(1),V(1)

INPUT AND OUTPUT SEQUENCE
IO: U(0),U(1),U(2),V(0),U(0),V(1),V(2),U(2),U(3),...,U(6)
OO: Y(0),...,Y(6)

HFC - APU INITIALIZATION
EVEN HFCU$A      )START ON FULL WORD BOUNDARY
DATA HFCU$A1     )START ADDR AT SA
DATA HFCU$A2     )SIZE , COMPUTED AT END
HFCU$ BEGIN APU(HFCU) )REFERENCE BEGIN STATEMENT
(02972) *
  
```

0667E 0000
 0667F 0040

PAGE 401 (00ND)(0CAL)00016N.M50.2, 29-Dec-88 14:30:48, Fd1 WOLF
APU3 - WFC(V,U)

```

(02973) * COMPUTE E2 - MINIMUM MEAN-SQUARED PREDICTION ERROR
(02974) * STEP #1 - COMPUTE K1
(02975) * INPUT U(0) - STORE IN M6 AND A0 FOR SOLV
(02976) WFC$B1: MOV(IQ,M6)\NOP
(02977) * CALL SUBROUTINE TO GO SOLV OF U(0)
(02978) * 1/U(0) OUTPUT IN M6
(02979) * CALL(HPCUSDV)
(02980) * COMPUTE K1=U(1)/U(0)
(02981) * COMPUTE K1=U(1)/U(0)
(02982) * COMPUTE K1=U(1)/U(0)
(02983) * COMPUTE K1=U(1)/U(0)
(02984) * STEP #2 - COMPUTE E1
(02985) * COMPUTE 1-K1**2
(02986) * K1**2
(02987) * MOV(R,A7)\NOP
(02988) * MOV(P,M7)\NOP
(02989) * MOV(P,M7)\NOP
(02990) * MOV(P,M7)\NOP
(02991) * MOV(P,M7)\NOP
(02992) * MOV(P,M7)\NOP
(02993) * MOV(P,M7)\NOP
(02994) * COMPUTE E1 = (1-K1**2)*U(0)
(02995) * MUL(M1,M6)\NOP
(02996) * STEP #3 - COMPUTE E2
(02997) * COMPUTE 1/E1 - 1/E1 IN M6
(02998) * MOV(P,A0)\NOP
(02999) * MOV(P,M6)\NOP
(03000) * CALL(HPCUSDV)
(03001) * COMPUTE U(2)=K1*U(1)
(03002) * MUL(M2,M5)\NOP
(03003) * MOV(IQA,A2)\NOP
(03004) * MOV(P,A4)\NOP
(03005) * SUB(A2,A4)\NOP
(03006) * MOV(R,M7)\NOP
(03007) * COMPUTE K2=(U(2)-K1*U(1))/E1
(03008) * MUL(M6,M7)\NOP
(03009) * STEP #4 - COMPUTE E2
(03010) * COMPUTE K2**2
(03011) * MOV(P,M3)\NOP
(03012) * MOV(P,M5)\NOP
(03013) * MUL(M3,M5)\NOP
(03014) * MOV(P,A5)\NOP
(03015) * COMPUTE 1-K2**2
(03016) * SUB(A7,A5)\NOP
(03017) * MOV(R,M2)\NOP
(03018) * COMPUTE E2
(03019) * MUL(M2,M6)\NOP
(03020) * MOV(P,M3)\NOP
(03021) * END OF E2 COMPUTATION
(03022) * COMPUTE HIGH FREQUENCY CORRECTED
(03023) * AUTOCORRELATION COEFF'S AND OUTPUT
(03024) *
(03025) *

```


PAGE 71: [00ND1<0C0116>00N16M.MSO.2, 29-Dec-88 14:38:48, Ed: WOLF
 APS3 - HFC(V,U,V)

A0C 06720 10AA002 (03124)	ADD(BR2,C103,TF)	JGEN V(1) ADDR
A0D 06722 1A0A902 (03125)	OUTPUT U(1),V(2) SEQUENCE	JGEN U(1) ADDR
A0E 06724 1CAAA02 (03126)	ADD(BR0,C93,TF)	JGEN V(2) ADDR
A0F 06726 1E50003 (03127)	ADD(BR2,C103,TF)	JSET UP COUNTER
A10 06728 20A1904 (03128)	GEN REST OF U(1) ADDRS	JGEN U(1) ADDR
A11 0672A 2219101 (03129)	LOAD(BR1,3)	JDEC INDEX AND JUMP
A12 0672C 240A904 (03130)	WPCS\$3: ADD(BR0,C93,TF)	JGEN U(6) ADDR
	WPCS\$4: SUBL(BR1,1),JUMPP(HFCS\$3)	
	HPCS\$4: ADD(BR0,C93,TF)	
A13 0672E 2620031 (03131)	END OF APS INPUT PROGRAM	JHALT INPUT
A14 06730 2000020 (03132)	CLEAR(RI)	
	MOP(0)	
	(03133)	
	(03134)	
	(03135)	
	(03136)	
	(03137)	
	(03138)	
	(03139)	
	OUTPUT APS PROGRAM	
	(03140)	
	REGISTER USAGE	
	(03141)	
	BR0 Y ADDR	
	(03142)	
	BR1 Y SIZE (UNUSED)	
	(03143)	
A15 06732 2A30032 (03144)	HPCS\$OP: SET(RA)	JTURN ON APU
	(03145)	
	(03146)	
A16 06734 2CC0000 (03147)	LOAD AND GEN Y ADDRS	JLOAD AND GEN Y(0)
A17 06736 2E50000 (03148)	HPCS\$5: LOAD(BR0,C03,TF)	JVBS-1 (UNUSED)
A18 06738 300A000 (03149)	LOAD(BR1,M\$)	JGEN Y(1) ADDR
	ADD(BR0,M\$5,TF)	
	(03150)	
	GEN REST OF Y ADDRS	
A19 0673A 3250003 (03151)	LOAD(BR1,3)	JSET UP COUNTER
A1A 0673C 340A000 (03152)	WPCS\$6: ADD(BR0,C03,TF)	JGEN U(1) ADDR
A1B 0673E 36111A1 (03153)	HPCS\$6: SUBL(BR1,1),JUMPP(HFCS\$6)	JDEC INDEX AND JUMP
A1C 06740 380A004 (03154)	HPCS\$7: ADD(BR0,C03,TF)	JGEN Y(6) ADDR
	(03155)	
A1D 06742 3A20030 (03156)	END OF APS OUTPUT PROGRAM	JHALT OUTPUT
A1E 06744 3C00020 (03157)	CLEAR(RO)	
	MOP(0)	
	(03158)	
	(03159)	
	HPCS\$A=BC	JCHAIN ANCHOR
	(03160)	
	(03161)	
06746	END	
	(03162)	
	(03163)	
06746 0000000 (03164)	HPCS\$I DATA 14F'0.0'	
...		
0000005A (03165)	HPCS\$Z=BL-HFCS\$	JMODULE SIZE

PAGE 72: (RDMO)DCAL6>BBM16M.WSO.2, 29-Dec-88 14:38:48, Fdi WOLF
APU3 - GAIN(Y,A,U,B,V,C)

```

(03166) * APU3 - GAIN(Y,A,U,B,V,C)
(03167) * WHR 9/88
(03168) *
(03169) * ROUTINE TO COMPUTE QUANTIZER SEGMENT
(03170) * SCALE FACTORS. DELTA GAINS AND OVERALL GAIN
(03171) * ARE ALSO COMPUTED AND QUANTIZED FOR YX.
(03172) *
(03173) * INPUTS:
(03174) * V(1),...,V(216) = SECOND RESIDUAL SAMPLES
(03175) * B = CONSTANT = 1/72
(03176) * C = CONSTANT = 1/3
(03177) *
(03178) * OUTPUTS:
(03179) * U(0),U(1),U(2) = CODED SEGMENT DELTA GAINS
(03180) * Y(0),Y(1),...,Y(5) = ALTERNATING SCALE FACTORS
(03181) * AND 1/SCALE FACTORS
(03182) * A = CODED OVERALL GAIN , G
(03183) *
PROCEDURE
  COMPUTE SUM(J) , J=1,2,3
  SUM(J)=SUM(V(1+N*72)**2),I=1,...72,N=0,1,2
  COMPUTE SG(J)=1/72*SUM(J) , J=1,2,3
  COMPUTE G = OVERALL GAIN
  QUANTIZE G => GM - OUTPUT CODED G => SA
  COMPUTE CH*CH
  COMPUTE 1/(CH*CH)
  COMPUTE DG(1)=SG(1)/(CH*CH)
  QUANTIZE DG(1) => DGH(U) - OUTPUT CODED DG => U(0,1,2)
  COMPUTE SCALE FACTOR Y(1)=CH*DGH(1) => Y(0,2,4)
  COMPUTE 1/SCALE FACTOR => Y(1,3,5)

INPUT AND OUTPUT SEQUENCE
  IQ: SB,SC,V(1),...,V(72),CPW13,V(73),...,V(144),CPW13,
  V(145),...,V(216),CPW13,QUANT AND CODING TABLES
  FOR G , DG(1) , DG(2) , DG(3)
  OQ: SA,U(0),Y(0),Y(1),U(1),Y(2),Y(3),U(2),Y(4),Y(5)

APU REGISTER USAGE
  A0 SUM**2,HC**2,SDIV,Y(1) M0 SUM**2,SUM(SG(1)),CH,SDIV
  A1 SC(1) M1 SUM(1),SG(1),DGH(1)
  A2 SC(2) M2 SUM(2),SG(2),DGH(2)
  A3 SC(3) M3 SUM**2,SUM(3),SG(3),DGH(3)
  A4 SC(1)*SG(2),SDIV M4 SUM**2,SDIV
  A5 QUANT M5 1/72,CH
  A6 QUANT,SDIV M6 SUM**2,CH**2,SDIV,Y(1)
  A7 SUM**2,QUANT M7 1/3,1/(CH*CH)

GAIN - APU INITIALIZATION
  EVEN DATA GAIN$A
  )START ON FULL WORD BOUNDARY
  )START ADDR AT SA

```

06762 0000

PAGE 73: (0000)<0CA16>00016M.WSD.2, 29-Dec-88 14:30:40, Ed: WMLF
APU3 - GAIN(V,A,M,R,N,C)

	06763 0004	(03219)	DATA	GAINSSZ		SIZE, COMPUTED AT END
	(03220)					
	(03221)		GAINSS: BEGIN APO(GAIN)			
	(03222)		INPUT CONSTANTS 1/72 AND 1/3			
A00	06764 00E00000	(03223)	GAINSSA: MOV(IQA,M5) \ NOP			
A01	06766 00E00000	(03224)	MOV(IQA,M7) \ NOP			
	(03225)		COMPUTE SUM OF SQUARED SAMPLES			
	(03226)		SEGMENT #1			
A02	06768 30000061	(03227)	CALL(GAINSS)			
A03	0676A 20372037	(03228)	CLEAR(VI)			
A04	0676C 00090000	(03229)	MOV(R,M1) \ NOP			
	(03230)		SEGMENT #2			
A05	0676E 30000061	(03231)	CALL(GAINSS)			
A06	06770 20372037	(03232)	CLEAR(VI)			
A07	06772 000A0000	(03233)	MOV(R,M2) \ NOP			
	(03234)		SEGMENT #3			
A08	06774 30000061	(03235)	CALL(GAINSS)			
A09	06776 20372037	(03236)	CLEAR(VI)			
A0A	06778 00000000	(03237)	MOV(R,M3) \ NOP			
	(03238)		COMPUTE SC'S - SC(1)=1/72*SUM(1)			
	(03239)		SEGMENT #1			
A0B	0677A 04A00000	(03240)	MUL(M1,M5) \ NOP			
A0C	0677C 00010000	(03241)	MOV(P,A1) \ NOP			
A0D	0677E 00A90000	(03242)	MOV(P,M1) \ NOP			
	(03243)		SEGMENT #2			
A0E	06780 05200000	(03244)	MUL(M2,M5) \ NOP			
A0F	06782 00200000	(03245)	MOV(P,A2) \ NOP			
A10	06784 00AA0000	(03246)	MOV(P,M2) \ NOP			
	(03247)		SEGMENT #3			
A11	06786 05A00000	(03248)	MUL(M3,M5) \ NOP			
A12	06788 00030000	(03249)	MOV(P,A3) \ NOP			
A13	0678A 00AD0000	(03250)	MOV(P,M3) \ NOP			
	(03251)		COMPUTE OVERALL GAIN - G			
A14	0678C 42000000	(03252)	ADD(A1,A2) \ NOP			
A15	0678E 00940000	(03253)	MOV(R,A4) \ NOP			
A16	06790 44000000	(03254)	ADD(A3,A4) \ NOP			
A17	06792 00000000	(03255)	MOV(R,M0) \ NOP			
A18	06794 04000000	(03256)	MUL(M0,M7) \ NOP			
	(03257)		QUANTIZE OVERALL GAIN G = CH			
A19	06796 30000052	(03258)	CALL(GAINSSQ)			
A1A	06798 0000009C	(03259)	NOP \ MOV(R,00)			
	(03260)		COMPUTE GH**2			
A1B	0679A 00000000	(03261)	MOV(R,M0) \ NOP			
A1C	0679C 00000000	(03262)	MOV(R,M5) \ NOP			
A1D	0679E 04200000	(03263)	MUL(M0,M5) \ NOP			
	(03264)		COMPUTE 1/(GH*GH)			
A1E	067A0 00AF0000	(03265)	MOV(P,M6) \ NOP			
A1F	067A2 00000000	(03266)	MOV(P,M0) \ NOP			
A20	067A4 30000042	(03267)	CALL(GAINSDV)			
A21	067A6 00AF0000	(03268)	MOV(P,M7) \ NOP			
	(03269)		SEGMENT #1			
	(03270)		COMPUTE DC			
A22	067A8 04E00000	(03271)	MUL(M1,M7) \ NOP			

A23 067AA 30000052	(03272) * QUANTIZE DG	CALL(GAIN\$Q)	QUANTIZE DG
A24 067AC 0000009C	(03273) * NOP \ MOV(R,0Q)	NOP \ MOV(R,0Q)	OUTPUT CODED DG
A25 067AE 00090000	(03274) * MOV(R,M1) \ NOP	MOV(R,M1) \ NOP	M1=DCH(1)
A26 067B0 04A00000	(03275) * COMPUTE SCALE FACTOR V(1)	COMPUTE SCALE FACTOR V(1)	
A27 067B2 000C0000	(03276) * MUL(M1,M5) \ NOP	MUL(M1,M5) \ NOP	P=DCH(1)*CH
A28 067B4 00AF0000	(03277) * MOV(P,0Q) \ NOP	MOV(P,0Q) \ NOP	OUTPUT V(1)
A29 067B6 00000000	(03278) * COMPUTE 1/V(1)	COMPUTE 1/V(1)	
A2A 067B8 00AF0000	(03279) * MOV(P,M6) \ NOP	MOV(P,M6) \ NOP	M6=V(1)
A2B 067BA 00000000	(03280) * MOV(P,A0) \ NOP	MOV(P,A0) \ NOP	A0=V(1)
A2C 067BC 00000042	(03281) * CALL(GAIN\$DV)	CALL(GAIN\$DV)	
A2D 067BE 000C0000	(03282) * MOV(P,0Q) \ NOP	MOV(P,0Q) \ NOP	SDIV - 1/V(1)
A2E 067BF 00000000	(03283) * SEGMENT #2		OUTPUT 1/V(1)
A2F 067C0 00000000	(03284) * COMPUTE DG	COMPUTE DG	
A30 067C2 000A0000	(03285) * MUL(M2,M7) \ NOP	MUL(M2,M7) \ NOP	
A31 067C4 05200000	(03286) * QUANTIZE DG	QUANTIZE DG	P=SG(2)/CH**2
A32 067C6 00000000	(03287) * CALL(GAIN\$Q)	CALL(GAIN\$Q)	
A33 067C8 00AF0000	(03288) * NOP \ MOV(R,0Q)	NOP \ MOV(R,0Q)	QUANTIZE DG
A34 067CA 00000000	(03289) * MOV(R,M2) \ NOP	MOV(R,M2) \ NOP	OUTPUT CODED DG
A35 067CB 00000000	(03290) * COMPUTE SCALE FACTOR V(2)	COMPUTE SCALE FACTOR V(2)	M2=DCH(2)
A36 067CD 05200000	(03291) * MUL(M2,M5) \ NOP	MUL(M2,M5) \ NOP	
A37 067CE 000C0000	(03292) * MOV(P,0Q) \ NOP	MOV(P,0Q) \ NOP	P=DCH(2)*CH
A38 067CF 00AF0000	(03293) * COMPUTE 1/V(2)	COMPUTE 1/V(2)	OUTPUT V(2)
A39 067D0 00000000	(03294) * MOV(P,M6) \ NOP	MOV(P,M6) \ NOP	M6=V(2)
A3A 067D2 00000000	(03295) * MOV(P,A0) \ NOP	MOV(P,A0) \ NOP	A0=V(2)
A3B 067D4 00000042	(03296) * CALL(GAIN\$DV)	CALL(GAIN\$DV)	
A3C 067D6 000C0000	(03297) * MOV(P,0Q) \ NOP	MOV(P,0Q) \ NOP	SDIV - 1/V(2)
A3D 067D8 00000000	(03298) * SEGMENT #3		OUTPUT 1/V(2)
A3E 067DA 05200000	(03299) * COMPUTE DG	COMPUTE DG	
A3F 067DC 00000000	(03300) * MUL(M3,M7) \ NOP	MUL(M3,M7) \ NOP	
A40 067DE 00000000	(03301) * QUANTIZE DG	QUANTIZE DG	P=SG(3)/CH**2
A41 067DF 00000052	(03302) * CALL(GAIN\$Q)	CALL(GAIN\$Q)	
A42 067E0 0000009C	(03303) * NOP \ MOV(R,0Q)	NOP \ MOV(R,0Q)	QUANTIZE DG
A43 067E2 00000000	(03304) * MOV(R,M3) \ NOP	MOV(R,M3) \ NOP	OUTPUT CODED DG
A44 067E4 00000000	(03305) * COMPUTE SCALE FACTOR V(3)	COMPUTE SCALE FACTOR V(3)	M3=DCH(3)
A45 067E6 05A00000	(03306) * MUL(M3,M5) \ NOP	MUL(M3,M5) \ NOP	
A46 067E8 000C0000	(03307) * MOV(P,0Q) \ NOP	MOV(P,0Q) \ NOP	P=DCH(3)*CH
A47 067EA 00AF0000	(03308) * COMPUTE 1/V(3)	COMPUTE 1/V(3)	OUTPUT V(3)
A48 067EC 00000000	(03309) * MOV(P,M6) \ NOP	MOV(P,M6) \ NOP	M6=V(3)
A49 067EE 00000000	(03310) * MOV(P,A0) \ NOP	MOV(P,A0) \ NOP	A0=V(3)
A4A 067F0 00000042	(03311) * CALL(GAIN\$DV)	CALL(GAIN\$DV)	
A4B 067F2 000C0000	(03312) * MOV(P,0Q) \ NOP	MOV(P,0Q) \ NOP	SDIV - 1/V(3)
A4C 067F4 20322032	(03313) * CLEAR(RA)	CLEAR(RA)	OUTPUT 1/V(3)
A4D 067F6 10000000	(03314) * JUMP(0)	JUMP(0)	
A4E 067F8 00000000	(03315) * EJECT	EJECT	
A4F 067FA 00000000	(03316) * HALT APU	HALT APU	

```

PAGE 75: (00001)DC116>000100.2, 29-00c-00 14:30:40, Ed: WOLY
APU3 - GAIN(V,A,B,B,V,C)

(03323) * * * GAINSDV - DIVIDE SUBROUTINE - COMPUTES 1/C
(03324) * C IS ASSUMED TO BE IN M6 AND A0
(03325) *
A42 06700 16A00000 GAINSDV: R(2)\NOP
A43 0670A 00960000 MOV(R,A6)\NOP
A44 0670C 2E000000 RCP(A6)\NOP
A45 0670E 00000000 MOV(R,M6)\NOP
A46 06710 04400000 MUL(M6,M6)\NOP
A47 06712 00040000 MOV(P,A4)\NOP
A48 06714 4CC00000 SUB(A6,A4)\NOP
A49 06716 000C0000 MOV(R,M4)\NOP
A4A 06718 04000000 MUL(M6,M4)\NOP
A4B 0671A 00A00000 MOV(P,M6)\NOP
A4C 0671C 04400000 MUL(M6,M6)\NOP
A4D 0671E 00040000 MOV(P,A4)\NOP
A4E 06720 4CC00000 SUB(A6,A4)\NOP
A4F 06722 000C0000 MOV(R,M4)\NOP
A50 06724 04000000 MUL(M6,M4)\NOP
A51 06806 1000A000 RESULTS =1/C IN P1
      RETURN
      EJECT
(03326) *
(03327) *
(03328) *
(03329) *
(03330) *
(03331) *
(03332) *
(03333) *
(03334) *
(03335) *
(03336) *
(03337) *
(03338) *
(03339) *
(03340) *
(03341) *
(03342) *
(03343) *
(03344) *
R0=2.0
R1=1/C*0.001(-P0)
P=P0-C
A4=P0-C
R0=2-P0-C
R4=R
P=P0-R(-P1)
R0=P1
P=P1-C
A4=P
R0=2-P1-C
R4=R
P=P1*(2-P1-C)(-P2)

```

PAGE 76: (00ND1)DC116>00N16M.N50.2, 29-04c-00 14130:40, 2d1 WOLF
AP03 - GAIN(V,A,W,B,V,C)

```

(03345) * * * GAIN50 - QUANTIZATION SUBROUTINE
(03346) *
A52 06000 00000076 (03347) GAIN50: NOP \ MOV(IQA,A6)          JIMPUT 2*-13
A53 06004 00050015 (03348) * POT DATA IN A5 AND INIT CODE
A54 0600C 000002A0 (03349) * MOV(P,A5)\MOV(ZERO,A5)      JIMPUT DATA \ INIT CODE=0
A55 0600E 00F60000 (03350) * MOV(A5)                      JIMPUT A5=ZERO
A56 06010 4F104085 (03351) * INNER LOOP - FIND QUANT VALUE AND CODE
A57 06012 00F70000 (03352) * GAIN50: MOV(IQA,A6)\NOP      JIMPUT BOUNDARY - TM(N)\NOP
A58 06014 00E000E0 (03353) * MOV(A5,A6)\MOV(A5),ADD(A5,A6)  J(X-TM(N))\UPDATE CODE, CODE+1
A59 06016 9116005F (03354) * MOV(IQA,A7)\NOP              JIMPUT QUANTIZED VALUE L(N)\NOP
A60 0601A 08E000E0 (03355) * MOV(R,NULL)\NOP              JENSURE TBI SET
A61 0601C 00000000 (03356) * JUMPS(GAIN503,TBI)          JUMP IF DONE TABLE
A62 0601E 901E0055 (03357) * JUMPC(GAIN501,T1)             JUMP IF X > TM (KEEP LOOKING)
A63 06020 20372037 (03358) * FUND0 THE RIGHT QUANTIZATION DIM. READ IN REST OF TABLE.
A64 06022 02E03A20 (03359) * GAIN502: MOV(IQA,NULL)        JIMPUT REST OF TABLE
A65 06024 10001000 (03360) * NOP                          JTO ALLOW FBI TO SET
A66 06026 02E03A20 (03361) * JUMPC(GAIN502,TBI)          JUMP OUT IF FBI IS SET
A67 06028 02E03A20 (03362) * TABLE FINISHED - OUTPUT DATA
A68 0602A 10001000 (03363) * GAIN503: CLEAR(NT)            J LET APS INPUT PRC PROCEED
A69 0602C 00000000 (03364) * R(A7)\ALIGN(A5)              JQUANT VALUE\PRC CODE VALUE
A70 0602E 00000000 (03365) * QUANTIZED VALUE IN R1 - CODE IN R2
A71 06030 00000000 (03366) * RETURN                        JRETURN FROM QUANTIZATION
A72 06032 00000000 (03367) *
A73 06034 00000000 (03368) *
A74 06036 00000000 (03369) *
EJECT

```


0606C (03423) • (03424) • END

PAGE 01: (DBWD)<DCA16>00N16M.NSO.2, 29-Dec-88 14:38:48, 24: WOLF
 APS3 - GAIN(V,A,W,B,V,C)

```

(03526) *
(03527) * OUTPUT APS PROGRAM
(03528) *
(03529) * REGISTER USAGE
(03530) *   BW0  SA, V BASE ADDR
(03531) *   BW1  W BASE ADDR
(03532) *   BW3  W SIZE-1
(03533) *
(03534) * GAINSS0: SET(RA)
A32 06000 64300032
(03535) *
(03536) * GENERATE SCALAR A ADDR
A33 0600A 60010010
(03537) *   MOVW(BW0,BW0,TF)
(03538) *
(03539) * LOAD SCALE FACTOR & INVERSE ADDR
A34 060DC 60400050
(03540) *   LOAD(BW0,C0)
A35 060DE 6A700000
(03541) *   LOAD(BW3,M$)
A36 060E0 6C020000
(03542) *   SUB(BW0,M$)
(03543) *
(03544) * LOAD DELTA GAIN CODE ADDR
A37 060E2 6E501006
(03545) *   LOAD(BW1,C12)
A38 060E4 70700000
(03546) *   LOAD(BW3,M$)
A39 060E6 72120000
(03547) *   SUB(BW1,M$)
(03548) * LOOP FOR OUTPUT OF W AND V
A3A 060E8 751A9006
(03549) *   ADD(BW1,C9),T2
A3B 060EA 760A0002
(03550) *   ADD(BW0,C0),TF
A3C 060EC 700A0002
(03551) *   ADD(BW0,C0),TF
A3D 060EE 7A313A01
(03552) *   SUBL(BW3,1),JUMPP(01)
A3E 060F0 7C200030
(03553) *   CLEAR(R0)
A3F 060F2 7E000020
(03554) *   NOP(0)
(03555) *
(03556) * GAINSSA=BC
000060EC
(03557) *
060F4
(03558) *   END
060F4 00000000
(03559) * GAINSS1 DATA 7F'0.0'
...
(03561) *
0000000E
(03562) * GAINSS2=BL-GAINSS

```

TURN ON APW

JGEN SA ADDR
)(CODED OVERALL GAIN)

W BASE ADDR
)(UNUSPD)
)PREPARE TO INCR LATER

W BASE ADDR
)BDS-1
)PREPARE TO INCR LATER

OUTPUT ADDR OF CODED DC
)OUTPUT ADDR OF SCALE FAC
)OUTPUT ADDR OF 1/SCALE FAC
)DEC COUNT AND JUMP
)DONE

JCHAIN ANCHOR

MODULE SIZE

```

(03563) * SPECIAL SUPPORT MODULES FOR FEOP & FEOT
(03564) * FORWARD & INVERSE EVEN-ODD SEPARATE MODULES.
(03565) * ALSO CALLED BY MOST FOR PPTMR AND PPTMR.
(03566) * (ALSO CALLED BY MOST FOR PPTLR AND PPTLR.)
(03567) * 15 MAY 79
(03568) * J F BURNS
(03569) *
(03570) * ENTER HERE FOR FEOT
(03571) *
(03572) *
(03573) * FEOPSSM MOVIR R6, PLSSET + PLSGO SET-GO LITERAL
(03574) * NOP FEOP31 & SAVE IT
(03575) * EVEN
(03576) *
(03577) * ENTER HERE FOR FEOP
(03578) *
(03579) * FEOPSSM MOVIR R6, PLSCLR + PLSGO CLR-GO LITERAL
(03580) * & SAVE IT
(03581) *
(03582) * COMMON PATH FOR FEOS SET-UP
(03583) *
(03584) * FEOP31 MOVIR R4, 4 * H$ (R1) GET W-ID
(03585) * ANDIR R4, MSKSLAYT CHVT TO BCT INDEX
(03586) * LRS R4, 7
(03587) * EVEN
(03588) * MOVIR R5, DCT3AT + H$ (R4) GET PVR-OF-2 BS
(03589) * INCR R5, 1 CHVT TO REAL BS
(03590) * LLS R5, 2 CHVT TO COS TOL INDEX
(03591) * MOVIR R4, COS$ ADDR OF APS CONST
(03592) * PUSHML R4, SINCONST (R5) LOAD .SCOS(PI/N)
(03593) * PUSHML R4, SINCONST + W$ (R5) LOAD .SSIN(PI/N)
(03594) *
(03595) * CALL R1, AP$BNDR NORMAL BND, BUT GO = 0
(03596) *
(03597) * MOVIR R6, AP$GO (R3) SET OR CLEAR GO CTRL
(03598) *
(03599) * RETURN
(03600) *
(03601) * EVEN

```

222

PAGE 04: (RUND)DCAL16>BHW16H.M50.2, 29-Dec-88 14:38:48, Ed: WOLF
APU3 - FEOSL FOURIER EVEN-ODD SEPARATE LONG (MAP-388)

```

A1A 06954 00F00F0 (03655)      MOV(TQA,A0)      A0=R(K)
A1B 06956 00F100F1 (03656)      MOV(TQA,A1)      A1=R(N-K)
A1C 06958 41004000 (03657)      ADD(A0,A1) \ SUB(A0,A1)  R=2UR\2VI
A1D 0695A 00F700F2 (03658)      MOV(TQA,A2)      A2=IK
A1E 0695C 00F300F3 (03659)      MOV(TQA,A3)      A3=I(N-K)
A1F 0695E 0510535 (03660)      MOV(A5),NUL(M2,M4) \ MOV(A5),NUL(M2,M5)
A20 06960 00940009 (03661) \ A5=S*SIN(K-1)\S*COS(K-1)
A21 06962 43504050 (03662)      MOV(R,A4) \ MOV(R,M1)  A4=2UR\2VI
A22 06964 00490054 (03663)      MOV(EXO),ADD(A2,A3) \ MOV(EXO),SUB(A2,A3)
A23 06966 00800097 (03664) \ EXO=2UR\2VI
A24 06968 00860086 (03665)      MOV(EXT,M1) \ MOV(EXT,A4)
A25 0696A 400R4500 (03666) \ MOV(R,M0) \ MOV(R,A7)
A26 0696C 00570040 (03667)      MOV(P,A6)
A27 0696E 008C0080 (03668)      MOV(EXO),SUB(A6,A5) \ MOV(EXO),ADD(A6,A5)
A28 06970 04000400 (03669) \ EXO=2UR\20F
A29 06972 00980098 (03670) \ MOV(EXT,A7) \ MOV(EXT,M0)
A2A 06974 0040004C (03671)      MOV(R,M4) \ MOV(R,M5)  A7=20I\MB=2VR
A2B 06976 00800080 (03672)      NUL(M0,M4) \ NUL(M1,M5)  M4=COS\MS=SIN
A2C 06978 04300491 (03673)      MOV(R,EXO) \ MOV(EXT,M4) \ MOV(EXT,M5)
A2D 0697A 00510050 (03674)      MOV(EXT,M5) \ MOV(EXT,M4)
A2E 0697C 49004900 (03675) \ MOV(P,EXO)
A2F 0697E 42924192 (03676) \ MOV(M0),NUL(M0,M5) \ MOV(A1),NUL(M1,M4)
A30 06980 00800081 (03677) \ MOV(EXT,A1) \ MOV(EXT,A0)
A31 06982 00800080 (03678) \ SUB(A0,A1)
A32 06984 00510050 (03679) \ MOV(A2),ADD(A4,A2) \ MOV(A2),SUB(A4,A2)
A33 06986 411C411C (03680) \ MOV(P,A0) \ MOV(P,A1)
A34 06988 41P242P2 (03681) \ MOV(P,EXO)
A35 0698A 07520A52 (03682) \ MOV(EXT,A1) \ MOV(EXT,A0)
A36 0698C 009C009C (03683) \ MOV(OQ),ADD(A0,A1) \ MOV(OQ),ADD(A0,A1)
A37 0698E 901C0019 (03684) \ MOV(A2),SUB(A7,A2) \ MOV(A2),ADD(A7,A2)
A38 06990 00600060 (03685) \ MOV(A2),R(A2) \ MOV(A2),NEG(A2)
A39 06992 10000059 (03686) \ MOV(R,OQ) \ MOV(R,OQ)
A40 06994 05A00500 (03687) \ JUMPC(FEOSL$2,EO)
A41 06996 00F000F0 (03688) \ NOP
A42 06998 00F100F1 (03689) \ JUMPC(FEOSL$4)
A43 0699A 41004000 (03690) \ MAIN LOOP FOR INVERSE (K=1,2,...,N/2)
A44 0699C 00F200F2 (03691) \ FEOSL$3
A45 0699E 00F100F3 (03692) \ MUL(M3,M5) \ MUL(M3,M4)
A46 0699F 00F000F0 (03693) \ MOV(TQA,A0)
A47 0699A 41004000 (03694) \ MOV(TQA,A1)
A48 0699C 00F200F2 (03695) \ ADD(A0,A1) \ SUB(A0,A1)
A49 0699E 00F100F3 (03696) \ MOV(TQA,A2)
A50 0699F 00F000F0 (03697) \ MOV(TQA,A3)
A51 0699A 41004000 (03698) \ S*SIN(K-1)\S*COS(K-1)
A52 0699C 00F200F2 (03699) \ A0=R(K)
A53 0699E 00F100F3 (03700) \ A1=R(N-K)
A54 0699F 00F000F0 (03701) \ R=2UR\2VI
A55 0699A 41004000 (03702) \ A2=IK
A56 0699C 00F200F2 (03703) \ A3=I(N-K)

```

PAGE 05: [00RDJ] <DCAL16>RHWIGH.MSO.2, 29-Dec-88 14:30:46, Ed: WOLF
APUJ - FEOSL FOURIER EVEN-ODD SEPARATE LONG (MAP-300)

```

A40 069A0 05150535 (03700)      MOV(A5),MUL(M2,M4) \ MOV(A5),MUL(M2,M5)
                                A5=S*SIN(K-1)\S*COS(K-1)
A41 069A2 009A0009 (03710)      MOV(R,A4) \ MOV(R,M1)
                                A4=20R\2VI
A42 069A4 43504050 (03711)      MOV(EXT),ADD(A2,A3) \ MOV(EXT),SUB(A2,A3)
                                EXT=20R\2VI
A43 069A6 00490054 (03712)      MOV(EXT,M1) \ MOV(EXT,A4)
                                M1=2VI\A4=20R
                                M0=20R\A7=2VI
                                A6=C*COS(K-1)\C*SIN(K-1)
A44 069A8 00000097 (03713)      MOV(R,M0) \ MOV(R,A7)
                                MOV(P,A6)
                                MOV(EXT),ADD(A6,A5) \ MOV(EXT),SUB(A6,A5)
                                EXT=20R\2VI
A45 069A1 00000086 (03714)      MOV(EXT,A7) \ MOV(EXT,M0)
                                MOV(R,M4) \ MOV(R,M5)
                                MUL(M0,M4) \ MUL(M1,M5)
                                MOV(R,EXT)
                                MOV(EXT,M5) \ MOV(EXT,M4)
                                M5=SIN\M4=COS
                                EXT=20R*COS\2VI*SIN
                                MOV(A0),MUL(M0,M5) \ MOV(A1),MUL(M1,M4)
                                A0=20R*COS\A1=2VI*SIN
                                MOV(EXT,A1) \ MOV(EXT,A0)
                                SUB(A0,A1)
                                MOV(A2),ADD(A4,A2) \ MOV(A2),SUB(A4,A2)
                                A2=20R*COS-2VISIN
                                MOV(P,A0) \ MOV(P,A1)
                                MOV(P,EXT)
                                MOV(EXT,A1) \ MOV(EXT,A0)
                                MOV(OQ),ADD(A0,A1) \ MOV(OQ),ADD(A0,A1)
                                OUT=20R(K)\2VI(K)
                                MOV(A2),SUB(A7,A2) \ MOV(A2),ADD(A7,A2)
                                A2=20R*SIN\A1=2VI*COS
                                MOV(A2),R(A2) \ MOV(A2),NEG(A2)
                                MOV(R,OQ) \ MOV(R,OQ)
                                JUMPC(FEOSL$3,EO)
                                CLEAR(RA)
                                MOP
                                JUMP(0)
                                MOP
                                MOP
                                MOP
                                MOP
                                MOP
                                MOP
                                JUMP(FEOSL$3A)
                                FEOSL$2=RA-FEOSL$3A
                                END FEOSL$32
                                EVEN

```


PAGE 07: C00WD1C0CALC080W16M.MSD.2, 29-Dec-80 14:30:40, E4: WOLF
APS3 - F11AL APS PROGRAM FOR FEUSL

```

A10 06A20 30320002 (03013) SUB(003,001) RESET TO REAL OF N-K
A10 06A22 32291302 (03014) SUBL(002,2),JUMPP(02) CONTINUE UNTIL LAST ELEMENT (N-2-2K-20)
A1A 06A24 34200031 (03016) CLEAR(01) HALT INPUT
A1B 06A26 36000020 (03018) NOP(0)
A1C 06A28 38300032 (03020) * OUTPUT PROGRAM
A1D 06A2A 3A403012 (03024) * F11AL33
A1E 06A2C 3C500000 (03025) SET(RA)
A1F 06A2E 3E600000 (03026) LOAD(000,C33)
A20 06A30 4010003A (03027) LOAD(001,M33)
A21 06A32 42310010 (03028) LOAD(002,M33)
A22 06A34 4431002A (03029) ADL(001,2)
A23 06A36 4631002A (03030) MOV(003,000)
A24 06A38 4831002A (03031) ADD(003,001)
A25 06A3A 4A31002A (03032) ADD(003,001)
A26 06A3C 4C0200A4 (03033) ADD(003,001)
A27 06A3E 4E0200A4 (03034) ADD(003,001)
A28 06A40 50000016 (03035) ADD(000,C113,TF)
A29 06A42 52000002 (03036) ADD(000,VRI,TF)
A2A 06A44 54020002 (03037) SUB(000,VRI)
A2B 06A46 5631003A (03038) SUBL(003,4)
A2C 06A48 58110032 (03039) SUBL(001,2)
A2D 06A4A 5A00000A (03040) *
A2E 06A4C 5C020002 (03041) * OUTPUT ADDRESS GENERATION LOOP
A2F 06A4E 5E0A0002 (03042) *
A30 06A50 60A00002 (03043) *
A31 06A52 62020002 (03044) *
A32 06A54 64370002 (03045) P4
A33 06A56 66120002 (03046) ADD(000,C113,TF)
A34 06A58 68200030 (03047) SUB(003,VRI,TF)
A35 06A5A 6A000020 (03048) ADD(003,VRI,TF)
A36 06A5C 6C000020 (03049) SUB(003,VRI)
A37 06A5E 6E000020 (03050) SUBL(001,2),JUMPP(P4)
A38 06A60 68200030 (03051) CLEAR(00)
A39 06A62 6A000020 (03052) NOP(0)
A3A 06A64 6C000020 (03053) NOP(0)
A3B 06A66 6E000020 (03054) NOP(0)
A3C 06A68 68200030 (03055) *
A3D 06A6A 6A000020 (03056) *
A3E 06A6C 6C000020 (03057) *
A3F 06A6E 6E000020 (03058) *
A40 06A70 68200030 (03059) * F11AL3CA-1C ASSIGN A VALUE TO CHAIN ANCHOR
A41 06A72 6A000020 (03060) *
A42 06A74 6C000020 (03061) *
A43 06A76 6E000020 (03062) *
A44 06A78 68200030 (03063) *
A45 06A7A 6A000020 (03064) *
A46 06A7C 6C000020 (03065) *
A47 06A7E 6E000020 (03066) *
A48 06A80 68200030 (03067) *
A49 06A82 6A000020 (03068) *
A50 06A84 6C000020 (03069) *
A51 06A86 6E000020 (03070) *
A52 06A88 68200030 (03071) *
A53 06A8A 6A000020 (03072) *
A54 06A8C 6C000020 (03073) *
A55 06A8E 6E000020 (03074) *
A56 06A90 68200030 (03075) *
A57 06A92 6A000020 (03076) *
A58 06A94 6C000020 (03077) *
A59 06A96 6E000020 (03078) *
A60 06A98 68200030 (03079) *
A61 06A9A 6A000020 (03080) *
A62 06A9C 6C000020 (03081) *
A63 06A9E 6E000020 (03082) *
A64 06AA0 68200030 (03083) *
A65 06AA2 6A000020 (03084) *
A66 06AA4 6C000020 (03085) *
A67 06AA6 6E000020 (03086) *
A68 06AA8 68200030 (03087) *
A69 06AAA 6A000020 (03088) *
A70 06AAC 6C000020 (03089) *
A71 06AAE 6E000020 (03090) *
A72 06AB0 68200030 (03091) *
A73 06AB2 6A000020 (03092) *
A74 06AB4 6C000020 (03093) *
A75 06AB6 6E000020 (03094) *
A76 06AB8 68200030 (03095) *
A77 06ABA 6A000020 (03096) *
A78 06ABC 6C000020 (03097) *
A79 06ABE 6E000020 (03098) *
A80 06AB0 68200030 (03099) *
A81 06AB2 6A000020 (03100) *
A82 06AB4 6C000020 (03101) *
A83 06AB6 6E000020 (03102) *
A84 06AB8 68200030 (03103) *
A85 06ABA 6A000020 (03104) *
A86 06ABC 6C000020 (03105) *
A87 06ABE 6E000020 (03106) *
A88 06AB0 68200030 (03107) *
A89 06AB2 6A000020 (03108) *
A90 06AB4 6C000020 (03109) *
A91 06AB6 6E000020 (03110) *
A92 06AB8 68200030 (03111) *
A93 06ABA 6A000020 (03112) *
A94 06ABC 6C000020 (03113) *
A95 06ABE 6E000020 (03114) *
A96 06AB0 68200030 (03115) *
A97 06AB2 6A000020 (03116) *
A98 06AB4 6C000020 (03117) *
A99 06AB6 6E000020 (03118) *
A100 06AB8 68200030 (03119) *
A101 06ABA 6A000020 (03120) *
A102 06ABC 6C000020 (03121) *
A103 06ABE 6E000020 (03122) *
A104 06AB0 68200030 (03123) *
A105 06AB2 6A000020 (03124) *
A106 06AB4 6C000020 (03125) *
A107 06AB6 6E000020 (03126) *
A108 06AB8 68200030 (03127) *
A109 06ABA 6A000020 (03128) *
A110 06ABC 6C000020 (03129) *
A111 06ABE 6E000020 (03130) *
A112 06AB0 68200030 (03131) *
A113 06AB2 6A000020 (03132) *
A114 06AB4 6C000020 (03133) *
A115 06AB6 6E000020 (03134) *
A116 06AB8 68200030 (03135) *
A117 06ABA 6A000020 (03136) *
A118 06ABC 6C000020 (03137) *
A119 06ABE 6E000020 (03138) *
A120 06AB0 68200030 (03139) *
A121 06AB2 6A000020 (03140) *
A122 06AB4 6C000020 (03141) *
A123 06AB6 6E000020 (03142) *
A124 06AB8 68200030 (03143) *
A125 06ABA 6A000020 (03144) *
A126 06ABC 6C000020 (03145) *
A127 06ABE 6E000020 (03146) *
A128 06AB0 68200030 (03147) *
A129 06AB2 6A000020 (03148) *
A130 06AB4 6C000020 (03149) *
A131 06AB6 6E000020 (03150) *
A132 06AB8 68200030 (03151) *
A133 06ABA 6A000020 (03152) *
A134 06ABC 6C000020 (03153) *
A135 06ABE 6E000020 (03154) *
A136 06AB0 68200030 (03155) *
A137 06AB2 6A000020 (03156) *
A138 06AB4 6C000020 (03157) *
A139 06AB6 6E000020 (03158) *
A140 06AB8 68200030 (03159) *
A141 06ABA 6A000020 (03160) *
A142 06ABC 6C000020 (03161) *
A143 06ABE 6E000020 (03162) *
A144 06AB0 68200030 (03163) *
A145 06AB2 6A000020 (03164) *
A146 06AB4 6C000020 (03165) *
A147 06AB6 6E000020 (03166) *
A148 06AB8 68200030 (03167) *
A149 06ABA 6A000020 (03168) *
A150 06ABC 6C000020 (03169) *
A151 06ABE 6E000020 (03170) *
A152 06AB0 68200030 (03171) *
A153 06AB2 6A000020 (03172) *
A154 06AB4 6C000020 (03173) *
A155 06AB6 6E000020 (03174) *
A156 06AB8 68200030 (03175) *
A157 06ABA 6A000020 (03176) *
A158 06ABC 6C000020 (03177) *
A159 06ABE 6E000020 (03178) *
A160 06AB0 68200030 (03179) *
A161 06AB2 6A000020 (03180) *
A162 06AB4 6C000020 (03181) *
A163 06AB6 6E000020 (03182) *
A164 06AB8 68200030 (03183) *
A165 06ABA 6A000020 (03184) *
A166 06ABC 6C000020 (03185) *
A167 06ABE 6E000020 (03186) *
A168 06AB0 68200030 (03187) *
A169 06AB2 6A000020 (03188) *
A170 06AB4 6C000020 (03189) *
A171 06AB6 6E000020 (03190) *
A172 06AB8 68200030 (03191) *
A173 06ABA 6A000020 (03192) *
A174 06ABC 6C000020 (03193) *
A175 06ABE 6E000020 (03194) *
A176 06AB0 68200030 (03195) *
A177 06AB2 6A000020 (03196) *
A178 06AB4 6C000020 (03197) *
A179 06AB6 6E000020 (03198) *
A180 06AB8 68200030 (03199) *
A181 06ABA 6A000020 (03200) *
A182 06ABC 6C000020 (03201) *
A183 06ABE 6E000020 (03202) *
A184 06AB0 68200030 (03203) *
A185 06AB2 6A000020 (03204) *
A186 06AB4 6C000020 (03205) *
A187 06AB6 6E000020 (03206) *
A188 06AB8 68200030 (03207) *
A189 06ABA 6A000020 (03208) *
A190 06ABC 6C000020 (03209) *
A191 06ABE 6E000020 (03210) *
A192 06AB0 68200030 (03211) *
A193 06AB2 6A000020 (03212) *
A194 06AB4 6C000020 (03213) *
A195 06AB6 6E000020 (03214) *
A196 06AB8 68200030 (03215) *
A197 06ABA 6A000020 (03216) *
A198 06ABC 6C000020 (03217) *
A199 06ABE 6E000020 (03218) *
A200 06AB0 68200030 (03219) *
A201 06AB2 6A000020 (03220) *
A202 06AB4 6C000020 (03221) *
A203 06AB6 6E000020 (03222) *
A204 06AB8 68200030 (03223) *
A205 06ABA 6A000020 (03224) *
A206 06ABC 6C000020 (03225) *
A207 06ABE 6E000020 (03226) *
A208 06AB0 68200030 (03227) *
A209 06AB2 6A000020 (03228) *
A210 06AB4 6C000020 (03229) *
A211 06AB6 6E000020 (03230) *
A212 06AB8 68200030 (03231) *
A213 06ABA 6A000020 (03232) *
A214 06ABC 6C000020 (03233) *
A215 06ABE 6E000020 (03234) *
A216 06AB0 68200030 (03235) *
A217 06AB2 6A000020 (03236) *
A218 06AB4 6C000020 (03237) *
A219 06AB6 6E000020 (03238) *
A220 06AB8 68200030 (03239) *
A221 06ABA 6A000020 (03240) *
A222 06ABC 6C000020 (03241) *
A223 06ABE 6E000020 (03242) *
A224 06AB0 68200030 (03243) *
A225 06AB2 6A000020 (03244) *
A226 06AB4 6C000020 (03245) *
A227 06AB6 6E000020 (03246) *
A228 06AB8 68200030 (03247) *
A229 06ABA 6A000020 (03248) *
A230 06ABC 6C000020 (03249) *
A231 06ABE 6E000020 (03250) *
A232 06AB0 68200030 (03251) *
A233 06AB2 6A000020 (03252) *
A234 06AB4 6C000020 (03253) *
A235 06AB6 6E000020 (03254) *
A236 06AB8 68200030 (03255) *
A237 06ABA 6A000020 (03256) *
A238 06ABC 6C000020 (03257) *
A239 06ABE 6E000020 (03258) *
A240 06AB0 68200030 (03259) *
A241 06AB2 6A000020 (03260) *
A242 06AB4 6C000020 (03261) *
A243 06AB6 6E000020 (03262) *
A244 06AB8 68200030 (03263) *
A245 06ABA 6A000020 (03264) *
A246 06ABC 6C000020 (03265) *
A247 06ABE 6E000020 (03266) *
A248 06AB0 68200030 (03267) *
A249 06AB2 6A000020 (03268) *
A250 06AB4 6C000020 (03269) *
A251 06AB6 6E000020 (03270) *
A252 06AB8 68200030 (03271) *
A253 06ABA 6A000020 (03272) *
A254 06ABC 6C000020 (03273) *
A255 06ABE 6E000020 (03274) *
A256 06AB0 68200030 (03275) *
A257 06AB2 6A000020 (03276) *
A258 06AB4 6C000020 (03277) *
A259 06AB6 6E000020 (03278) *
A260 06AB8 68200030 (03279) *
A261 06ABA 6A000020 (03280) *
A262 06ABC 6C000020 (03281) *
A263 06ABE 6E000020 (03282) *
A264 06AB0 68200030 (03283) *
A265 06AB2 6A000020 (03284) *
A266 06AB4 6C000020 (03285) *
A267 06AB6 6E000020 (03286) *
A268 06AB8 68200030 (03287) *
A269 06ABA 6A000020 (03288) *
A270 06ABC 6C000020 (03289) *
A271 06ABE 6E000020 (03290) *
A272 06AB0 68200030 (03291) *
A273 06AB2 6A000020 (03292) *
A274 06AB4 6C000020 (03293) *
A275 06AB6 6E000020 (03294) *
A276 06AB8 68200030 (03295) *
A277 06ABA 6A000020 (03296) *
A278 06ABC 6C000020 (03297) *
A279 06ABE 6E000020 (03298) *
A280 06AB0 68200030 (03299) *
A281 06AB2 6A000020 (03300) *
A282 06AB4 6C000020 (03301) *
A283 06AB6 6E000020 (03302) *
A284 06AB8 68200030 (03303) *
A285 06ABA 6A000020 (03304) *
A286 06ABC 6C000020 (03305) *
A287 06ABE 6E000020 (03306) *
A288 06AB0 68200030 (03307) *
A289 06AB2 6A000020 (03308) *
A290 06AB4 6C000020 (03309) *
A291 06AB6 6E000020 (03310) *
A292 06AB8 68200030 (03311) *
A293 06ABA 6A000020 (03312) *
A294 06ABC 6C000020 (03313) *
A295 06ABE 6E000020 (03314) *
A296 06AB0 68200030 (03315) *
A297 06AB2 6A000020 (03316) *
A298 06AB4 6C000020 (03317) *
A299 06AB6 6E000020 (03318) *
A300 06AB8 68200030 (03319) *
A301 06ABA 6A000020 (03320) *
A302 06ABC 6C000020 (03321) *
A303 06ABE 6E000020 (03322) *
A304 06AB0 68200030 (03323) *
A305 06AB2 6A000020 (03324) *
A306 06AB4 6C000020 (03325) *
A307 06AB6 6E000020 (03326) *
A308 06AB8 68200030 (03327) *
A309 06ABA 6A000020 (03328) *
A310 06ABC 6C000020 (03329) *
A311 06ABE 6E000020 (03330) *
A312 06AB0 68200030 (03331) *
A313 06AB2 6A000020 (03332) *
A314 06AB4 6C000020 (03333) *
A315 06AB6 6E000020 (03334) *
A316 06AB8 68200030 (03335) *
A317 06ABA 6A000020 (03336) *
A318 06ABC 6C000020 (03337) *
A319 06ABE 6E000020 (03338) *
A320 06AB0 68200030 (03339) *
A321 06AB2 6A000020 (03340) *
A322 06AB4 6C000020 (03341) *
A323 06AB6 6E000020 (03342) *
A324 06AB8 68200030 (03343) *
A325 06ABA 6A000020 (03344) *
A326 06ABC 6C000020 (03345) *
A327 06ABE 6E000020 (03346) *
A328 06AB0 68200030 (03347) *
A329 06AB2 6A000020 (03348) *
A330 06AB4 6C000020 (03349) *
A331 06AB6 6E000020 (03350) *
A332 06AB8 68200030 (03351) *
A333 06ABA 6A000020 (03352) *
A334 06ABC 6C000020 (03353) *
A335 06ABE 6E000020 (03354) *
A336 06AB0 68200030 (03355) *
A337 06AB2 6A000020 (03356) *
A338 06AB4 6C000020 (03357) *
A339 06AB6 6E000020 (03358) *
A340 06AB8 68200030 (03359) *
A341 06ABA 6A000020 (03360) *
A342 06ABC 6C000020 (03361) *
A343 06ABE 6E000020 (03362) *
A344 06AB0 68200030 (03363) *
A345 06AB2 6A000020 (03364) *
A346 06AB4 6C000020 (03365) *
A347 06AB6 6E000020 (03366) *
A348 06AB8 68200030 (03367) *
A349 06ABA 6A000020 (03368) *
A350 06ABC 6C000020 (03369) *
A351 06ABE 6E000020 (03370) *
A352 06AB0 68200030 (03371) *
A353 06AB2 6A000020 (03372) *
A354 06AB4 6C000020 (03373) *
A355 06AB6 6E000020 (03374) *
A356 06AB8 68200030 (03375) *
A357 06ABA 6A000020 (03376) *
A358 06ABC 6C000020 (03377) *
A359 06ABE 6E000020 (03378) *
A360 06AB0 68200030 (03379) *
A361 06AB2 6A000020 (03380) *
A362 06AB4 6C000020 (03381) *
A363 06AB6 6E000020 (03382) *
A364 06AB8 68200030 (03383) *
A365 06ABA 6A000020 (03384) *
A366 06ABC 6C000020 (03385) *
A367 06ABE 6E000020 (03386) *
A368 06AB0 68200030 (03387) *
A369 06AB2 6A000020 (03388) *
A370 06AB4 6C000020 (03389) *
A371 06AB6 6E000020 (03390) *
A372 06AB8 68200030 (03391) *
A373 06ABA 6A000020 (03392) *
A374 06ABC 6C000020 (03393) *
A375 06ABE 6E000020 (03394) *
A376 06AB0 68200030 (03395) *
A377 06AB2 6A000020 (03396) *
A378 06AB4 6C000020 (03397) *
A379 06AB6 6E000020 (03398) *
A380 06AB8 68200030 (03399) *
A381 06ABA 6A000020 (03400) *
A382 06ABC 6C000020 (03401) *
A383 06ABE 6E000020 (03402) *
A384 06AB0 68200030 (03403) *
A385 06AB2 6A000020 (03404) *
A386 06AB4 6C000020 (03405) *
A387 06AB6 6E000020 (03406) *
A388 06AB8 68200030 (03407) *
A389 06ABA 6A000020 (03408) *
A390 06ABC 6C000020 (03409) *
A391 06ABE 6E000020 (03410) *
A392 06AB0 68200030 (03411) *
A393 06AB2 6A000020 (03412) *
A394 06AB4 6C000020 (03413) *
A395 06AB6 6E000020 (03414) *
A396 06AB8 68200030 (03415) *
A397 06ABA 6A000020 (03416) *
A398 06ABC 6C000020 (03417) *
A399 06ABE 6E000020 (03418) *
A400 06AB0 68200030 (03419) *
A401 06AB2 6A000020 (03420) *
A402 06AB4 6C000020 (03421) *
A403 06AB6 6E000020 (03422) *
A404 06AB8 68200030 (03423) *
A405 06ABA 6A000020 (03424) *
A406 06ABC 6C000020 (03425) *
A407 06ABE 6E000020 (03426) *
A408 06AB0 68200030 (03427) *
A409 06AB2 6A000020 (03428) *
A410 06AB4 6C000020 (03429) *
A411 06AB6 6E000020 (03430) *
A412 06AB8 68200030 (03431) *
A413 06ABA 6A000020 (03432) *
A414 06ABC 6C000020 (03433) *
A415 06ABE 6E000020 (03434) *
A416 06AB0 68200030 (03435) *
A417 06AB2 6A000020 (03436) *
A418 06AB4 6C000020 (03437) *
A419 06AB6 6E000020 (03438) *
A420 06AB8 68200030 (03439) *
A421 06ABA 6A000020 (03440) *
A422 06ABC 6C000020 (03441) *
A423 06ABE 6E000020 (03442) *
A424 06AB0 68200030 (03443) *
A425 06AB2 6A000020 (03444) *
A426 06AB4 6C000020 (03445) *
A427 06AB6 6E000020 (03446) *
A428 06AB8 68200030 (03447) *
A429 06ABA 6A000020 (03448) *
A430 06ABC 6C000020 (03449) *
A431 06ABE 6E000020 (03450) *
A432 06AB0 68200030 (03451) *
A433 06AB2 6A000020 (03452) *
A434 06AB4 6C000020 (03453) *
A435 06AB6 6E000020 (03454) *
A436 06AB8 68200030 (03455) *
A437 06ABA 6A000020 (03456) *
A438 06ABC 6C000020 (03457) *
A439 06ABE 6E000020 (03458) *
A440 06AB0 68200030 (03459) *
A441 06AB2 6A000020 (03460) *
A442 06AB4 6C000020 (03461) *
A443 06AB6 6E000020 (03462) *
A444 06AB8 68200030 (03463) *
A445 06ABA 6A000020 (03464) *
A446 06ABC 6C000020 (03465) *
A447 06ABE 6E000020 (03466) *
A448 06AB0 682000
```

PAGE 88: (CMDJ)<DC116>BNH16M-MS0.2, 29-Dec-88 14130140, Edt WOLF
APS3 - FILAL APS PROGRAM FOR PROSL

06A60 0000000 (03066) FILALSCI DATA 0F'0.0"

...

(03067) *
(03068) *
00000000 (03069) FILALSSZ= 0L-FILAL\$
(03070) *
(03071) *
(03072) *
(03073) *

COMPUTE MODULE SIZE

EVEN

```

(03074) - APU3 - APC(V,A,U,V,M,R,S,T) APC (WITH NOISE SHAPING)
(03075) - FIELD 0/00
(03076) -
(03077) - RUNS WITH APU3-APC (APCS)
(03078) -
(03079) - FUNCTION:
(03080) - COMPUTE CODED APC RESIDUAL FROM PREEMPHASIZED
(03081) - INPUT SPEECH. (SEE DETAILED DOCUMENTATION OF
(03082) - APC LOOP IN ALGORITHM SPECIFICATION.)
(03083) -
(03084) - PARAMETERS:
(03085) - OUT: V(FIXED): CODED RESIDUAL SAMPLES
(03086) - IN: A(REAL): PITCH LAG (SA)
(03087) - IN: U(REAL): PREEMPHASIZED SPEECH
(03088) - IN: V(REAL): FILTER COEFFS: 6 SPECTRAL, 3 PITCH, 6 M.S.
(03089) - IN: W(REAL): 3 QUANTIZER SEGMENT SCALE FACTORS,
(03090) - ALTERNATING WITH THEIR RECIPROCAL,
(03091) - IN/OUT: R(REAL): PITCH FILTER HISTORY: LAST, CURRENT FRAMES
(03092) - IN/OUT: S(REAL): SPECTRAL FILTER HISTORY
(03093) - IN/OUT: T(REAL): NOISE SHAPING FILTER HISTORY
(03094) -
(03095) - SYMBOLS USED IN COMMENTS IN THIS MODULE:
(03096) - I: INDEX OF CURRENT ITERATION OF APC LOOP (01005-1)
(03097) - M: PITCH LAG (IN SAMPLES) (SA)
(03098) - ANS(1:16) NOISE SHAPING FILTER COEFFS (ZEROS) (V(9:14))
(03099) - AN(1:16) SPECTRAL FILTER (ALSO M.S. FILTER POLES) COEFFS (V(0:15))
(03100) - C1H PITCH FILTER COEFF #1 (V(6))
(03101) - C2H PITCH FILTER COEFF #2 (V(7))
(03102) - C3H PITCH FILTER COEFF #3 (V(8))
(03103) - Q1(1-6:I-1) NOISE SHAPING FILTER INPUT (T(0:15))
(03104) - V(1-6:I-1) SPECTRAL FILTER INPUT (S(0:15))
(03105) - RH(1-M+1) PITCH FILTER INPUT #1 (MOST RECENT) (RH(0)=R(FRMSZ)...
(03106) - RH(1-M) ...WHERE FRMSZ FIXED AT 216)
(03107) - RH(1-M-1) PITCH FILTER INPUT #2
(03108) - QP(1) M.S. FILTER (ZEROS) OUTPUT: SUMCANS(J)*Q1(1-J), J=1:6
(03109) - QP(1) M.S. FILTER (POLES) OUTPUT: SUMC AN(J)*Q1(1-J), J=1:6
(03110) - VHP(1) SPECTRAL FILTER OUTPUT: SUMC AN(J)*VH(1-J), J=1:6
(03111) - RHP(1) PITCH FILTER OUTPUT: C1H*RH(1-M+1) +
(03112) - C2H*RH(1-M) +
(03113) - C3H*RH(1-M-1)
(03114) - SP(1) SPEECH INPUT SAMPLE FOR CURRENT APC LOOP ITERATION (U(1))
(03115) - WO(1) UNQUANTIZED APC RESIDUAL: SP(1)*QP(1)-QP(1)*VHP(1)+RHP(1)
(03116) - CFAC(1:13) QUANTIZER SCALE FACTORS FOR FRAME SEGMENTS 1:3 (V(0,2,4))
(03117) - QD(1) NORMALIZED APC RESIDUAL: WO/CFAC
(03118) - LU(1) CODED APC RESIDUAL SAMPLE (V(1))
(03119) - UH(1) QUANTIZED APC RESIDUAL SAMPLE
(03120) - VU(1) SCALED QUANTIZED APC RESIDUAL SAMPLE: UH*CFAC
(03121) - Q(1) CURRENT SAMPLE QUANTIZATION NOISE: WH(1)-WO(1)
(03122) -
(03123) - THIS APU PROGRAM (IN CONJUNCTION WITH ITS APS PROGRAM)
(03124) - MODIFIES ONE INSTRUCTION IN ITS APS PROGRAM BY:
(03125) - 0. READING PITCH FROM SCALAR A, AND FIXING IT
(03126) - 1. WRITING IT INTO THE RH UP THE BUS-1 COPY OF THE INSTRUCTION

```

```

(03927) ) 2. WAITING ON ONE TO BE SURE IT'S THERE (PLOS 1 NOP THAT STORER
(03928) ) 3AYS IS NECESSARY)
(03929) ) 3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
(03930) ) 4. WRITING IT TO APSMEM IN PSEUDOMEMORY
(03931) ) 5. WAITING FOR ONE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
(03932) ) PROCEED.
(03933) )
(03934) )
(03935) )
(03936) )
(03937) )
(03938) )
(03939) )
(03940) )
(03941) )
(03942) )
(03943) )
(03944) )
(03945) )
(03946) )
(03947) )
(03948) )
(03949) )
(03950) )
(03951) )
(03952) )
(03953) )
(03954) )
(03955) )
(03956) )
(03957) )
(03958) )
(03959) )
(03960) )
(03961) )
(03962) )
(03963) )
(03964) )
(03965) )
(03966) )
(03967) )
(03968) )
(03969) )
(03970) )
(03971) )
(03972) )
(03973) )
(03974) )
(03975) )
(03976) )
(03977) )
(03978) )
(03979) )

```

GENERAL SCHEME1

- 1) FIX PITCH, SEND TO APS PROGRAM
- 2) INITIALIZE REGISTERS FOR LOOP: GET VH'S, RH'S
- 3) APC LOOP:
 - COMPUTE FILTER OUTPUTS
 - COMPUTE RESIDUAL: $WO = SP(I) + QP(I) - QP(I) + RHP(I) + VHP(I)$
 - NORMALIZE RESIDUAL: $VO = WO/CFAC$
 - CODE/QUANTIZE NORMALIZED RESIDUAL
 - SCALE QUANTIZED RESIDUAL: $WH = WH + GPAC$
 - UPDATE FILTER INPUTS: $Q(I) = WH(I) - VO(I)$
 - $Q(I) = Q(I) - QP(I)$
 - $VH(I) = VH(I) - VHP(I)$
 - $RH(I) = RH(I) - RHP(I)$

DO NEXT APC LOOP ITERATION: $I = I + 1$

NOTE: SPECTRAL FILTER INPUTS AND PITCH FILTER INPUTS (VH'S & RH'S) ARE RETAINED FROM ITERATION TO ITERATION IN APS REGISTERS. W.S. FILTER INPUTS, AND ALL FILTER COEFFS, ARE RECALC FOR EACH ITERATION OF THE APC LOOP. W.S. FILTER INPUTS ARE ALSO UPDATED AND WRITTEN OUT AFTER EACH ITERATION'S USE.

REGISTER USAGE (FOR MAIN APC LOOP):

ADM1	ADM2
----	----
A0: RHP	A0: WO SUB/WD/Q
A1: RH(I-M)	A1: QP1
A2: RH(I-M-1)	A2: VHP
A3: SCRATCH	A3: SCRATCH
M0: ANS(1)/	M0: ANS(2)/
AH(1)/	AH(2)
RH(I-M-1)	
C3H	
M1: ANS(3)/	M1: ANS(4)/
AH(3)	AH(4)
M2: ANS(5)/	M2: ANS(6)/
AH(5)	AH(6)
M3: RH(I-M)	M3: WO/
	WH/
	RH(I-M+1)
	M4: Q1(I-2)/
	VH(I-2)
	M5: Q1(I-4)/
	VH(I-4)
	M6: Q1(I-6)/
	VH(I-6)
	M7: (1/GPAC)/
	CFAC/
	C1H

!!! ALL BUFFERS MUST BE RE-LNG,CONTIGUOUS,
!!! EXCEPT BUFFER Y WHICH MUST BE PRD,LNG,CNTG.


```

(04086) )
(04087) )
(04088) )
(04089) )
(04090) )
(04091) )
(04092) )
(04093) )
(04094) )
(04095) )
(04096) )
(04097) )
(04098) )
(04099) )
(04100) )
(04101) )
(04102) )
(04103) )
(04104) )
(04105) )
(04106) )
(04107) )
(04108) )
(04109) )
(04110) )
(04111) )
(04112) )
(04113) )
(04114) )
(04115) )
(04116) )
(04117) )
(04118) )
(04119) )
(04120) )
(04121) )
(04122) )
(04123) )
(04124) )
(04125) )
(04126) )
(04127) )
(04128) )
(04129) )
(04130) )
(04131) )
(04132) )
(04133) )
(04134) )
(04135) )
(04136) )
(04137) )
(04138) )

A16 06A9E 00000EE
A17 06A9B 00000EA
A18 06A9C 00000E9
A19 06A9D 00000E8
A1A 06A9E 05400540
A1B 06A9B 00000E0
A1C 06A9A 00000E9
A1D 06A9C 00000E0
A1E 06A9E 00000E0
A1F 06A9D 04030403

A20 06AB2 00DC0EC
A21 06AB4 00000E0
A22 06AB6 00EC0DC
A23 06AB8 00E0000
A24 06ABA 04170417

A25 06ABC 47604760

A26 06ABE 00000E4
A27 06ACB 00EA0000

A28 06AC2 85570557

A29 06AC4 47734773

      NOP \ MOV(IQA,M6)
      NOP \ MOV(IQA,M2)
      MOV(IQA,M6) \ MOV(IQ,00)
      MOV(IQA,M2) \ NOP
      MUL(M2,M6)

      MOV(IQ,00) \ MOV(IQA,M5)
      NOP \ MOV(IQA,M1)
      MOV(IQA,M5) \ MOV(IQ,00)
      MOV(IQA,M1) \ NOP
      MOV(I3),MUL(M1,M5)

      MOV(IQ,00) \ MOV(IQA,M4)
      NOP \ MOV(IQA,M3)
      MOV(IQA,M4) \ MOV(IQ,00)
      MOV(IQA,M3) \ NOP
      MOV(I7),MUL(M3,M4)

      ADD PRODUCTS SO FAR...

      ADD(I3,I7)

      START MOVING IN COEFFS FOR M.S. POLES & SPECTRAL FILTER
      AN(6),...,AN(1)

      NOP \ MOV(IQA,M2)
      MOV(IQA,M2) \ NOP

      MOVE LAST M.S. ZEROS PRODUCT TO ADDER
      START MULT FOR M.S. POLES

      MOV(I7),MUL(M2,M6)

      ADD LAST M.S. ZEROS PRODUCT TO ACCUM
      MOV(I3),ADD(I3,I7)

      --- \ M6<=Q1(I-6)
      --- \ M2<=ANS(6)
      M6<=Q1(I-5) \ Q0<=Q1(I-5)
      M2<=ANS(5) \ ---
      PC=ANS(5)Q1(I-5) \ PC=ANS(6)Q1(I-6)

      Q0<=Q1(I-4) \ M5<=Q1(I-4)
      --- \ M1<=ANS(4)
      M5<=Q1(I-3) \ Q0<=Q1(I-3)
      M1<=ANS(3) \ ---
      A3<=FIRST M.S. ZEROS PRODUCTS
      PC=ANS(3)Q1(I-3) \ PC=ANS(4)Q1(I-4)

      Q0<=Q1(I-2) \ M4<=Q1(I-2)
      --- \ M0<=ANS(2)
      M4<=Q1(I-1) \ Q0<=Q1(I-1)
      M0<=ANS(1) \ ---
      A7<=MIDDLE M.S. ZEROS PRODUCTS
      PC=ANS(1)Q1(I-1) \ PC=ANS(2)Q1(I-2)

      R<=SUM(1ST 2 SETS OF M.S.ZEROS PRODS)

      --- \ M2<=AN(6)
      M2<=AN(5) \ ---

      A7<=LAST M.S. ZEROS PRODUCT
      PC=AN(5)Q1(I-5) \ PC=AN(6)Q1(I-6)

      A3<=SUM(1ST 2 SETS OF M.S.ZEROS PRODS)
      R<=SUM(ALL 3 SETS OF M.S.ZEROS PRODS)

```


PAGE 94: C0000D1C0A16>00W16M.N50.2, 29-Dec-88 14:30:46, Ed: WOLF
AP03 - APC(V,A,U,V,W,R,S,T) APC (WITH NOISE SHAPING)

[illegible]


```

PAGE 90: (BND)DCALCDBN16M.MS0.2, 29-Dec-88 14:38:48, Ed: WOLF
APU3 - APC(T,A,U,V,M,R,S,T) APC (WITH NOISE SHAPING)

--- \ A7<=PARTIAL VMP
--- \ RC=VMP(I)
--- \ A3<=VMP(I))RC=SP+QP-QP1+VMP

A5C 06024 00000057 (04245)
A5D 0602C 00004760 (04246)
A5E 0602E 00004212 (04247)
      MOV \ MOV(EXT,A7)
      MOV \ ADD(A3,A7)
      MOV \ ADD(A2),ADD(A0,A2)
      CONTINUE WITH PITCH FILTER..

A5F 06030 04130000 (04251)
      MOV(A3),MUL(M0,M4) \ MOV(P,EIO)
      P<=RN(I-M-1)C3M \ ---
      A7<=RN(I-M-1)C3M \ ---
      RC=SUN(IST 2 PITCH PRODS) \
      AB<=SP(I)+QP(I)-QP1(I)+VMP(I)

A60 06032 00570000 (04253)
A61 06034 47600000 (04254)
      MOV(EXT,A7) \ NOP
      ADD(A3,A7) \ MOV(R,A0)

A62 06036 00000057 (04259)
A63 06038 00000000 (04260)
A64 0603A 00000000 (04261)
      START SET-UP FOR NO NORMALIZATION
      NOP \ MOV(IA,M7)
      NOP
      NOP

A65 0603C 00070000 (04263)
A66 0603E 47730000 (04264)
A67 06040 00000000 (04265)
A68 06042 00000000 (04266)
A69 06044 00000000 (04267)
A6A 06046 00000057 (04268)
A6B 06048 00004760 (04269)
A6C 0604A 00000000 (04270)
A6D 0604C 00000000 (04271)
A6E 0604E 00000000 (04272)
A6F 06050 00000000 (04273)
      MOV(P,A7) \ NOP
      MOV(A3),ADD(A3,A7) \ NOP
      NOP
      NOP
      MOV(R,EIO) \ NOP
      MOV(R,A0) \ MOV(EXT,A7)
      NOP \ ADD(A0,A7)
      NOP
      NOP \ MOV(R,A0)
      NOP \ MOV(R,M3)

A70 06052 00000050 (04274)
      NORMALIZE APC RESIDUAL:
      -----
      CALCULATE UD (NORMALIZED RESIDUAL)
      BY MULTIPLYING WD BY (1/GFAC)
      NOP \ MUL(M3,M7)
      --- \ PC=WD(1/GFAC)

CODE-DECODE NORMALIZED RESIDUAL:
-----
CODE AND QUANTIZE UD, GENERATING IU (CODED) & UH (QUANTIZED)

CODING TABLE (FOLDED BINARY CODE): (DON'T WORRY ABOUT BOUNDARY CONDITIONS)
      IU
      UD RANGE
      1 UD < -THRESH 1 VAL(1)
      2 -THRESH < UD < 0 0 VAL(2)
      3 0 < UD < THRESH 2 VAL(3)

```


PAGE 98: CERN03C0CA16>R0W16M.M50.2, 29-Dec-88 14:38:48, Ed: WOLV
APU) - APC(Y,A,V,W,R,S,T) APC (WITH NOISE SHAPING)

```

(04351) )
(04352) ) HERE FOR BIT 2
(04353) )
(04354) ) APC$B2:
APC 0608A 001C0000 MOV(ZERO,0Q) \ MOV(IQA, NULL)
APC 0608C 00000000 NOP \ MOV(IQA, M3)
APC 0608E 00000000 NOP \ MOV(IQA, NULL)
APC 06090 00000000 NOP \ MOV(IQA, NULL)
APC 06092 10000005 JUMP(APC$COM)

(04360) )
(04361) ) HERE FOR BIT 1
(04362) )
(04363) ) APC$B1:
APC 06094 009C0000 MOV(R,0Q) \ MOV(IQA, M3)
APC 06096 00000000 NOP \ MOV(IQA, NULL)
APC 06098 00000000 NOP \ MOV(IQA, NULL)
APC 0609A 00000000 NOP \ MOV(IQA, NULL)

(04368) )
(04369) )
(04370) ) HERE WHEN DONE WITH CODING AND QUANTIZATION
(04371) ) OH IS IN M3.
(04372) )
(04373) ) APC$COM:
(04374) )
(04375) )
(04376) )
(04377) ) SCALE THE QUANTIZED RESIDUAL:
(04378) )
(04379) )
(04380) )
(04381) )
(04382) ) UPDATE ARRAYS:
(04383) )
(04384) )
APC 0609E 00000007 NOP \ MOV(P, A7)
APC 060A0 00004000 NOP \ SUB(A7, A0)
APC 060A2 00004910 NOP \ MOV(A0), SUB(A0, A1)
APC 060A4 00004AFC NOP \ MOV(0Q), SUM(A7, A2)
APC 060A6 00000000 NOP \ MOV(R, EXD)
APC 060A8 00540000 MOV(EXT, A4) \ NOP
APC 060AA 40000000 SUB(A4, A0) \ NOP
APC 060AC 009C0000 MOV(R, 0Q) \ NOP

(04394) )
(04395) )
(04396) ) DONE WITH APC LOOP.
(04397) ) LOOP TIL (PI)
(04398) )
APC 060AE 90100015 JUMPC(APC$TOP, FI)
APC 060B0 00000000 NOP
(04401) )
(04402) )
(04403) ) WHEN DONE, WRITE OUT VH'S

```


(88MB3) <DCAL16> 88MB16M.W50.2, 29-Dec-88 14:30:40, Ed: WOLF
 APS3 - APC(V,A,U,V,W,R,S,T) APC (WITH NOISE SHAPING)

```
(84421) - APS3 - APC(V,A,U,V,W,R,S,T) APC (WITH NOISE SHAPING)
(84422) ) KFIELD 9/88
(84423) )
(84424) ) BINDING DONE BY SBH$APC TO ACCOMMODATE R, S & T BUFFERS
(84425) )
(84426) ) !!! ALL BUFFERS MUST BE BL,LNG,CONTIGUOUS,
(84427) ) !!! EXCEPT BUFFER Y WHICH MUST BE FID,LNG,CNTG.
(84428) )
(84429) ) THIS APS PROGRAM (IN CONJUNCTION WITH ITS APS PROGRAM)
(84430) ) MODIFIES ONE INSTRUCTION IN ITSELF BY:
(84431) ) 0. READING PITCH FROM SCALAR A, AND FIXING IT
(84432) ) 1. WRITING IT INTO THE RH OF THE BUS-1 COPY OF THE INSTRUCTION
(84433) ) 2. WAITING ON ONE TO BE SURE IT'S THERE (PLUS 1 NOP THAT STORER
(84434) ) SAYS IS NECESSARY)
(84435) ) 3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
(84436) ) 4. WRITING IT TO APSNEM IN PSEUDOMEMORY
(84437) ) 5. WAITING FOR ONE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
(84438) ) PROCEED.
(84439) )
(84440) ) INPUT SEQUENCE:
(84441) ) SYMBOLS
(84442) ) -----
(84443) ) (2**15), PITCH,
(84444) ) (MOD APS INSTR),
(84445) ) RH(-M), RH(-M-1),
(84446) ) VH(-6),...,VH(-1),
(84447) ) LOOP PRMSZ (-216) TIMES:
(84448) ) 1
(84449) ) 0
(84450) ) 1
(84451) ) 1
(84452) ) 1
(84453) ) 1
(84454) ) 1
(84455) ) 1
(84456) ) 1
(84457) ) 1
(84458) ) 1
(84459) ) 1
(84460) ) 1
(84461) ) 1
(84462) ) 1
(84463) ) 1
(84464) ) 1
(84465) ) 1
(84466) ) 1
(84467) ) END OF LOOP
(84468) )
(84469) )
(84470) )
(84471) ) OUTPUT SEQUENCE:
(84472) ) SYMBOLS
(84473) ) -----
```

```
-----
(2**15), SA,
(MOD APS INSTR),
R(PRMSZ-M), R(PRMSZ-M-1),
S(0),...,S(5),
U(1),
V(0),V(14),
...
T(5),V(9),
V(5),...,V(8),
V(6),V(7),V(8),
R(PRMSZ-I-M+1),
W(1) FOR ITERS 0:71
W(3) FOR ITERS 72:143
W(5) FOR ITERS 144:215,
(CODING THRESH),
(2**15),
W(0) FOR ITERS 0:71
W(2) FOR ITERS 72:143
W(4) FOR ITERS 144:215,
(QUANT VAL(1)),
(QUANT VAL(2)),
(QUANT VAL(3)),
(QUANT VAL(4)),
LFI)
-----
```

```
-----
(2**15),
(CODING THRESH),
(2**15),
GFAC,
(QUANT VAL(1)),
(QUANT VAL(2)),
(QUANT VAL(3)),
(QUANT VAL(4)),
END OF LOOP
LFI)
-----
```

```

PAGE 1011 100ND130CAL1>0001GH.N50.2, 29-Dec-88 14130140, Ed: VOLV
          APS3 - APC(V,A,W,V,W,S,T) APC (WITH NOISE SHAPING)

(04474) ) (FID PITCH), (MOD APS INSTR), (FID PITCH), (MOD APS INSTR),
(04475) ) FID FMSZ (-216) TIMES!
(04476) ) 1 Q1(I-6),...,Q1(I-2), T(0),...,T(4),
(04477) ) 1 T(1), V(1),
(04478) ) 1 Q1(I-1), T(5),
(04479) ) 1 RH(1), R(FMSZ+1),
(04480) ) FID OF LOOP
(04481) ) WH(I-6),...,WH(I-1), S(0),...,S(5),
(04482) ) 1 Z(0)
(04483) )
(04484) )
(04485) ) DEFINE SYMBOLIC FRAMESIZE
(04486) )
          FMSZ = 216
00000000 (04487) )
          ZVEN
          ADDR 0 )PTR TO COMS INS BLOCK)
          ADDR 0 )SCALAR ADDRESS)
          DATA 0 )NUMBER OF SCALARS)
          DATA APCSZ )MODULE SIZE
          ADDR APCSSC )PTR TO CHAIN ANCHOR
          APC3
(04496) ) BEGIN APS(APCS)
(04497) ) JSM(APCS$0,P2) )SET UP AND START OUTPUT PROGRAM
          SET(RO)
(04498) )
(04499) ) INPUT PROGRAM
          BR0: SCRATCH
          BR1: PITCH / R(FMSZ+1-M+1)
          BR2: W(1)
          BR3: INPUT LOOP ITERATION COUNTER ((FMSZ-1)/0)
          APC$1:
(04506) ) APC$1:
(04507) )
(04508) )
(04509) ) DO INSTRUCTION MODIFICATION STUFF:
(04510) )
          LOAD(BR0,SVTSUM(1),TF) GEN (2**-15) ADDR
          APC$1:
          LOAD(BR0,M$(1),TF) GEN SA ADDR
          SET(W1) WAIT FOR APS TO SIGNAL
          NOP WRITE COMPLETE
          LOAD(BR0,APCS$MOD(1),TF) GEN MOD INSTR ADDR
          SET(W1) WAIT FOR WRITE COMPLETE
          NOP
          APC$1:
(04512) )
          APC$1:
(04513) )
          APC$1:
(04514) )
          APC$1:
(04515) )
          APC$1:
(04516) )
          APC$1:
(04517) )
          APC$1:
(04518) )
          APC$1:
(04519) )
          APC$1:
(04520) )
          APC$1:
(04521) )
          APC$1:
(04522) )
          APC$1:
(04523) ) GEN R(FMSZ-M), R(FMSZ-M-1)
          APC$1:
(04524) ) ALSO GET INITIAL R(FMSZ+1-M+1) (MINUS SPACING)
          APC$1:
(04525) )
          APC$1:
(04526) ) APC$1:

```


PAGE 103: [DWD]<DCA16>DWDH16.M50.2, 29-Dec-88 14:38:40, Ed: WOLF
 IP33 - APC(V,A,U,V,N,R,S,T) APC (WITH NOISE SHAPING)

```

(04500) APC$IT3:
A20 06C10 40C00000 LOAD(0R0,M$$,TF) GEN ADDR T(3)
(04501) APC$V11:
A21 06C12 42C00000 LOAD(0R0,M$$,TF) GEN ADDR V(11)
(04503) APC$IT4:
A22 06C14 44C00000 LOAD(0R0,M$$,TF) GEN ADDR T(4)
(04505) APC$V10:
A23 06C16 46C00000 LOAD(0R0,M$$,TF) GEN ADDR V(10)
(04507) APC$IT5:
A24 06C18 48C00000 LOAD(0R0,M$$,TF) GEN ADDR T(5)
(04509) APC$V9:
A25 06C1A 41C00000 LOAD(0R0,M$$,TF) GEN ADDR V(9)
(04593) ) GEN V(5),...,V(8)
(04594) )
(04595) APC$V5:
A26 06C1C 4CC00000 LOAD(0R0,M$$,TF) GEN ADDR V(5)
A27 06C1E 4E200002 SUB(0R0,V$,TF) GEN ADDR V(4)
A28 06C20 50200002 SUB(0R0,V$,TF) GEN ADDR V(3)
A29 06C22 52020002 SUB(0R0,V$,TF) GEN ADDR V(2)
A2A 06C24 54020002 SUB(0R0,V$,TF) GEN ADDR V(1)
A2B 06C26 56020002 SUB(0R0,V$,TF) GEN ADDR V(0)
(04602) ) GEN V(6),V(7),V(8)
(04603) )
(04604) )
(04605) APC$V6:
A2C 06C28 58C00000 LOAD(0R0,M$$,TF) GEN ADDR V(6)
A2D 06C2A 5A0A0002 ADD(0R0,V$,TF) GEN ADDR V(7)
A2E 06C2C 5C0A0002 ADD(0R0,V$,TF) GEN ADDR V(8)
(04609) ) GEN R(FRMSZ+I-N+1)
(04610) )
(04611) )
A2F 06C2E 5E9A0002 ADD(PA1,V$,TF) GEN ADDR R(FRMSZ+I-N+1)
(04612) )
(04613) )
(04614) ) FOR I=0:71 GEN: W(1),CODING THRESH,(2*-15),W(0)
(04615) ) I=72:143 : W(3),CODING THRESH,(2*-15),W(2)
(04616) ) I=144:215 : W(5),CODING THRESH,(2*-15),W(4)
(04617) )
(04618) ) I=0:71 => ITERATION COUNTER = BR3 = (FRMSZ-1):(FRMSZ-72) = (215:144)
(04619) ) I=72:143 => ITERATION COUNTER = BR3 = (FRMSZ-73):(FRMSZ-144) = (143:72)
(04620) ) I=144:215 => ITERATION COUNTER = BR3 = (FRMSZ-145):(FRMSZ-216) = (71:0)
(04621) ) SEE IF BR3 IS IN (215:144) => W(1)/W(0)
(04622) )
(04623) )
A30 06C30 60400007 LOAD(0R0,143)
A31 06C32 62037A7 SUBD(0R0,BR3),JUMPP(APCS$2) BR3 <= 143 - (ITER CTR)
(04625) ) JUMP IF BR3 NOT IN (215:144)
(04626) )
(04627) APC$W1:
A32 06C34 64C00000 LOAD(0R0,M$$,TF) GEN W(1)
A33 06C36 66C399FC LOAD(0R0,CTN0+W$,V$(1),TF) GEN POSITIVE VALUED CODING THRESH
A34 06C38 68C703CE LOAD(0R0,SVT$W1(1),TF) GEN (2*-15)
(04631) APC$W0:
A35 06C3A 6AC00000 LOAD(0R0,M$$,TF) GEN V(0)
  
```


PAGE 1961 (00001) <DC116>00016M-MS0.2, 29-Dec-88 14:38:49, E4: WOLF
 APS3 - APC(V,A,V,V,V,B,S,T) APC (WITH NOISE SHAPING)

```

(04732) ; GEN T(5)
(04733) ;
(04734) ;
(04735) ; ADD(000,VS,TF)
(04736) ;
(04737) ; GEN R(FMSZ+1)
(04738) ;
(04739) ; ADD(001,VS,TF)
(04740) ;
(04741) ; END OF LOOP
(04742) ; DECREMENT ITERATION COUNTER (003)
(04743) ; IF RESULT IS NEGATIVE, DONE.
(04744) ;
(04745) ; SUBL(003,1),JUMPF(02)
(04746) ;
(04747) ; AFTER LOOP, CLEAN UP:
(04748) ; GEN S(0),...,S(5)
(04749) ;
(04750) ; APC$031
(04751) ; LOAD(000,VS,TF)
(04752) ; ADD(000,VS,TF)
(04753) ; ADD(000,VS,TF)
(04754) ; ADD(000,VS,TF)
(04755) ; ADD(000,VS,TF)
(04756) ; ADD(000,VS,TF)
(04757) ;
(04758) ; DONE.
(04759) ;
(04760) ; CLEAR(R0)
(04761) ; NOP(0)
(04762) ; APC$C=PC
(04763) ; END
(04764) ;
(00000000) APC$Z=0L-APCS$
(04765) ;
  
```

GEN ADDR T(5)

GEN ADDR R(FMSZ+1)

GEN ADDR S(0)
 GEN ADDR S(1)
 GEN ADDR S(2)
 GEN ADDR S(3)
 GEN ADDR S(4)
 GEN ADDR S(5)

DONE WITH OUTPUT

(04766) • SONAPC SPECIAL BINDING MODULE FOR APC

(04819)	EVEN	R4,BCT\$BA(R6)	JBIT LEN., 2BIT BUS#, 17BIT BA.
(04820)	MOVNRL	R7,APCS\$+W\$-APCS\$V	JSET LEFT WORD OF INSTR
(04821)	MOVNR	R4,R7,\$FFF0	J'OR' IN HIGH 4 BITS
(04822)	TORKR	R4,R7,\$FFF0	JASSUME FID-LMG1 CLR WL BIT
(04823)	CRR	BITS\$B.R4	JMOVE PHORD TO BUS1
(04824)	MOVNRL	R4,APCS\$+W\$-APCS\$V	
(04825)	J		
(04826)	J		
(04827)	J		
(04828)	MOVNR	R6,2*W\$(R1)	JGET BID
(04829)	MOVER	R6,R6,WK\$LEFT	JCONVERT TO OCT INDEX
(04830)	LRS	R6,7	
(04831)	EVEN		
(04832)	MOVNRL	R4,BCT\$BA(R6)	JBIT LEN., 2BIT BUS#, 17BIT BA.
(04833)	MOVNR	R7,APCS\$+W\$-APCS\$V	JSET LEFT WORD OF INSTR
(04834)	TORKR	R4,R7,\$FFF0	J'OR' IN HIGH 4 BITS
(04835)	MOVNRL	R4,APCS\$+W\$-APCS\$V	JMOVE PHORD TO BUS1
(04836)	J		
(04837)	J		
(04838)	J		
(04839)	MOVNR	R6,2*W\$(R1)	JGET BID
(04840)	MOVER	R6,R6,WK\$RAVT	JCONVERT TO OCT INDEX
(04841)	LLS	R6,1	
(04842)	EVEN		
(04843)	J	V(5)... (TEMPORARILY PUT V(6) ADDR WHERE V(5) WILL GO)	
(04844)	MOVNRL	R4,BCT\$BA(R6)	JBIT LEN., 2BIT BUS#, 17BIT BA.
(04845)	MOVNR	R7,APCS\$+W\$-APCS\$V	JSET LEFT WORD OF INSTR
(04846)	TORKR	R4,R7,\$FFF0	J'OR' IN HIGH 4 BITS
(04847)	MOVNRL	R4,APCS\$+W\$-APCS\$V	JMOVE PHORD TO BUS1
(04848)	LAF	R7,APCS\$+W\$-APCS\$V	JSET ADDR OF BUFFER START
(04849)	ADDIR	R7,5*W\$	JADD HW OFFSET TO DESIRED SAMPLE
(04850)	SIF	R7,APCS\$+W\$-APCS\$V	JSTORE ADDR OF DESIRED SAMPLE
(04851)	J		
(04852)	J		
(04853)	MOVNRL	R4,BCT\$BA(R6)	JBIT LEN., 2BIT BUS#, 17BIT BA.
(04854)	MOVNR	R7,APCS\$+W\$-APCS\$V	JSET LEFT WORD OF INSTR
(04855)	TORKR	R4,R7,\$FFF0	J'OR' IN HIGH 4 BITS
(04856)	MOVNRL	R4,APCS\$+W\$-APCS\$V	JMOVE PHORD TO BUS1
(04857)	LAF	R7,APCS\$+W\$-APCS\$V	JSET ADDR OF BUFFER START
(04858)	ADDIR	R7,6*W\$	JADD HW OFFSET TO DESIRED SAMPLE
(04859)	SIF	R7,APCS\$+W\$-APCS\$V	JSTORE ADDR OF DESIRED SAMPLE
(04860)	J		
(04861)	J		
(04862)	MOVNRL	R4,BCT\$BA(R6)	JBIT LEN., 2BIT BUS#, 17BIT BA.
(04863)	MOVNR	R7,APCS\$+W\$-APCS\$V	JSET LEFT WORD OF INSTR
(04864)	TORKR	R4,R7,\$FFF0	J'OR' IN HIGH 4 BITS
(04865)	MOVNRL	R4,APCS\$+W\$-APCS\$V	JMOVE PHORD TO BUS1
(04866)	LAF	R7,APCS\$+W\$-APCS\$V	JSET ADDR OF BUFFER START
(04867)	ADDIR	R7,9*W\$	JADD HW OFFSET TO DESIRED SAMPLE
(04868)	SIF	R7,APCS\$+W\$-APCS\$V	JSTORE ADDR OF DESIRED SAMPLE
(04869)	J		
(04870)	J		
(04871)	J		
(04872)	MOVNRL	R4,BCT\$BA(R6)	JBIT LEN., 2BIT BUS#, 17BIT BA.

0600E 0706C16 (04072)	MOVNR	R7,APC\$+W\$*APC\$V10	GET LEFT WORD OF INSTR
06010 564FF7F (04073)	IORER	R4,R7,\$FF7F	'OR' IN HIGH 4 BITS
06012 84406C16 (04074)	MOVNML	R4,APC\$+W\$*APC\$V10	MOVE WORD TO BUS1
06014 87706C16 (04075)	LAF	R7,APC\$+W\$*APC\$V10	GET ADDR OF BUFFER START
06016 9C700B14 (04076)	ADDRR	R7,10*W\$	ADD HW OFFSET TO DESIRED SAMPLE
06018 87706C16 (04077)	SAP	R7,APC\$+W\$*APC\$V10	STORE ADDR OF DESIRED SAMPLE
06018 87706C16 (04078)			
06019 87706C16 (04079)			
0601A C04C0502 (04080)	V(11)...		
0601C 87706C12 (04081)	MOVNML	R4,BCT\$BA(R6)	18BIT LEN., 28BIT BUS, 17BIT BA.
0601E 564FF7F (04082)	MOVNR	R7,APC\$+W\$*APC\$V11	GET LEFT WORD OF INSTR
06020 84406C12 (04083)	IORER	R4,R7,\$FF7F	'OR' IN HIGH 4 BITS
06022 87706C12 (04084)	MOVNML	R4,APC\$+W\$*APC\$V11	MOVE WORD TO BUS1
06024 9C700B16 (04085)	LAF	R7,APC\$+W\$*APC\$V11	GET ADDR OF BUFFER START
06026 87706C12 (04086)	ADDRR	R7,11*W\$	ADD HW OFFSET TO DESIRED SAMPLE
06028 87706C12 (04087)	SAP	R7,APC\$+W\$*APC\$V11	STORE ADDR OF DESIRED SAMPLE
06028 87706C12 (04088)			
0602A C04C0502 (04089)	V(12)...		
0602C 87706C0E (04090)	MOVNML	R4,BCT\$BA(R6)	18BIT LEN., 28BIT BUS, 17BIT BA.
0602E 564FF7F (04091)	MOVNR	R7,APC\$+W\$*APC\$V12	GET LEFT WORD OF INSTR
06030 84406C0E (04092)	IORER	R4,R7,\$FF7F	'OR' IN HIGH 4 BITS
06032 87706C0E (04093)	MOVNML	R4,APC\$+W\$*APC\$V12	MOVE WORD TO BUS1
06034 9C700B18 (04094)	LAF	R7,APC\$+W\$*APC\$V12	GET ADDR OF BUFFER START
06036 87706C0E (04095)	ADDRR	R7,12*W\$	ADD HW OFFSET TO DESIRED SAMPLE
06038 87706C0E (04096)	SAP	R7,APC\$+W\$*APC\$V12	STORE ADDR OF DESIRED SAMPLE
06038 87706C0E (04097)			
0603A C04C0502 (04098)	V(13)...		
0603C 87706C0A (04099)	MOVNML	R4,BCT\$BA(R6)	18BIT LEN., 28BIT BUS, 17BIT BA.
0603E 564FF7F (04100)	MOVNR	R7,APC\$+W\$*APC\$V13	GET LEFT WORD OF INSTR
06040 84406C0A (04101)	IORER	R4,R7,\$FF7F	'OR' IN HIGH 4 BITS
06042 87706C0A (04102)	MOVNML	R4,APC\$+W\$*APC\$V13	MOVE WORD TO BUS1
06044 9C700B1A (04103)	LAF	R7,APC\$+W\$*APC\$V13	GET ADDR OF BUFFER START
06046 87706C0A (04104)	ADDRR	R7,13*W\$	ADD HW OFFSET TO DESIRED SAMPLE
06048 87706C0A (04105)	SAP	R7,APC\$+W\$*APC\$V13	STORE ADDR OF DESIRED SAMPLE
06048 87706C0A (04106)			
0604A C04C0502 (04107)	V(14)...		
0604C 87706C06 (04108)	MOVNML	R4,BCT\$BA(R6)	18BIT LEN., 28BIT BUS, 17BIT BA.
0604E 564FF7F (04109)	MOVNR	R7,APC\$+W\$*APC\$V14	GET LEFT WORD OF INSTR
06050 84406C06 (04110)	IORER	R4,R7,\$FF7F	'OR' IN HIGH 4 BITS
06052 87706C06 (04111)	MOVNML	R4,APC\$+W\$*APC\$V14	MOVE WORD TO BUS1
06054 9C700B1C (04112)	LAF	R7,APC\$+W\$*APC\$V14	GET ADDR OF BUFFER START
06056 87706C06 (04113)	ADDRR	R7,14*W\$	ADD HW OFFSET TO DESIRED SAMPLE
06058 87706C06 (04114)	SAP	R7,APC\$+W\$*APC\$V14	STORE ADDR OF DESIRED SAMPLE
06058 87706C06 (04115)			
0605A C04C0502 (04116)	W BASE BINDING		
0605C 87706C03 (04117)	MOVNR	R6,3*H\$(R1)	
0605E 564CF7F (04118)	MOVNR	R6,R6,MSK\$LBYT	
06060 3C67 (04119)	LPS	R6,7	
06062 0000 (04120)	EVEN		
06064 0000 (04121)			
06066 0000 (04122)			
06068 0000 (04123)	W(0)...		
0606A C04C0502 (04124)	MOVNML	R4,BCT\$BA(R6)	18BIT LEN., 28BIT BUS, 17BIT BA.
0606C 87706C3A (04125)	MOVNR	R7,APC\$+W\$*APC\$M0	GET LEFT WORD OF INSTR

GET BID
CONVERT TO OCT INDEX


```

0605C 564FF7F0 (04925) TORER R4,R7,5FFF0
0605E 84406C3A (04926) MOVNRL R4,APCS$+W$-APCS$W0
                                W(1)...
06060 C04C0502 (04928) MOVNRL R4,BCT$BA(R6)
06062 F0706C34 (04929) MOVNRL R7,APCS$+W$-APCS$W1
06064 564FF7F0 (04930) TORER R4,R7,5FFF0
06066 84406C3A (04931) MOVNRL R4,APCS$+W$-APCS$W1
06068 EF706C34 (04932) MOVNRL R7,APCS$+W$-APCS$W1
0606A 9C700002 (04933) LAF R7,1*W$
0606C EF706C34 (04934) ADDIR R7,APCS$+W$-APCS$W1
                                W(2)...
0606E C04C0502 (04936) MOVNRL R4,BCT$BA(R6)
06070 F0706C40 (04937) MOVNRL R7,APCS$+W$-APCS$W2
06072 564FF7F0 (04938) TORER R4,R7,5FFF0
06074 84406C40 (04939) MOVNRL R4,APCS$+W$-APCS$W2
06076 EF706C40 (04940) LAF R7,APCS$+W$-APCS$W2
06078 9C700004 (04941) ADDIR R7,2*W$
0607A EF706C40 (04942) SAF R7,APCS$+W$-APCS$W2
                                W(3)...
0607C C04C0502 (04944) MOVNRL R4,BCT$BA(R6)
0607E F0706C42 (04945) MOVNRL R7,APCS$+W$-APCS$W3
06080 564FF7F0 (04946) TORER R4,R7,5FFF0
06082 84406C42 (04947) MOVNRL R4,APCS$+W$-APCS$W3
06084 EF706C42 (04948) LAF R7,APCS$+W$-APCS$W3
06086 9C700006 (04949) ADDIR R7,3*W$
06088 EF706C42 (04950) SAF R7,APCS$+W$-APCS$W3
                                W(4)...
0608A C04C0502 (04952) MOVNRL R4,BCT$BA(R6)
0608C F0706C52 (04953) MOVNRL R7,APCS$+W$-APCS$W4
0608E 564FF7F0 (04954) TORER R4,R7,5FFF0
06090 84406C52 (04955) MOVNRL R4,APCS$+W$-APCS$W4
06092 EF706C52 (04956) LAF R7,APCS$+W$-APCS$W4
06094 9C700008 (04957) ADDIR R7,4*W$
06096 EF706C52 (04958) SAF R7,APCS$+W$-APCS$W4
                                W(5)...
06098 C04C0502 (04960) MOVNRL R4,BCT$BA(R6)
0609A F0706C4C (04961) MOVNRL R7,APCS$+W$-APCS$W5
0609C 564FF7F0 (04962) TORER R4,R7,5FFF0
0609E 84406C4C (04963) MOVNRL R4,APCS$+W$-APCS$W5
060A0 EF706C4C (04964) LAF R7,APCS$+W$-APCS$W5
060A2 9C70000A (04965) ADDIR R7,5*W$
060A4 EF706C4C (04966) SAF R7,APCS$+W$-APCS$W5
                                R BASE BINDING
060A6 F0620003 (04971) MOVNRL R6,3*HS(R1)
060A8 506C00FF (04972) MOVNRL R6,R6,MSK$RBYT
060AA 3A61 (04973) LLS R6,1
                                JGET BID
                                JCONVERT TU OCT INDEX

```

```

06DAB 0000 (04970) )
              (04979) )
              ) R(0) (INPUT)...
06DAC C04C0502 (04981) ) MOVNRL R4,BCT$BA(R6)
06DAE F0706022 (04982) ) MOVNR R7,APCS$+W$-APCS$18
06DAB 564E7F70 (04983) ) TORR R4,R7,$FF70
06D02 04406022 (04984) ) MOVNRL R4,APCS$+W$-APCS$18
              (04985) )
06D04 C04C0502 (04986) ) R(0) (OUTPUT)...
06D06 F0706C70 (04987) ) MOVNRL R4,BCT$BA(R6)
06D08 564E7F70 (04988) ) MOVNR R7,APCS$+W$-APCS$08
06D0A 04406C70 (04989) ) TORR R4,R7,$FF70
              (04990) ) MOVNRL R4,APCS$+W$-APCS$08
              (04991) )
              ) S BASE BINDING
              (04992) )
              (04993) )
06DBC F0620004 (04994) ) MOVNR R6,4*W$ (R1)
06DBE 506CFF70 (04995) ) MOVNR R6,R6,MSK$RBYT
06DC0 3C67 (04996) ) LRS R6,7
06DC1 0000 (04997) ) EVEN
              (04998) )
              ) S(0) (INPUT)...
06DC2 C04C0502 (05000) ) MOVNRL R4,BCT$BA(R6)
06DC4 F07060F0 (05001) ) MOVNR R7,APCS$+W$-APCS$18
06DC6 564E7F70 (05002) ) TORR R4,R7,$FF70
06DC8 044060F0 (05003) ) MOVNRL R4,APCS$+W$-APCS$18
              (05004) )
              ) S(0) (OUTPUT)...
06DCA C04C0502 (05005) ) MOVNRL R4,BCT$BA(R6)
06DCC F0706C08 (05006) ) MOVNR R7,APCS$+W$-APCS$08
06DCE 564E7F70 (05007) ) TORR R4,R7,$FF70
06DD0 04406C08 (05008) ) MOVNRL R4,APCS$+W$-APCS$08
              (05009) )
              ) T BASE BINDING
              (05010) )
              (05011) )
              (05012) )
06DD2 F0620004 (05013) ) MOVNR R6,4*W$ (R1)
06DD4 506C00F7 (05014) ) MOVNR R6,R6,MSK$RBYT
06DD6 3A61 (05015) ) LLS R6,1
06DD7 0000 (05016) ) EVEN
              (05017) )
              ) T(0) (INPUT)...
06DD8 C04C0502 (05018) ) MOVNRL R4,BCT$BA(R6)
06DDA F0706C04 (05019) ) MOVNR R7,APCS$+W$-APCS$18
06DDC 564E7F70 (05020) ) TORR R4,R7,$FF70
06DDE 04406C04 (05021) ) MOVNRL R4,APCS$+W$-APCS$18
              (05022) )
              ) T(1) (INPUT)...
              (05023) )
              (05024) )
06DF0 C04C0502 (05025) ) MOVNRL R4,BCT$BA(R6)
06DE2 F0706C08 (05026) ) MOVNR R7,APCS$+W$-APCS$18
06DE4 564E7F70 (05027) ) TORR R4,R7,$FF70
06DE6 04406C08 (05028) ) MOVNRL R4,APCS$+W$-APCS$18
06DE8 E7706C08 (05029) ) LAR R7,APCS$+W$-APCS$18
06DEA 9C700002 (05030) ) ADDR R7,1*W$

```

18BIT LEN., 28BIT BUS0, 170BIT BA.
GET LEFT WORD OF INSTR
'OR' IN HIGH 4 BITS
MOVE WORD TO BUS1

18BIT LEN., 28BIT BUS0, 170BIT BA.
GET LEFT WORD OF INSTR
'OR' IN HIGH 4 BITS
MOVE WORD TO BUS1

GET BID
CONVERT TO OCT INDEX

18BIT LEN., 28BIT BUS0, 170BIT BA.
GET LEFT WORD OF INSTR
'OR' IN HIGH 4 BITS
MOVE WORD TO BUS1

18BIT LEN., 28BIT BUS0, 170BIT BA.
GET LEFT WORD OF INSTR
'OR' IN HIGH 4 BITS
MOVE WORD TO BUS1

GET BID
CONVERT TO OCT INDEX

18BIT LEN., 28BIT BUS0, 170BIT BA.
GET LEFT WORD OF INSTR
'OR' IN HIGH 4 BITS
MOVE WORD TO BUS1

18BIT LEN., 28BIT BUS0, 170BIT BA.
GET LEFT WORD OF INSTR
'OR' IN HIGH 4 BITS
MOVE WORD TO BUS1
GET ADDR OF BUFFER START
ADD HV OFFSET TO DESIRED SAMPLE

06D0E EF706C08 (05031)	SAP	R7,APCS\$W\$*APCS\$1T1	STORE ADDR OF DESIRED SAMPLE
(05032)			
(05033)	T(2) (INPUT)...		
06D0E C04C0502 (05034)	MOVRL	R4,ACT\$BA(R6)	18BIT LEN., 2BIT BUS#, 17BIT BA.
06D0F F0706C0C (05035)	MOVRL	R7,APCS\$W\$*APCS\$1T2	GET LEFT WORD OF INSTR
06D0F 564FF7F0 (05036)	IORL	R4,R7,\$FF0	OR- IN HIGH 4 BITS
06D0F 84406C0C (05037)	MOVRL	R4,APCS\$W\$*APCS\$1T2	MOVE WORD TO BUS1
06D0F 8E706C0C (05038)	LAF	R7,APCS\$W\$*APCS\$1T2	GET ADDR OF BUFFER START
06D0F 9C70000A (05039)	ADDIR	R7,3-W\$	ADD HW OFFSET TO DESIRED SAMPLE
06D0F EF706C0C (05040)	SAP	R7,APCS\$W\$*APCS\$1T2	STORE ADDR OF DESIRED SAMPLE
(05041)			
(05042)	T(3) (INPUT)...		
06D0F C04C0502 (05043)	MOVRL	R4,ACT\$BA(R6)	18BIT LEN., 2BIT BUS#, 17BIT BA.
06D0F F0706C10 (05044)	MOVRL	R7,APCS\$W\$*APCS\$1T3	GET LEFT WORD OF INSTR
06D0F 564FF7F0 (05045)	IORL	R4,R7,\$FF0	OR- IN HIGH 4 BITS
06D0E 84406C10 (05046)	MOVRL	R4,APCS\$W\$*APCS\$1T3	MOVE WORD TO BUS1
06D0E EF706C10 (05047)	LAF	R7,APCS\$W\$*APCS\$1T3	GET ADDR OF BUFFER START
06D0E 9C70000A (05048)	ADDIR	R7,3-W\$	ADD HW OFFSET TO DESIRED SAMPLE
06D0E EF706C10 (05049)	SAP	R7,APCS\$W\$*APCS\$1T3	STORE ADDR OF DESIRED SAMPLE
(05050)			
(05051)	T(4) (INPUT)...		
06D0A C04C0502 (05052)	MOVRL	R4,ACT\$BA(R6)	18BIT LEN., 2BIT BUS#, 17BIT BA.
06D0C F0706C14 (05053)	MOVRL	R7,APCS\$W\$*APCS\$1T4	GET LEFT WORD OF INSTR
06D0E 564FF7F0 (05054)	IORL	R4,R7,\$FF0	OR- IN HIGH 4 BITS
06D10 84406C14 (05055)	MOVRL	R4,APCS\$W\$*APCS\$1T4	MOVE WORD TO BUS1
06D12 EF706C14 (05056)	LAF	R7,APCS\$W\$*APCS\$1T4	GET ADDR OF BUFFER START
06D14 9C700000 (05057)	ADDIR	R7,1-W\$	ADD HW OFFSET TO DESIRED SAMPLE
06D16 EF706C14 (05058)	SAP	R7,APCS\$W\$*APCS\$1T4	STORE ADDR OF DESIRED SAMPLE
(05059)			
(05060)	T(5) (INPUT)...		
06D10 C04C0502 (05061)	MOVRL	R4,ACT\$BA(R6)	18BIT LEN., 2BIT BUS#, 17BIT BA.
06D1A F0706C10 (05062)	MOVRL	R7,APCS\$W\$*APCS\$1T5	GET LEFT WORD OF INSTR
06D1C 564FF7F0 (05063)	IORL	R4,R7,\$FF0	OR- IN HIGH 4 BITS
06D1E 84406C10 (05064)	MOVRL	R4,APCS\$W\$*APCS\$1T5	MOVE WORD TO BUS1
06D20 EF706C10 (05065)	LAF	R7,APCS\$W\$*APCS\$1T5	GET ADDR OF BUFFER START
06D22 9C70000A (05066)	ADDIR	R7,5-W\$	ADD HW OFFSET TO DESIRED SAMPLE
06D24 EF706C10 (05067)	SAP	R7,APCS\$W\$*APCS\$1T5	STORE ADDR OF DESIRED SAMPLE
(05068)			
(05069)	T(6) (OUTPUT)...		
06D26 C04C0502 (05070)	MOVRL	R4,ACT\$BA(R6)	18BIT LEN., 2BIT BUS#, 17BIT BA.
06D28 F0706C76 (05071)	MOVRL	R7,APCS\$W\$*APCS\$0T0	GET LEFT WORD OF INSTR
06D2A 564FF7F0 (05072)	IORL	R4,R7,\$FF0	OR- IN HIGH 4 BITS
06D2C 84406C76 (05073)	MOVRL	R4,APCS\$W\$*APCS\$0T0	MOVE WORD TO BUS1
(05074)			
06D2E 00000F63 (05075)	JMP	AP\$ENDRO	

PAGE 113: (00NDJ)C0CA16200N16M.MSD.2, 29-Dec-80 14:30:40, ED: VOLFF
APU3 - GTHR(V,A,B,V,W)

```

(05076) * APJ3 - GTHR(V,A,U,B,V,W)
(05077) * KFIELD 9/80
(05078) *
(05079) * RUNS WITH GTHRSS
(05080) *
(05081) * FUNCTION:
(05082) * GATHER CODED TRANSMISSION DATA INTO OUTPUT BUFFER Y.
(05083) * Y <=
(05084) * ( 1) CODED APC RESIDUAL SAMPLES
(05085) * ( 3) CODED ENERGY
(05086) * ( 3) CODED DELTA GAINS
(05087) * ( 1) CODED PITCH LAG
(05088) * ( 3) CODED PITCH PREDICTOR CURFFS
(05089) * ( 6) CODED SPECTRAL PREDICTOR COEFFS
(05090) * ----
(05091) * (230)

```

ON FUNCTION CALL, V(9-215) IS EXPECTED TO CONTAIN
216 CODED APC RESIDUAL SAMPLES.
ALL PARAMETERS (BUFFERS & SCALARS) ARE TREATED AS FWD, LNC
(FOR REAL SCALARS, THIS IMPLIES USING THE LEFT WORD OF
THE 32 BIT SCALAR.)

```

V(216)          (= SA
V(217-219)    (= V(0-2)
V(220)        (= SB
V(221-223)    (= V(0-2)
V(224-229)    (= W(0-5)

SEQUENCE:
SA, V(0), V(1), V(2),
SB, V(0), V(1), V(3),
W(0), ..., W(5), [PI

```

INPUT SEQUENCE:

```

OUTPUT SEQUENCE:
Y(216), ..., Y(229), [EO]

```

EVEN	GTHRUSSA
DATA	GTHRUSSZ
DATA	

BEGIN APU(GTHRV)

```
MOV(101,00) \ NOP
JUMP(GTHRUSSA,EN)
```

CLEAR(RA)
JUMP(0)

END
= 0A-GTHRU\$A

06E30 0000
06E31 0004

100 06E32 08FC0000
101 06E34 901C0000

002 06E36 20322032
003 06E38 10000000
00000004
06E7A

ADDRESS	OPERATION	OPERANDS	REMARKS
05125	AP3 - CTRN(V,A,U,B,V,W)		
05126	KFIELD 9/80		
05127			
05128	INPUT SEQUENCE:		
05129	SA, U(0), U(1), U(2),		
05130	SB, V(0), V(1), V(3),		
05131	W(0), ..., W(5), [P1]		
05132			
05133	OUTPUT SEQUENCE:		
05134	V(216), ..., V(229), [EO]		
05135			
05136	ALL DATA ITEMS ARE FWD,LNG (TE FORMAT)		
05137			
05138	EVEN		
05139	ADDR	CTHRS\$1	1PTR TO CONS INS BLOCK
05140	ADDR	CTHRS\$+2*CTHRS\$	1SCALAR BLOCK
05141	DATA	2	1# OF SCALARS
05142	DATA	CTHRS\$2	1MODULE SIZE
05143	ADDR	CTHRS\$A	1PTR TO CHAIN ANCHOR
05144			
05145	CTHRS\$: BEGIN APS(CTHRS)		
05146	JSM(CTHRS\$0,P2)		
05147	SET(R0)		
05148			
05149	INPUT PCN		
05150			
05151	CTHRS\$:		
05152	LOAD(BR0,M\$\$(1),TE)		
05153	LOAD(BR0,M\$\$(1))		
05154			
05155	GEN U(0-2)		
05156			
05157	LOAD(BR1,C13,TE)		
05158	LOAD(BR2,M\$)		
05159	LOAD(BR2,M\$)		
05160			
05161	ADD(BR1,C97,TE)		
05162	ADD(BR1,C97,TE)		
05163			
05164	GEN SB		
05165			
05166	MOV(BR0,BR0,TE)		
05167			
05168	GEN V(0-2)		
05169			
05170	LOAD(BR1,C23,TE)		
05171	LOAD(BR2,M\$)		
05172	LOAD(BR2,M\$)		
05173			
05174	ADD(BR1,C107,TE)		
05175	ADD(BR1,C107,TE)		
05176			
05177	GEN W(0-5)		

PAGE 115: (00001)MC16>00016M.N50.2, 29-Dec-88 14:38:48, Ed: MOLF
AP33 - CTRN(Y,A,W,B,V,W)

A07 06260 1E503002	(05170) ;		
A10 04E62 20600000	(05179)	LOAD(RR1,C3)	LOAD W(0) ADDR
A11 04E64 22120000	(05100)	LOAD(RR2,M5)	JUMMV
	(05101)	SUB(RR1,M5)	LOAD W(-1) ADDR
A12 04E66 24600005	(05102) ;		JGEN 6 SAMPLES
A13 04E68 271A0000	(05103)	LOAD(RR2,6-1)	JGEN W ELEMENTY ADDR
A14 04E6A 20291301	(05104) 81:	ADD(RR1,C113,TE)	
	(05105)	SUBL(RR2,1),JUMPP(01)	
A15 04E6C 2A200031	(05106) ;		
A16 04E6E 2C000020	(05107)	CLEAR(R1)	DONE WITH INPUT
	(05108)	NOF(0)	
	(05109)	EJECT	

PAGE 116: (0000)<DC116>000106.M50.2, 79-Dec-80 14:30:40, Edt VOLP
 IPS3 - CTR(Y,A,W,B,V,W)

```

(05190) ;
(05191) ; OUTPUT PCM
(05192) ;
(05193) CTRSS01
A17 06E70 2E300032 (05194) SET(RA)
(05195) ;
(05196) ; GEN Y(216-229)
(05197) ;
A18 06E72 3050000A (05198) LOAD(0W1,C03)
A19 06E74 32600000 (05199) LOAD(0W2,M$)
A1A 06E76 34120000 (05200) SUB(0W1,M$)
(05201) ;
A1B 06E78 36110000 (05202) ADD(0W1,216)
A1C 06E7A 30600000 (05203) LOAD(0W2,19-1)
(05204) ;
A1D 06E7C 3A11000A (05205) #2: ADD(0W1,C03,TE)
A1E 06E7E 3C211001 (05206) SUBL(0W2,1),JUMPF(02)
(05207) ;
A1F 06E80 3E200030 (05208) CLEAR(RO)
A20 06E82 40000020 (05209) NOP(0)
(05210) ;
000067C (05211) CTRSSA-JC
(05212) ;
06E84 (05213) END
(05214) ;
06E84 00000000 (05215) CTRSS1 DATA 10P*0.0"
...
(05216) ;
0000056 (05217) CTRSS2=BL-CTRSS
MODULE SIZE

```

```

PAGE 117: (000010C0A16)00016M.H50.2; 29-Dec-88 14:38:48, Ed: VOLP
APU3 - THIST(V,U,V,W,R,S) SET UP INITTER FRAME HISTORIES
(05218) ' APU3 - THIST(V,U,V,W,R,S) SET UP INITTER FRAME HISTORIES
(05219) ' RTIELD 7/88
(05220) '
(05221) ' RUNS WITH APU3-THIST (THSS)
(05222) '
(05223) ' Y(0)-Y(WBS-1) <= U(0)-U(WBS-1)
(05224) ' V(0)-V(WBS-1) <= W(0)-W(WBS-1)
(05225) ' R(0)-R(WBS-1) <= S(0)-S(WBS-1)
(05226) '
(05227) ' (REQUIRES SPECIAL BINDING TO ACCOMMODATE R & S BUFFERS)
(05228) '
(05229) ' INPUT SEQUENCE:
(05230) ' U(0),...,U(WBS-1),
(05231) ' V(0),...,V(WBS-1),
(05232) ' S(0),...,S(WBS-1), EPI3
(05233) '
(05234) ' OUTPUT SEQUENCE:
(05235) ' Y(0),...,Y(WBS-1),
(05236) ' V(0),...,V(WBS-1),
(05237) ' R(0),...,R(WBS-1), (EO)
(05238) '
(05239) ' EVEN
(05240) ' DATA THUS$A
(05241) ' DATA THUS$Z
(05242) '
(05243) ' THUS$ BEGIN APU3(THISTW)
(05244) ' THUS$A: MOV(TQ1,OQ) \ NOP
(05245) ' JUMPC(THUS$A,EO)
(05246) '
(05247) ' CLEAR(MA)
(05248) ' JUMP(0)
(05249) ' THUS$Z = BA-THUS$A
(05250) ' END

```


PAGE 110: (BNND)DC116>BN16M.MSO.2, 29-Dec-88 14:30:40, Ed: WOLF
APS3 - THIST(Y,V,W,R,S) APS PROGRAM FOR THIST

```

(05251) APS3 - THIST(Y,V,W,R,S) APS PROGRAM FOR THIST
(05252) / RFIELD 7/80
(05253) /
(05254) / BINDING DONE BY SEMSTM TO ACCOMMODATE R & S BUFFERS
(05255) /
(05256) / INPUT SEQUENCE:
(05257) / U(0),...,U(WBS-1),
(05258) / V(0),...,V(WBS-1),
(05259) / S(0),...,S(RBS-1), [PI]
(05260) /
(05261) / OUTPUT SEQUENCE:
(05262) / Y(0),...,Y(WBS-1),
(05263) / V(0),...,V(WBS-1),
(05264) / R(0),...,R(RBS-1), [EO]
(05265) /
(05266) /
(05267) / ADDR 0 / (PTR TO CONS INS BLOCK)
(05268) / ADDR 0 / (SCALAR ADDRESS)
(05269) / DATA 0 / (NUMBER OF SCALARS)
(05270) / DATA THSZ / (MODULE SIZE)
(05271) / ADDR THSZ / (PTR TO CHAIN ANCHOR)
(05272) /
(05273) THSZ BEGIN APS(THIST)
(05274) JSN(THSZ,0,72) / SET UP AND START OUTPUT PROGRAM
(05275) SET(RO)
(05276) /
(05277) / INPUT PROGRAM
(05278) / BR0: SAMPLE ADDRESSES
(05279) / BR1: BUFFER SIZES
(05280) /
(05281) THSZ:
(05282) /
(05283) / DO W BUFFER...
(05284) THSZBR0: LOAD(BR0,MSS) /W BASE (WHEN BOUND)
(05285) THSZBR1: LOAD(BR1,MSS) /W SIZE-1 (WHEN BOUND)
(05286) THSZBR1: SUB(BR0,MSS) /W INCR (WHEN BOUND)
(05287) /
(05288) THSZBR12:
(05289) BR1 ADD(BR0,MSS,TF) /GEN W ADDRS (WHEN BOUND)
(05290) BR1 SUBL(BR1,1),JUMPP(BR1) /LOOP OBS TIMES
(05291) /
(05292) / DO W BUFFER...
(05293) THSZBR0: LOAD(BR0,MSS) /W BASE (WHEN BOUND)
(05294) THSZBR1: LOAD(BR1,MSS) /W SIZE-1 (WHEN BOUND)
(05295) THSZBR1: SUB(BR0,MSS) /W INCR (WHEN BOUND)
(05296) /
(05297) THSZBR12:
(05298) BR2 ADD(BR0,MSS,TF) /GEN W ADDRS (WHEN BOUND)
(05299) BR2 SUBL(BR1,1),JUMPP(BR2) /LOOP OBS TIMES
(05300) /
(05301) / DO S BUFFER...
(05302) THSZBR0: LOAD(BR0,MSS) /S BASE (WHEN BOUND)
(05303) THSZBR1: LOAD(BR1,MSS) /S SIZE-1 (WHEN BOUND)

```


(05354) - SM5TH SPECIAL BINDING MODULE FOR TWIST
(05355) ;
(05356) ; PCD FORMAT (16 BIT FORMAT SHOWN)

```

WORD-----LEFT BYTE-----RIGHT BYTE-----
01      PTR TO NEXT PCB AND FUNCTION LIST FLAG (LSB)
10      CONTROL BITS
11      201

```

PTN TO NEXT PCB AND FUNCTION LIST FLAG (LSD)
CONTROL BITS
261

18 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

[illegible]

```

(05390) )
(05391) ) V UPPER BINDING:
(05392) )
(05393) ) V BASE....
(05394) ) MOVNR R6,MS(R1)
(05395) ) MOVNR R6,R6,MSEX$LEFT JGET BIO
(05396) ) LMS R6,7 JCONVERT TO DCT INDEX
(05397) ) EVFM
(05398) ) MOVNPL R4,DCT$R8(R6) JIDIT LEN., 2BIT BUSB, 17BIT BA.
(05399) ) MOVNR R7,THSS$MS*THSS$BA JGET LEFT NWORD OF INSTA
(05400) ) TORNR R4,R7,$FF$0 J"OR" IN NICH 4 BITS
(05401) ) MOVNML R4,THSS$MS*THSS$BA JMOVE PCWORD TO BUSI
(05402) )
(05403) ) VSI...
(05404) ) MOVNR R4,DCT$AD(R6) JSAMPLE INCREMENT
(05405) ) SAP R4,THSS$MS*THSS$YS11
(05406) ) SAP R4,THSS$MS*THSS$YS12

```

[illegible]

```

PAGE 122: (000010C010)00010M.H00.2, 29-000-00 14:30:00, EDI WULF
          S0M5TH SPECIAL BINDING MODULE FOR THIST

00F64 F04C0605 (05460)      MOVNR R4,HS+BC$AD(R6)      )SIZE-1
00F66 E0406E0B (05461)      MOVNR R4,TH$+W$-TH$S0S1+H$
00F68 E0406E0F (05462)      MOVNR R4,TH$+W$-TH$S0S2+H$
                                (05463) )
                                (05464) )
00F6A F04C0604 (05465)      MOVNR R4,BC$AD(R6)      )SAMPLE INCREMENT
00F6C E0406E0C (05466)      SAF R4,TH$+W$-TH$S0S1
00F6E E0406E0E (05467)      SAF R4,TH$+W$-TH$S0S12
                                (05468) )
                                (05469) )
                                (05470) )
                                (05471) )
                                (05472) )
00F70 F0620003 (05473)      MOVNR R6,3*W$ (R1)
00F72 E06C0F0B (05474)      MOVNR R6,R6,MSK$0BVT )GET BID
00F74 3C67      LRS R6,7      )CONVERT TO BCT INDEX
00F75 0000      EVEN
00F76 C04C0502 (05476)      MOVNR R4,BC$0A(R6) )BIT LEN., 2BIT 0000, 17BIT 0A-
00F78 F0706E06 (05477)      MOVNR R7,TH$+W$-TH$S0BA )GET LEFT WORD OF INSTR
00F7A 540F0F08 (05478)      I0NR R4,R7,$FF00 )"OR" IN HIGH 4 BITS
00F7C 04060E06 (05479)      MOVNR R4,TH$+W$-TH$S0BA )MOVE FWORD TO 00S1
                                (05480) )
                                (05481) )
                                (05482) )
                                (05483) )
                                (05484) )
                                (05485) )
                                (05486) )
                                (05487) )
                                (05488) )
                                (05489) )
00F84 F0620003 (05489)      MOVNR R6,3*W$ (R1)
00F86 E06C0F0F (05490)      MOVNR R6,R6,MSK$0BVT )GET BID
00F88 3A61      LLS R6,1      )CONVERT TO BCT INDEX
00F89 0000      EVEN
00F8A C04C0502 (05493)      MOVNR R4,BC$0A(R6) )BIT LEN., 2BIT 0000, 17BIT 0A-
00F8C F0706E02 (05494)      MOVNR R7,TH$+W$-TH$S0BA )GET LEFT WORD OF INSTR
00F8E 540F0F08 (05495)      I0NR R4,R7,$FF00 )"OR" IN HIGH 4 BITS
00F90 04060E02 (05496)      MOVNR R4,TH$+W$-TH$S0BA )MOVE FWORD TO 00S1
                                (05497) )
                                (05498) )
                                (05499) )
                                (05500) )
                                (05501) )
                                (05502) )
                                (05503) )
                                (05504) )
                                (05505) )
                                (05506) )
                                (05507) )
                                (05508) )
                                (05509) )
00F92 F04C0605 (05499)      MOVNR R4,HS+BC$AD(R6)      )SIZE-1
00F94 E0406E05 (05500)      MOVNR R4,TH$+W$-TH$S0S1+H$
00F96 E0406E09 (05501)      MOVNR R4,TH$+W$-TH$S0S2+H$
                                (05502) )
                                (05503) )
                                (05504) )
                                (05505) )
                                (05506) )
                                (05507) )
                                (05508) )
                                (05509) )
00F9E 00000F03 (05509)      JMP IP$00R00
                                )SAMPLE INCREMENT

```

PAGE 173: CDBND1<DCAL6>00N16M.MSD.2, 29-DEC-88 14130:40, ED: VOLV
AP03 - SCLRES(V,A,U,V)

```

(05510) * AP3 = SCLRES(V,A,0,V)
(05511) * WNR 9/00
(05512)
(05513) * ROUTINE TO COMPUTE QUANTIZER SCALE FACTORS
(05514) * AND SCALE RESIDUAL
(05515) *
(05516) * INPUTS:
(05517) * V(1),...,V(216) = RECEIVED RESIDUAL SAMPLES
(05518) * U(1),U(2),U(3) = RECEIVED DELTA GAINS (DG(1))
(05519) * SA
(05520) * = RECEIVED OVERALL GAIN
(05521) *
(05522) * OUTPUT:
(05523) * V(1),...,V(216) = SCALED RESIDUAL SAMPLES
(05524) *
(05525) * PROCEDURE:
(05526) * COMPUTE CFAC(I)=SA*U(I), I=1,2,3
(05527) * SCALE RESIDUAL V(I)=V(I)*CFAC(J)
(05528) * J=1 FOR I=1,...,72
(05529) * J=2 FOR I=73,...,144
(05530) * J=3 FOR I=145,...,216
(05531) *
(05532) * INPUT SEQUENCE
(05533) * IQ1 SA,U(1),U(2),U(3),V(1),...,V(216)
(05534) *
(05535) * OUTPUT SEQUENCE
(05536) * DO: DUM,DUM,V(1),...,V(216)
(05537) *
(05538) * APU REGISTER USAGE
(05539) * ADDER REGISTERS NOT USED
(05540) *
(05541) * NO V(1)
(05542) * M1 DG(1)
(05543) * M2 DG(2)
(05544) * M3 DG(3)
(05545) * M4
(05546) * M5 SA
(05547) * M6 CFAC
(05548) *
(05549) * SCL - APU - INITIALIZATION
(05550) *
(05551) * EVEN
(05552) * DATA SCL$SA
(05553) * DATA SCL$Z
(05554) *
(05555) * SCL$: BEGIN APU(SCLRES)
(05556) * SCL$SA: MOV(IQA,M1)
(05557) * MOV(IQA,M2)
(05558) * MOV(IQA,M2)
(05559) * MOV(IQA,M3)
(05560) * 1ST SEGMENT
(05561) * MUL(M1,M5)
(05562) * CALL(SCL$M)
(05563) * 2ND SEGMENT
(05564) * MUL(M2,M5)
(05565) *
(05566) *
(05567) *
(05568) *
(05569) *
(05570) *
(05571) *
(05572) *
(05573) *
(05574) *
(05575) *
(05576) *
(05577) *
(05578) *
(05579) *
(05580) *
(05581) *
(05582) *
(05583) *
(05584) *
(05585) *
(05586) *
(05587) *
(05588) *
(05589) *
(05590) *
(05591) *
(05592) *
(05593) *
(05594) *
(05595) *
(05596) *
(05597) *
(05598) *
(05599) *
(05600) *
(05601) *
(05602) *
(05603) *
(05604) *
(05605) *
(05606) *
(05607) *
(05608) *
(05609) *
(05610) *
(05611) *
(05612) *
(05613) *
(05614) *
(05615) *
(05616) *
(05617) *
(05618) *
(05619) *
(05620) *
(05621) *
(05622) *
(05623) *
(05624) *
(05625) *
(05626) *
(05627) *
(05628) *
(05629) *
(05630) *
(05631) *
(05632) *
(05633) *
(05634) *
(05635) *
(05636) *
(05637) *
(05638) *
(05639) *
(05640) *
(05641) *
(05642) *
(05643) *
(05644) *
(05645) *
(05646) *
(05647) *
(05648) *
(05649) *
(05650) *
(05651) *
(05652) *
(05653) *
(05654) *
(05655) *
(05656) *
(05657) *
(05658) *
(05659) *
(05660) *
(05661) *
(05662) *
(05663) *
(05664) *
(05665) *
(05666) *
(05667) *
(05668) *
(05669) *
(05670) *
(05671) *
(05672) *
(05673) *
(05674) *
(05675) *
(05676) *
(05677) *
(05678) *
(05679) *
(05680) *
(05681) *
(05682) *
(05683) *
(05684) *
(05685) *
(05686) *
(05687) *
(05688) *
(05689) *
(05690) *
(05691) *
(05692) *
(05693) *
(05694) *
(05695) *
(05696) *
(05697) *
(05698) *
(05699) *
(05700) *
(05701) *
(05702) *
(05703) *
(05704) *
(05705) *
(05706) *
(05707) *
(05708) *
(05709) *
(05710) *
(05711) *
(05712) *
(05713) *
(05714) *
(05715) *
(05716) *
(05717) *
(05718) *
(05719) *
(05720) *
(05721) *
(05722) *
(05723) *
(05724) *
(05725) *
(05726) *
(05727) *
(05728) *
(05729) *
(05730) *
(05731) *
(05732) *
(05733) *
(05734) *
(05735) *
(05736) *
(05737) *
(05738) *
(05739) *
(05740) *
(05741) *
(05742) *
(05743) *
(05744) *
(05745) *
(05746) *
(05747) *
(05748) *
(05749) *
(05750) *
(05751) *
(05752) *
(05753) *
(05754) *
(05755) *
(05756) *
(05757) *
(05758) *
(05759) *
(05760) *
(05761) *
(05762) *
(05763) *
(05764) *
(05765) *
(05766) *
(05767) *
(05768) *
(05769) *
(05770) *
(05771) *
(05772) *
(05773) *
(05774) *
(05775) *
(05776) *
(05777) *
(05778) *
(05779) *
(05780) *
(05781) *
(05782) *
(05783) *
(05784) *
(05785) *
(05786) *
(05787) *
(05788) *
(05789) *
(05790) *
(05791) *
(05792) *
(05793) *
(05794) *
(05795) *
(05796) *
(05797) *
(05798) *
(05799) *
(05800) *
(05801) *
(05802) *
(05803) *
(05804) *
(05805) *
(05806) *
(05807) *
(05808) *
(05809) *
(05810) *
(05811) *
(05812) *
(05813) *
(05814) *
(05815) *
(05816) *
(05817) *
(05818) *
(05819) *
(05820) *
(05821) *
(05822) *
(05823) *
(05824) *
(05825) *
(05826) *
(05827) *
(05828) *
(05829) *
(05830) *
(05831) *
(05832) *
(05833) *
(05834) *
(05835) *
(05836) *
(05837) *
(05838) *
(05839) *
(05840) *
(05841) *
(05842) *
(05843) *
(05844) *
(05845) *
(05846) *
(05847) *
(05848) *
(05849) *
(05850) *
(05851) *
(05852) *
(05853) *
(05854) *
(05855) *
(05856) *
(05857) *
(05858) *
(05859) *
(05860) *
(05861) *
(05862) *
(05863) *
(05864) *
(05865) *
(05866) *
(05867) *
(05868) *
(05869) *
(05870) *
(05871) *
(05872) *
(05873) *
(05874) *
(05875) *
(05876) *
(05877) *
(05878) *
(05879) *
(05880) *
(05881) *
(05882) *
(05883) *
(05884) *
(05885) *
(05886) *
(05887) *
(05888) *
(05889) *
(05890) *
(05891) *
(05892) *
(05893) *
(05894) *
(05895) *
(05896) *
(05897) *
(05898) *
(05899) *
(05900) *
(05901) *
(05902) *
(05903) *
(05904) *
(05905) *
(05906) *
(05907) *
(05908) *
(05909) *
(05910) *
(05911) *
(05912) *
(05913) *
(05914) *
(05915) *
(05916) *
(05917) *
(05918) *
(05919) *
(05920) *
(05921) *
(05922) *
(05923) *
(0592
```



```

(05592) * APS3 - SCLRES(V,A,U,V)
(05593) * WNR 9/80
(05594) *
(05595) * APS ROUTINE FOR SCL
(05596) *
(05597) * INPUTS:
(05598) * IQ: SA,U(1),U(2),U(3),V(1),...,V(216)
(05599) * OUTPUT:
(05600) * OQ: DUM,DUM,V(1),...,V(216)
(05601) *
(05602) * INPUT PROGRAM REGISTER USAGE
(05603) * BR0 SA,U,V
(05604) * BR1 SIZE (UNUSED)
(05605) * BR2 COUNTER(72)
(05606) * BR3 COUNTER(3)
(05607) *
(05608) * HEADER BLOCK
(05609) * EVEN
(05610) * ADDR SCL$1
(05611) * ADDR SCL$+2*SCL$5
(05612) * DATA 1
(05613) * DATA SCL$2
(05614) * ADDR SCL$3A
(05615) * EVEN
(05616) *
(05617) * END OF HEADER BLOCK
(05618) *
(05619) * START OF APS PROGRAM
(05620) SCL$1: BEGIN APS(SCL$)
(05621) *
(05622) * JSM(SCL$OP,P2)
(05623) * SET(MO)
(05624) * INPUT SCALER
(05625) SCL$5: LOAD(BR0,M$$(1),TF)
(05626) *
(05627) * LOAD DELTA GAINS - U
(05628) * LOAD(BR0,C13,TF)
(05629) * LOAD(BR1,M$)
(05630) * ADD(BR0,M$$(TF)
(05631) * ADD(BR0,C13,TF)
(05632) *
(05633) * LOAD RESIDUAL SAMPLES
(05634) * LOAD(BR0,C23)
(05635) * LOAD(BR1,M$)
(05636) * SUB(BR0,M$)
(05637) * LOAD COUNTERS IN BR2 AND BR3
(05638) * LOAD(BR3,-1)
(05639) SCL$5:1: LOAD(BR2,72-2)
(05640) * LOOP FOR SEGMENT SAMPLES INPUT
(05641) SCL$5:2: ADD(BR0,C103,TF)
(05642) * SUBL(BR2,1),JUMPP(SCL$5:2)
(05643) * ADDL(BR0,M$$(TF),JUMPI(BA+1)
(05644) * SET(VI)

```

)PTR TO CONSTRUCTED INSTRUCTION BLOCK
)PTR TO SCALER BLOCK
)SET NUMBER OF SCALERS
)SET MODULE SIZE
)POINTER TO CHAIN ANCHOR
)START APS PRC ON FULL WORDS

)LOAD AND GEN SA ADDR(GAIN)

)LOAD AND GEN U(1) ADDR
) (UNUSED)
)GEN U(2) ADDR
)GEN U(3) ADDR

)LOAD V SA
) (UNUSED)
)PREPARE TO INC LATER

)BR3=B OF SEGMENTS - 1
)BR2=B OF SAMPLES/SEC - 2

)GEN V(1) ADDR
)DEC INCR AND JUMP
)ON LAST VALUE AND ENSURE SET VI
)SIGNAL END OF INPUT

PAGE 126: (ORNDJ) (CA16) (MMN) (MSD-2, 29-Dec-88 14:30:48, Ed: WOLF
 APS3 - SCLRES(Y,A,U,V)

A10 06FFC 20000020 (05645)	NOP(0)		
A11 06FFE 22390001 (05646)	SUBL(BR3,1),JUMPP(SCL\$1)		LOOP FOR 2ND AND 3RD SEGMENT
A12 07000 24200031 (05648)	CLEAR(R1)		
A13 07002 26000020 (05649)	NOP(0)		DOONE INPUT
	OUTPUT APS PROGRAM		
	REGISTER USAGE		
	Y BASE ADDR		
	Y SIZE (UNUSED), DUMMYS		
	COUNTER (72)		
	COUNTER (3)		
A14 07004 20300032 (05650)	SCL\$OP: SET(RA)		TURN ON APU
	LOAD OUTPUT SCALED RESIDUAL SAMPLES		
	LOAD(BV0,C03)		LOAD Y 0A
	LOAD(BV1,M\$)		(UNUSED)
	SUB(BV0,M\$)		PREPARE TO INC LATER
	LOAD COUNTERS IN BV2 AND BV3		
	LOAD(BV3,3-1)		BV3 = 3 OF SEGMENTS - 1
	SCL\$3: LOAD(BV2,72-2)		BV2 = 2 OF SAMPLES - 2
	LOAD(BV1,DV\$1),TF)		LOAD DUMMY ADDR
	LOAD(BV1,DV\$1),TF)		LOAD DUMMY ADDR
	ADD(BV0,C03,TF)		GEN Y(1) ADDR
	SCL\$4: SUBL(BV2,1),JUMPP(SCL\$4)		DEC COUNTER AND JUMP
	ADDL(BV0,M\$,TF)		GEN LIST V(1) OF SECH
	SUBL(BV3,1),JUMPP(SCL\$3)		JUMP FOR NEXT SECH
	CLEAR(R0)		
	NOP(0)		
	SCL\$A=BC		ASSIGN VALUE TO CHAIN ANCHOR
	END		
	6P-0.0"		BLOCK FOR CONSTR INSTRU
	SCL\$2=BL-SCL\$3		COMPUTE BUS1 MODULE SIZE

266

PAGE 120: (0000) <0C116> 00016M-MS0.2, 29-Dec-88 14:30:40, Ed: WOLF
AP03-PITSVM(V,A,U,V) 3-TAP PITCH SYNTHESIS FILTER

```

0702C 0000 (05730) DATA PITSUSSA JAPU START ADDRESS
0702D 0020 (05739) DATA PITSUSZ JSIZE
(05740)
(05741) PITSUSZ: BEGIN APU(PITSVM)
000 0702E 00E00000 (05742) MOV(IQA,M0) \ NOP
001 07030 00E00000 (05743) MOV(IQA,M4) \ NOP
002 07032 04000000 (05744) MUL(M0,M4) \ NOP
003 07034 00000000 (05745) MOV(P,A0) \ NOP
004 07036 31000000 (05746) ALIGN(A0) \ NOP
005 07038 009C0000 (05747) MOV(R,OQ) \ NOP
(05748) HERE WE WAIT ON OQE, PROCEED APS, READ/WRITE THE INSTRUCTION, WAIT
(05749) AND PROCEED THE APS AGAIN, AND THEN GET DOWN TO THE COMPUTATION.
006 0703A 90100000 (05750) B1: JUMPC(B1,OQE)
007 0703C 00000000 (05751) NOP
008 0703E 20372037 (05752) CLEAR(M1)
009 07040 00PC0000 (05753) MOV(IQA,OQ) \ NOP
00A 07042 90100000 (05754) B2: JUMPC(B2,OQE)
00B 07044 00000000 (05755) NOP
00C 07046 20372037 (05756) CLEAR(M1)
00D 07048 00EC00EE (05757) MOV(IQA,M4) \ MOV(IQA,M6)
00E 0704A 00E000EC (05758) MOV(IQA,M5) \ MOV(IQA,M4)
00F 0704C 00E000ED (05759) MOV(IQA,M6) \ MOV(IQA,M5)
(05760) MAIN LOOP
010 0704E 00E000E0 (05761) B3: MOV(IQA,M0)
011 07050 04104111 (05762) MOV(A1),MUL(M0,M4)
012 07052 49104910 (05763) MOV(A0),SUB(A0,A1)
013 07054 00F20000 (05764) MOV(IQA,A2) \ NOP
014 07056 000000F2 (05765) NOP \ MOV(IQA,A2)
015 07058 009C009C (05766) MOV(R,OQ)
016 0705A 00E900E9 (05767) MOV(IQA,M1)
017 0705C 04010401 (05768) MOV(A1),MUL(M1,M5)
018 0705E 49404940 (05769) SUB(A2,A1)
019 07060 00E000E0 (05770) MOV(IQA,M2) \ NOP
01A 07062 000000EA (05771) NOP \ MOV(IQA,M2)
01B 07064 05510551 (05772) MOV(A1),MUL(M2,M6)
01C 07066 49104910 (05773) MOV(A0),SUB(A0,A1)
01D 07068 901C0010 (05774) JUMPC(R3,ED)
01E 0706A 20372032 (05775) CLEAR(M4)
01F 0706C 10000000 (05776) JUMP(R0)
(05777) PITSUSZ:=BA-PITSUSSA
0706E 00000020 (05778) END

```

PAGE 129: (00NDJ) <DCAL6>BNHGM.MSO.2, 29-Dec-88 14:30:40, Ed: WOLF
APS3-PITSVN(Y,A,U,V) APS PROGRAM FOR PITSVN

```

(05779) * APS3-PITSVN(Y,A,U,V)      APS PROGRAM FOR PITSVN
(05780) ?
(05781) ? J. WOLF 7/1/88
(05782) ?
(05783) * THIS APS PROGRAM MODIFIES ONE INSTRUCTION IN ITSELF BY:
(05784) ? 0. READING PITCH FROM SCALAR A, AND FIXING IT
(05785) ? 1. WRITING IT INTO THE RN OF THE BUS-1 COPY OF THE INSTRUCTION
(05786) ? 2. WAITING ON ONE TO BE SURE IT'S THERE (PLUS 1 NOP THAT STORES
(05787) ? SAYS IS NECESSARY)
(05788) ? 3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
(05789) ? 4. WRITING IT TO APSMEM IN PSEUDOMEMORY
(05790) ? 5. WAITING FOR ONE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
(05791) ? PROCEED.
(05792) ?
(05793) * HEADER BLOCK
(05794) ?
(05795) EVEN
(05796) ADDR PITSS$1
(05797) ADDR PITSS1+2*PITSS$8
(05798) DATA 1
(05799) DATA PITSS$2
(05800) ADDR PITSS$A
(05801) EVEN
(05802) ?
(05803) * INPUT PROGRAM
(05804) ? BR0: V(X), THEN V(M-N+X)
(05805) ? BR1: U(W)
(05806) ? BR2: LOOP COUNTER, STARTS AT BUS-1
(05807) ?
(05808) PITSS1: BEGIN APS(PITSS)
(05809) JSM(PITSS0,P2)
(05810) SET(M0)
(05811) LOAD(BR0,SVTSM1(1),TF) JGEN (2**15) ADDR
(05812) PITSS$8: LOAD(BR0,M$$(1),TF) JGEN SA ADDRESS (PITCH)
(05813) SET(M1)
(05814) NOP(B0)
(05815) LOAD(BR0,PITSS$M(1),TF) JLOAD MODIFIED INSTR THRU APV INTO APS MEMORY
(05816) SET(M1)
(05817) NOP
(05818) LOAD(BR0,C23,TF)
(05819) LOAD(BR1,M$$(1))
(05820) ADD(BR0,M$$(1),TF)
(05821) ADD(BR0,C103,TF) JGEN V(2)
(05822) * NOW LOAD INFO FOR ACCESSING THE Y AND W BUFFERS
(05823) LOAD(BR0,C03)
(05824) LOAD(BR2,M$$(1))
(05825) LOAD(BR2,M$$(1))
(05826) ADD(BR0,216*W$)
(05827) * NEXT INSTRUCTION GETS THE PITCH JAMMED INTO IT
(05828) PITSS$M=M1
(05829)
(05830) LOAD(BR2,M$$(1))
(05831) ADDR(BR2,BR2)
(05832) SUBB(BR0,BR2)
(05833)
(05834) * PTR TO CONSTRUCTED INSTRUCTION BLOCK
(05835) * PTR TO SCALAR BLOCK
(05836) * NUMBER OF SCALARS
(05837) * MODULE SIZE
(05838) * PTR TO CHAIN ANCHOR
(05839)
(05840) * SET AND START OUTPUT PROGRAM
(05841)
(05842) * LOAD PITCH
(05843) * 2*PITCH FOR FULLWORD ADDRESSING
(05844) * POINT TO V(0-M)

```

```

PAGE 1301 (00000)DCAL16>00016M.MS0.2, 29-Dec-88 14:30:40, Ed: WOLF
          AP33-PITSSN(V,A,W,V)      APS PROGRAM FOR PITSSN

A14 0709E 2050100E (05032)      LOAD(R01,C13)
A15 070A0 2A600000 (05033)      LOAD(R02,M$)
A16 070A2 2C120000 (05034)      SUB(R01,M$)
                                     JMAIN LOOP
A17 070A4 2C09003A (05036) 04: ADDL(R00,M$,TF)
A18 070A6 30919000 (05037)      ADD(R01,C93,TF)
A19 070A8 329A9002 (05038)      ADD(R01,C93,TF)
A1A 070AA 3A090032 (05039)      SUBL(R00,M$,TF)
A1B 070AC 36090032 (05040)      SUBL(R00,M$,TF)
A1C 070AE 3009003E (05041)      ADDL(R00,3-M$,TF)
A1D 070B0 3A291702 (05042)      SUML(R02,2),JUNPP(04)
A1E 070B2 3C200031 (05043)      CLEAR(R1)
A1F 070B4 3E000020 (05044)      NOP(0)
                                     (05045)
                                     (05046) } OUTPUT PROGRAM
                                     (05047) } BW0: V(M)
                                     (05048) } BW1: LOOP COUNTER (STARTS AT WBS-1)
                                     (05049) } BW3: SCRATCH
                                     (05050) }
A20 070B6 40300032 (05051)      PITSS0: SET(RA)
A21 070B8 43727099 (05052)      JGENERATE BUS1 ADR FOR THE INSTRUCTION TO BE MODIFIED (RIGHT HALF)
A22 070BA 44F37FC0 (05053)      LOAD(RW3,PITSSM+M$(1),TF)
A23 070BC 46F20794 (05054)      JGENERATE PSEUDOMEMORY ADR FOR LOADING THE APS MEMORY
A24 070BE 48F20794 (05055)      LOAD(RW3,APSMEH(1),TF)
A25 070C0 4A400010 (05056)      JNOW GET ON WITH THE "ORDINARY" OUTPUT
A26 070C2 4C700000 (05057)      LOAD(RW3,DW$(1),TF)
A27 070C4 4E020000 (05058)      LOAD(RW3,DW$(1),TF)
A28 070C6 500A0100 (05059)      LOAD(RW0,C03)
A29 070C8 52701000 (05060)      LOAD(RW0,C03)
A2A 070CA 54500000 (05061)      LOAD(RW0,M$)
A2B 070CC 56320000 (05062)      SUB(RW0,M$)
A2C 070CE 5801003A (05063)      ADDL(RW0,M$,TF)
A2D 070D0 5A112CB1 (05064)      SUBL(RW1,1),JUNPP(05)
A2E 070D2 5C200030 (05065)      CLEAR(R0)
A2F 070D4 5E000020 (05066)      NOP(0)
A30 070D6 60070C00 (05067)      END
                                     PITSS$A=EC
070D6 00000000 (05072) PITSS$1: DATA 0F'0.0'
...
00000070 (05074) PITSS$2=0L-PITSS1

```

JY(M-M+1) (REMEMBER BUFFER V IS COMPACT)

JUBA
JUBS-1
JU INCR

JY(M-M+1)
JU(M)
JU(M+1)
JY(M-M)
JY(M-M-1)
JY(M-M+2)
JVE-RE DOING 2 POINTS AT ONCE

JSTART APS
JGENERATE BUS1 ADR FOR THE INSTRUCTION TO BE MODIFIED (RIGHT HALF)

LOAD(RW3,PITSSM+M\$(1),TF)
JGENERATE PSEUDOMEMORY ADR FOR LOADING THE APS MEMORY

LOAD(RW3,APSMEH(1),TF)
JNOW GET ON WITH THE "ORDINARY" OUTPUT

LOAD(RW3,DW\$(1),TF)
LOAD(RW3,DW\$(1),TF)
LOAD(RW0,C03)
LOAD(RW0,C03)
SUB(RW0,M\$)
ADD(RW0,216*W\$)
LOAD(RW3,C13)
LOAD(RW1,M\$)
SUB(RW3,M\$)

JUBA
(UNUSED) YBS-1
JU INCR
JPOINT TO 1ST OUTPUT SAMPLE-1 IN Y
(UNUSED) WBA
JUBS-1
(UNUSED) 0 INCR

JASSIGN VALUE TO CHAIN ANCHOR
JBLOCK FOR CONSTRUCTED INSTRUCTIONS

JCOMPUTE BUS1 MODULE SIZE

PAGE 131: (0000)<0CAL0>00010M-MSU.2, 29-00C-00 14:30:40, E01 VOLV
APU3 - DEAL(V,A,W,B,V,W)

```

(05075) * APU3 - DEAL(V,A,W,B,V,W)
(05076) * FIELD 9/80
(05077) *
(05078) * RUNS WITH DEAL$
(05079) *
(05080) * FUNCTION:
(05081) * DEAL DECODED RECEIVED DATA FROM INPUT BUFFER V.
(05082) * V ->
(05083) * ( 1) DECODED APC RESIDUAL SAMPLES
(05084) * ( 3) DECODED ENERGY
(05085) * ( 3) DECODED DELTA GAINS
(05086) * ( 1) DECODED PITCH LAG
(05087) * ( 3) DECODED PITCH PREDICTOR COEFFS
(05088) * ( 6) DECODED SPECTRAL PREDICTOR COEFFS
(05089) * -----
(05090) * (230)

```

THE 216 DECODED APC RESIDUAL SAMPLES ARE NOT MOVED FROM THE V
BUFFER. THEY ARE SUBSEQUENTLY REFERENCED DIRECTLY AS V(0-215).
ALL PARAMETERS (BUFFERS & SCALARS) ARE TREATED AS BL,LNG

```

V(216) => SA
V(217-219) => U(0-2)
V(220) => SB
V(221-223) => V(0-2)
V(224-229) => W(0-5)

```

INPUT SEQUENCE:
V(216), ..., V(229), (PI)

OUTPUT SEQUENCE:
SA, U(0), W(1), W(2),
SB, V(0), V(1), V(3),
W(0), ..., W(5), (EO)

```

EVEN DEAL$SA
DATA DEAL$SA
DATA DEAL$SZ

```

```

DEAL$S BEGIN APU(DEAL$)
DEAL$SA:
MOV(10A,00) \ NOP
JUMPC(DEAL$SA,EO)

```

*
;COPY INPUT TO OUTPUT
;LOOP TIL DONE

```

CLEAR(RA)
JUMP(0)
DEAL$SZ = RA-DEAL$SA
END

```

070E6 0000
070E7 0004

A00 070E8 00FC0000
A01 070FA 90C00000
A02 070FC 2032032
A03 070FE 10000000
00000004
070FF
05921


```
(06015) * APU3-DCORZ DISCRETE CORRELATION APU PROGRAM
(06016) * BINDS TO APS3-0211A
(06017) *
(06018) * EQ: V(J)=SUM(U(K+J)*V(K)) FOR K=0,1,2,...,M-1) FOR J=0,1,2,...,N-1
(06019) * WHERE M=SIZE OF REFERENCE, AND N=SIZE OF OUTPUT
(06020) *
(06021) * LEADING AND TRAILING ZEROS ARE AUTOMATICALLY APPLIED BY THE APS PROGRAM
(06022) *
(06023) * ALGORITHM CALCULATES FOUR RUNNING SUMS FOR FOUR SUCCESSIVE POSITIONS
(06024) * BETWEEN THE REFERENCE FUNCTION AND THE INPUT, AS FOLLOWS:
(06025) *
(06026) * S1=U(K)*V(K)+U(K+1)*V(K+1)+U(K+2)*V(K+2)+U(K+3)*V(K+3)+...
(06027) * S2=U(K)+U(K-1)*U(K+1)+U(K+2)+U(K+3)+U(K+4)+...
(06028) * S3=U(K)+U(K-2)+U(K+1)+U(K+2)+U(K+3)+U(K+4)+...
(06029) * S4=U(K)+U(K-3)+U(K+1)+U(K+2)+U(K+3)+U(K+4)+...
(06030) *
(06031) *
(06032) * REGISTER ASSIGNMENTS ARE AS FOLLOWS:
(06033) *
(06034) * INPUT-K K+1 K+2 K+3
(06035) * M0: U(K) ----- U(K+2) -----
(06036) * M1: ----- U(K+1) ----- V(K+3)
(06037) * M4: V(K) -----
(06038) * M5: V(K-3) V(K+1) -----
(06039) * M6: V(K-2) ----- V(K+2) -----
(06040) * M7: V(K-1) ----- V(K+3)
(06041) *
(06042) * A1=S1(K), A2=S2(K-1), A3=S3(K-2), A4=S4(K-3)
(06043) *
(06044) *
(06045) *
```

```
00000003
0714E 0000
0714F 0005

A00 07150 00110012 DCMR$A MOV(ZERO,A1)\MOV(ZERO,A2) ZERO SUMS S1,S2
A01 07152 00130014 MOV(ZERO,A3)\MOV(ZERO,A4) ZERO SUMS S3,S4
A02 07154 00090009 MOV(ZERO,M1) ZERO U(-1)
A03 07156 00000000 MOV(ZERO,M5) ZERO V(-3)
A04 07158 00070002 MOV(ZERO,M6) ZERO V(-2)
A05 0715A 00090007 MOV(ZERO,M7) ZERO V(-1)
A06 0715C 04E004C0 MUL(M1,M7)\MUL(M1,M6) PS1(-1)PS2(-2)
A07 0715E 02600200 R(A3)\R(A4) S3(-3)S4(-4)
A08 07160 911C0007 JUMPS(DCEND,EO)
(06062) *
(06063) * INNER LOOP CALCULATES FOUR SUMS AT A TIME
(06064) * FOR FOUR PAIRS OF U,V PER PASS FOR K=0,4,8,...,M-1
(06065) *
(06066) *
A09 07162 00E000E0 DCMR0 MOV(IQA,M0) M0=U(K)
```

A0A 07164 04550435 (06060)	MOV(A5), MUL(M0, M6) \ MOV(A5), MUL(M0, M5)	A5=P01(K-1), P03(K-2) \ A5=P02(K-2), P0
A0B 07166 4534554 (06069)	MOV(A3), ADD(A1, A5) \ MOV(A4), ADD(A2, A5)	A3=P03(K-3), S1(K-1) \ A4=S4(K-4), S2(K-
A0C 07168 08C008C (06070)	MOV(IQ1, M4)	M4=V(K)
A0D 0716A 04168476 (06071)	MOV(A6), MUL(M0, M4) \ MOV(A6), MUL(M0, M7)	A6=P03(K-2), P01(K) \ A6=P04(K-3), P02(K
A0E 0716C 46714692 (06072)	MOV(A1), ADD(A3, A6) \ MOV(A2), ADD(A4, A6)	A1=S1(K-1), S3(K-2) \ A2=S2(K-2), S4(K-
A0F 0716E 91160026 (06073)	JUMPS(DCCL0, PVI)	EXIT IF K=M-1 (LAST REF SAMPLE INDE
A10 07170 08C900E9 (06074)	•	
A11 07172 04F50405 (06075)	DCOR1	M1=0(K+1)
A12 07174 4534554 (06076)	MOV(A5), MUL(M1, M7) \ MOV(A5), MUL(M1, M6)	A5=P01(K), P03(K-1) \ A5=P02(K-1), P04(K
A13 07176 08C008C (06077)	MOV(A3), ADD(A1, A5) \ MOV(A4), ADD(A2, A5)	A3=P03(K-2), S1(K) \ A4=S4(K-3), S2(K-1)
A14 07178 04060496 (06078)	MOV(IQ1, M5)	M5=V(K+1)
A15 0717A 04060496 (06079)	MOV(A6), MUL(M1, M5) \ MOV(A6), MUL(M1, M4)	A6=P03(K-1), P01(K+1) \ A6=P04(K-2), P0
A16 0717C 46714692 (06080)	MOV(A1), ADD(A3, A6) \ MOV(A2), ADD(A4, A6)	A1=S1(K), S3(K-1) \ A2=S2(K-1), S4(K-2)
A17 0717E 91160030 (06081)	JUMPS(DCCL1, PVI)	EXIT IF K+1=M-1 (LAST REF SAMP INDE
A18 0717F 08C900E9 (06082)	•	
A19 07180 04F50405 (06083)	DCOR2	M0=0(K+2)
A20 07182 4534554 (06084)	MOV(A5), MUL(M0, M4) \ MOV(A5), MUL(M0, M7)	A5=P01(K+1), P03(K) \ A5=P02(K), P04(K-
A21 07184 08C008C (06085)	MOV(A3), ADD(A1, A5) \ MOV(A4), ADD(A2, A5)	A3=P03(K-1), S1(K+1) \ A4=S4(K-2), S2(K
A22 07186 04508436 (06086)	MOV(IQ1, M6)	M6=V(K+2)
A23 07188 04508436 (06087)	MOV(A6), MUL(M0, M6) \ MOV(A6), MUL(M0, M5)	A6=P03(K), P01(K+2) \ A6=P04(K-1), P02(K
A24 0718A 46714692 (06088)	MOV(A1), ADD(A3, A6) \ MOV(A2), ADD(A4, A6)	A1=S1(K+1), S3(K) \ A2=S2(K+1), S4(K-1)
A25 0718C 91160054 (06089)	JUMPS(DCCL2, PVI)	EXIT IF K+2=M-1 (LAST REF SAMP INDE
A26 0718E 08C900E9 (06090)	•	
A27 07190 04F50405 (06091)	DCOR3	M1=0(K+3)
A28 07192 4534554 (06092)	MOV(A5), MUL(M1, M5) \ MOV(A5), MUL(M1, M4)	A5=P01(K+2), P03(K+1) \ A5=P02(K+1), P0
A29 07194 08C008C (06093)	MOV(A3), ADD(A1, A5) \ MOV(A4), ADD(A2, A5)	A3=P03(K), S1(K+2) \ A4=S4(K-1), S2(K+1)
A30 07196 04508406 (06094)	MOV(IQ1, M7)	M7=V(K+3)
A31 07198 04508406 (06095)	MOV(A6), MUL(M1, M7) \ MOV(A6), MUL(M1, M6)	A6=P03(K+1), P01(K+3) \ A6=P04(K), P02(K
A32 0719A 46714692 (06096)	MOV(A1), ADD(A3, A6) \ MOV(A2), ADD(A4, A6)	A1=S1(K+2), S3(K+1) \ A2=S2(K+1), S4(K
A33 0719C 91160068 (06097)	JUMPS(DCCL3, PVI)	EXIT IF K+3=M-1 (LAST REF SAMP INDE
A34 0719E 08C900E9 (06098)	•	
A35 0719F 10000000 (06099)	JUMP(DCORG)	
A36 071A0 20372037 (06100)	•	
A37 071A2 08C900E9 (06101)	ENTRY FOR K=M-1 CLEAN-UP	
A38 071A4 08C900E9 (06102)	CLEAR(M1)	
A39 071A6 04508405 (06103)	MOV(IQ1, M1)	
A40 071A8 04508405 (06104)	MOV(A5), MUL(M1, M7) \ MOV(A5), MUL(M1, M6)	M1=0(M)
A41 071AA 08C008C (06105)	MOV(A3), ADD(A1, A5) \ MOV(A4), ADD(A2, A5)	A5=P01(M-1), P03(M-2) \ A5=P02(M-2), P0
A42 071AC 04508406 (06106)	MOV(A6), MUL(M1, M5) \ MOV(A6), MUL(M1, M4)	A3=P03(M-3), S1(M-1) \ A4=S4(M-4), S2(M-
A43 071AE 46714692 (06107)	MOV(OQ), ADD(A3, A6) \ MOV(A2), ADD(A4, A6)	A6=P03(M-2), P01(M) \ A6=P04(M-3), P02(M
A44 071B0 08110011 (06108)	MOV(ZERO, A1)	OUT=S1(M-1), S3(M-2) \ A2=S2(M-2), S4(M
A45 071B2 08C900E9 (06109)	•	CLEAR S1 FOR NEW SUM
A46 071B4 08C900E9 (06110)	MOV(IQ1, M0)	M0=0(M+1)
A47 071B6 04508405 (06111)	MOV(A5), MUL(M0, M4) \ MOV(A5), MUL(M0, M7)	A5=P01(M), P03(M-1) \ A5=P02(M-1), P04(M
A48 071B8 4534554 (06112)	MOV(A3), ADD(A1, A5) \ MOV(A4), ADD(A2, A5)	A3=P03(M-2), S1(M) \ A4=S4(M-3), S2(M-1)
A49 071BA 04508436 (06113)	MOV(A6), MUL(M0, M6) \ MOV(A6), MUL(M0, M5)	A6=P03(M-1), P01(M+1) \ A6=P04(M-2), P0
A50 071BC 911C0002 (06114)	JUMPS(DCEND, EU)	JUMP TO END IF NO MORE OUTPUT
A51 071BE 4671469C (06115)	MOV(A1), ADD(A3, A6) \ MOV(OQ), ADD(A4, A6)	A1=S1(M), S3(M-1) \ OUT=S2(M-1), S4(M-2
A52 071B0 08120012 (06116)	MOV(ZERO, A2)	CLEAR S2 FOR NEW SUM
A53 071B2 08C900E9 (06117)	•	
A54 071B4 08C900E9 (06118)	MOV(IQ1, M1)	M1=0(M+2)
A55 071B6 04508405 (06119)	MOV(A5), MUL(M1, M5) \ MOV(A5), MUL(M1, M4)	A5=P01(M+1), P03(M) \ A5=P02(M), P04(M-
A56 071B8 911C0002 (06120)	JUMPS(DCEND, EU)	JUMP TO END IF NO MORE OUTPUT


```

PAGE 130:  (0000)(<C416>00016M,N50.2, 29-Dec-68 14130140, Ed: WULF
APS3--ADCR APS PROGRAM FOR DISCRETE CORRELATION WITH AUTO ZERO FILL
(06220) * APS3-ADCR APS PROGRAM FOR DISCRETE CORRELATION WITH AUTO ZERO FILL
(06221) JBUFFER ATTRIBUTES FOR INITIAL TESTING
00000000 (06222) VBASE=0
00000032 (06223) VBASE=50
00000064 (06224) VBASE=100
(06225) J
0000000C (06226) VZ=12
00000005 (06227) VZ=5
00000007 (06228) VZ=15
00000007 (06229) VZ=2
00000002 (06230) VZ=2
00000002 (06231) VZ=2
00000002 (06232) VZ=2
(06233) J
(06234) J BINDING DONE BY SBOCORZ
(06235) J
(06236) J
(06237) J
(06238) J
(06239) J
(06240) J
(06241) J
(06242) J
(06243) J
(06244) J
0725A 00000000 (06245) J
0725C 0000726C (06246) J
0725E 0000 (06247) J
0725F 000C (06248) J
07260 00000000 (06249) J
(06250) J
(06251) J
(06252) J
(06253) J
000 07262 00204240 (06254) J
001 07264 02300030 (06255) J
(06256) J
002 07266 04640032 (06257) J
003 07268 06700004 (06258) J
004 0726A 00220002 (06259) J
(06260) J
005 0726C 0A500002 (06261) J
(06262) J
006 0726E 0C390039 (06263) J
007 07270 0E1900C1 (06264) J
(06265) J
008 07272 10C2078A (06266) J
009 07274 17A00002 (06267) J
00A 07276 141900C1 (06268) J
00B 07278 16390001 (06269) J
(06270) J
00C 0727A 10440000 (06271) J
00D 0727B 10440000 (06272) J
(06273) J
(06274) J
(06275) J
(06276) J
(06277) J
(06278) J
(06279) J
(06280) J
(06281) J
(06282) J
(06283) J
(06284) J
(06285) J
(06286) J
(06287) J
(06288) J
(06289) J
(06290) J
(06291) J
(06292) J
(06293) J
(06294) J
(06295) J
(06296) J
(06297) J
(06298) J
(06299) J
(06300) J
(06301) J
(06302) J
(06303) J
(06304) J
(06305) J
(06306) J
(06307) J
(06308) J
(06309) J
(06310) J
(06311) J
(06312) J
(06313) J
(06314) J
(06315) J
(06316) J
(06317) J
(06318) J
(06319) J
(06320) J
(06321) J
(06322) J
(06323) J
(06324) J
(06325) J
(06326) J
(06327) J
(06328) J
(06329) J
(06330) J
(06331) J
(06332) J
(06333) J
(06334) J
(06335) J
(06336) J
(06337) J
(06338) J
(06339) J
(06340) J
(06341) J
(06342) J
(06343) J
(06344) J
(06345) J
(06346) J
(06347) J
(06348) J
(06349) J
(06350) J
(06351) J
(06352) J
(06353) J
(06354) J
(06355) J
(06356) J
(06357) J
(06358) J
(06359) J
(06360) J
(06361) J
(06362) J
(06363) J
(06364) J
(06365) J
(06366) J
(06367) J
(06368) J
(06369) J
(06370) J
(06371) J
(06372) J
(06373) J
(06374) J
(06375) J
(06376) J
(06377) J
(06378) J
(06379) J
(06380) J
(06381) J
(06382) J
(06383) J
(06384) J
(06385) J
(06386) J
(06387) J
(06388) J
(06389) J
(06390) J
(06391) J
(06392) J
(06393) J
(06394) J
(06395) J
(06396) J
(06397) J
(06398) J
(06399) J
(06400) J
(06401) J
(06402) J
(06403) J
(06404) J
(06405) J
(06406) J
(06407) J
(06408) J
(06409) J
(06410) J
(06411) J
(06412) J
(06413) J
(06414) J
(06415) J
(06416) J
(06417) J
(06418) J
(06419) J
(06420) J
(06421) J
(06422) J
(06423) J
(06424) J
(06425) J
(06426) J
(06427) J
(06428) J
(06429) J
(06430) J
(06431) J
(06432) J
(06433) J
(06434) J
(06435) J
(06436) J
(06437) J
(06438) J
(06439) J
(06440) J
(06441) J
(06442) J
(06443) J
(06444) J
(06445) J
(06446) J
(06447) J
(06448) J
(06449) J
(06450) J
(06451) J
(06452) J
(06453) J
(06454) J
(06455) J
(06456) J
(06457) J
(06458) J
(06459) J
(06460) J
(06461) J
(06462) J
(06463) J
(06464) J
(06465) J
(06466) J
(06467) J
(06468) J
(06469) J
(06470) J
(06471) J
(06472) J
(06473) J
(06474) J
(06475) J
(06476) J
(06477) J
(06478) J
(06479) J
(06480) J
(06481) J
(06482) J
(06483) J
(06484) J
(06485) J
(06486) J
(06487) J
(06488) J
(06489) J
(06490) J
(06491) J
(06492) J
(06493) J
(06494) J
(06495) J
(06496) J
(06497) J
(06498) J
(06499) J
(06500) J
(06501) J
(06502) J
(06503) J
(06504) J
(06505) J
(06506) J
(06507) J
(06508) J
(06509) J
(06510) J
(06511) J
(06512) J
(06513) J
(06514) J
(06515) J
(06516) J
(06517) J
(06518) J
(06519) J
(06520) J
(06521) J
(06522) J
(06523) J
(06524) J
(06525) J
(06526) J
(06527) J
(06528) J
(06529) J
(06530) J
(06531) J
(06532) J
(06533) J
(06534) J
(06535) J
(06536) J
(06537) J
(06538) J
(06539) J
(06540) J
(06541) J
(06542) J
(06543) J
(06544) J
(06545) J
(06546) J
(06547) J
(06548) J
(06549) J
(06550) J
(06551) J
(06552) J
(06553) J
(06554) J
(06555) J
(06556) J
(06557) J
(06558) J
(06559) J
(06560) J
(06561) J
(06562) J
(06563) J
(06564) J
(06565) J
(06566) J
(06567) J
(06568) J
(06569) J
(06570) J
(06571) J
(06572) J
(06573) J
(06574) J
(06575) J
(06576) J
(06577) J
(06578) J
(06579) J
(06580) J
(06581) J
(06582) J
(06583) J
(06584) J
(06585) J
(06586) J
(06587) J
(06588) J
(06589) J
(06590) J
(06591) J
(06592) J
(06593) J
(06594) J
(06595) J
(06596) J
(06597) J
(06598) J
(06599) J
(06600) J
(06601) J
(06602) J
(06603) J
(06604) J
(06605) J
(06606) J
(06607) J
(06608) J
(06609) J
(06610) J
(06611) J
(06612) J
(06613) J
(06614) J
```


PAGE 140: (BRND)<OCAL6>BMMIGH.MSD.2, 29-Dec-88 14:30:40, R4: WOLF
 APS3-ADOCOR APS PROGRAM FOR DISCRETE CORRELATION WITH AUTO ZERO FILL

```

A37 07200 66C207BA (06326)      LOAD(BR0,ZERO(1),TF)
      (06327) ,
A38 07202 70640032 (06320) ADC$13: LOAD(BR2,YBASE(2))
A39 07204 72700004 (06329) ADC$13A: LOAD(BR3,VZ-1)
A3A 07206 74220002 (06330) ADC$13B: SUB(BR2,V1)
      (06331) ,
A3B 07208 76002260 (06332)      JUMP(ADC$10)
A3C 0720A 780A0006 (06333) ADC$14: ADD(BR0,3*01)
A3D 0720C 7A010011 (06334)      MOV8(BR0,BR0)
A3E 0720E 7C090010 (06335)      MOV8(BR0,BR0)
A3F 07210 7E110033 (06336)      SUBL(BV1,3)
A40 07212 80190012 (06337)      MOV8(BR1,BV1)
A41 07214 82003060 (06338)      JUMP(D211A0)
      (06339) ,
A42 07216 84300032 (06340) JOUTPUT PROGRAM
      (06341) ADC$15: SET(RA)
A43 07218 86400064 (06342) ADC$16: LOAD(BW2,YBASE(2))
A44 0721A 8870000E (06343) ADC$16A: LOAD(BW3,VZ-1)
A45 0721C 8A220002 (06344) ADC$16B: SUB(BW2,V1)
      (06345) ,
A46 0721E 8CAB0002 (06346) ADC$17: ADD(BW2,V1,TF)
A47 0721F 8E314601 (06347)      SUBL(BW3,1),JUMPP(ADC$17)
A48 07221 90000020 (06348)      NOP(0)
A49 07223 92000020 (06349)      NOP(0)
A4A 07225 94200030 (06350)      CLEAR(R0)
A4B 07227 96000020 (06351)      NOP(0)
A4C 07229 98000020 (06352)      NOP(0)
A4D 0722B 9A000020 (06353)      NOP(0)
      00000000 (06354)      ADC$A=0C
      0722E 00000000 (06355)      END
      0000009C (06356)      ADC$2=BL-ADC$
      (06357) ,
      (06358) ,
      (06359) ,
      (06360) ,

```

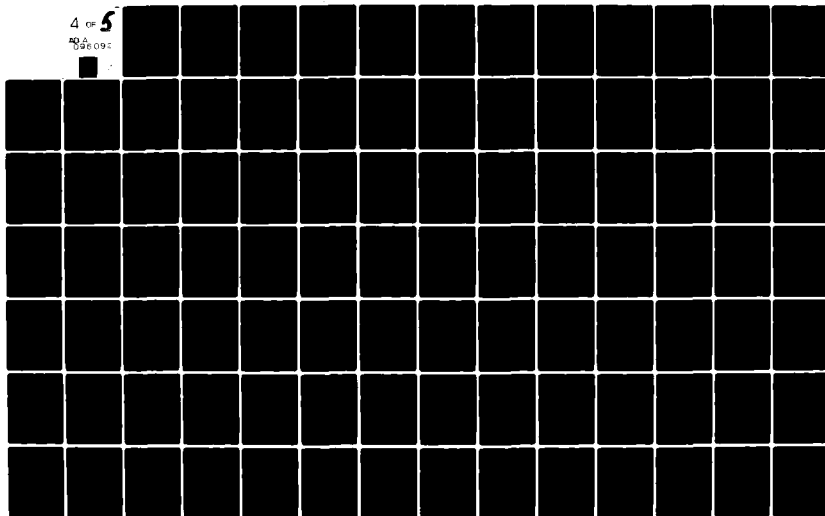
JASSIGN VALUE TO CHAIN

AD-A096 092

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 J J WOLF, K D FIELD, W H RUSSELL
DCA100-79-C-0037
NL

UNCLASSIFIED

4 of 5
NO
096092



280

```

PAGE 142: (08ND)C0CA1G08M16M.N50.2, 29-Dec-88 14:30:40, Ed: WOLF
SDC0RZ - SPECIAL BINDING MODULE FOR D0RZ

0732E C04C0502 (06414)      MOVNRL R4,BCT$BA(R6)
07330 F070727A (06415)      MOVNR R7,ADC$W$*ADC$2
07332 564EFFF0 (06416)      TORNR R4,R7,$FFF0
07334 0440727A (06417)      MOVNRL R4,ADC$W$*ADC$2
07340 (06418)
07341 (06419)
07342 (06420)
07343 (06421)
07344 (06422)
07345 (06423)
07346 (06424)
07347 (06425)
07348 (06426)
07349 (06427)
07350 (06428)
07351 (06429)
07352 (06430)
07353 (06431)
07354 (06432)
07355 (06433)
07356 (06434)
07357 (06435)
07358 (06436)
07359 (06437)
07360 (06438)
07361 (06439)
07362 (06440)
07363 (06441)
07364 (06442)
07365 (06443)
07366 (06444)
07367 (06445)
07368 (06446)
07369 (06447)
07370 (06448)
07371 (06449)
07372 (06450)
07373 (06451)
07374 (06452)
07375 (06453)
07376 (06454)
07377 (06455)
07378 (06456)
07379 (06457)
07380 (06458)
07381 (06459)
07382 (06460)
07383 (06461)
07384 (06462)
07385 (06463)
07386 (06464)
07387 (06465)
07388 (06466)

; (U2-1) BINDING
; (U2-1) = USIZE - 1 = U $S
MOVNR R4,HS*BCT$AD(R6)
MOVNR R4,ADC$W$*ADC$2A*HS

; UI BINDING
; UI=0 SAMPLE INCREMENT
MOVNR R5,BCT$AD(R6)
MOVNR R7,RS
SAP R5,ADC$W$*ADC$2B
SAP R5,ADC$W$*ADC$5
SAP R5,ADC$W$*ADC$7
SAP R5,ADC$W$*ADC$8
SAP R5,ADC$W$*ADC$9

; 4*UI BINDING
; 4*UI=4*(0 SAMPLE INCREMENT)
LLS R5,2
SAP R5,ADC$W$*ADC$11

; M2*UI BINDING
; (M2*UI) = (0 OF LEADING ZEROS) * (U $I)
MOVNR R3,R7
LLS R3,1
SAP R3,ADC$W$*ADC$3
SAP R3,ADC$W$*ADC$4

; V BUFFER BINDING
; VBASE BINDING
MOVNR R6,3*HS(R1)
MOVNR R6,R6,MSK$LBVT
LLS R6,7
EVEN
MOVNRL R4,BCT$BA(R6)
MOVNR R7,ADC$W$*ADC$1
TORNR R4,R7,$FFF0
MOVNRL R4,ADC$W$*ADC$1

MOVNRL R3,BCT$BA(R6)
MOVNR R5,ADC$W$*ADC$10
TORNR R3,R5,$FFF0
MOVNRL R3,ADC$W$*ADC$10

MOVNRL R4,BCT$BA(R6)
MOVNR R7,ADC$W$*ADC$13
TORNR R4,R7,$FFF0
MOVNRL R4,ADC$W$*ADC$13

; RS=U $I
; R7=RS=U $I
; MOVE TO APS MAIN MEMORY
; MOVE TO APS MAIN MEMORY
; MOVE TO APS MAIN MEMORY
; MOVE TO APS MAIN MEMORY
; MOVE TO APS MAIN MEMORY
; MOVE TO APS MAIN MEMORY
; RS=SI .L. 2=4*UI
; MOVE TO APS MAIN MEMORY

; R3=2*M2*UI
; R3=M2*UI
; MOVE TO APS MAIN MEMORY
; MOVE TO APS MAIN MEMORY

; GET V ID
; GET PROPER BYTE
; CONVERT TO BCT$ INDEX
; GET 17-BIT VBASE ADDR
; APS ADDR IN MAIN MEMORY
; MOVE TO APS IN MAIN MEMORY

; GET 17-BIT VBASE ADDR
; APS ADDR IN MAIN MEMORY
; MOVE TO APS IN MAIN MEMORY

; GET 17-BIT VBASE ADDR
; APS ADDR IN MAIN MEMORY
; MOVE TO APS IN MAIN MEMORY

```

PAGE 143: [0800J]OCAL16>08016M.NSO.2, 29-Dec-88 14:38:40, Ed: WOLP
SBOCORZ - SPECIAL BINDING MODULE FOR DCORZ

```

(06467) ; (VZ-1) BINDING
(06468) ; (VZ-1)=VSIZE-1-V NS
0736E F03C0605 MOVNR R3,NS+BC$AD(R6)
07370 E0307269 MOVNR R3,ADC$+W$+ADC$1A+H$
07372 E03072A9 MOVNR R3,ADC$+W$+ADC$1BA+H$
07374 E03072D5 MOVNR R3,ADC$+W$+ADC$13A+H$
(06473) ;
(06474) ; VI BINDING
(06475) ; VI=V SAMPLE INCREMENT
07376 F04C0604 MOVNR R4,BC$AD(R6)
07378 EF40726A SAF R4,ADC$+W$+ADC$1B
0737A EF407274 SAF R4,ADC$+W$+ADC$1B2
0737C EF407292 SAF R4,ADC$+W$+ADC$6
0737E EF4072AA SAF R4,ADC$+W$+ADC$10B
07380 EF4072C4 SAF R4,ADC$+W$+ADC$12
07382 EF4072D6 SAF R4,ADC$+W$+ADC$13B
(06483) ; YBUFFER BINDING
(06484) ; YBASE BINDING
(06485) ;
07384 F0620001 MOVNR R6,MS(R1)
07386 506CFF00 MOVNR R6,R6,MSR$LOVT
07388 3C67 LRS R6,7
07389 8000 EVEN
0738A C04C0502 MOVNRL R4,BC$BA(R6)
0738C F07072E0 MOVNR R7,ADC$+W$+ADC$16
0738E 564PFF0 MOVNR R4,R7,$PFF0
07390 044072E0 MOVNRL R4,ADC$+W$+ADC$16
(06494) ;
(06495) ; (VZ-1) BINDING
(06496) ; VZ-1 = VSIZE - 1 = BS
07392 F07C0605 MOVNR R7,HS+BC$AD(R6)
07394 E07072E0 MOVNR R7,ADC$+W$+ADC$16A+H$
(06499) ;
(06500) ; VI BINDING
(06501) ; VI = SAMPLE INCREMENT
07396 F03C0604 MOVNR R3,BC$AD(R6)
07398 EF3072EC SAF R3,ADC$+W$+ADC$16B
0739A EF3072EE SAF R3,ADC$+W$+ADC$17
(06505) ;
(06506) ;
0739C 0000FF63 JMP AP$BNDRO

```

```

JGET BS = # COLS - 1
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY

JH4 = V SI
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY

JGET Y ID
JGET PROPER BYTE
JCONVERT TO BCT$ INDEX

JGET 17-BIT YBASE ADDR
JAPS ADR IN MAIN MEM

JMODIFIED APS INSTR IN MAIN MEM

JGET BS = # OF COLS - 1
MOVE TO APS MAIN MEMORY

JR3 = Y SI
MOVE TO APS MAIN MEMORY
MOVE TO APS MAIN MEMORY

```

ADDRESS	INSTR	OPERATION	COMMENT
06500	WPIFFS	MODULR TO PROCESS THE WPIFF(ISA,ISB,FLID) FCB	
06509	FIELD 9/79		
06510			
06511			
06512			
06513			
06514			
06515			
06516			
06517			
06518			
06519			
06520			
06521			
06522			
06523			
06524			
06525			
06526			
06527			
06528			
06529			
06530			
06531			
06532			
06533			
0730E 704200FF	WPIFFS	MODULR TO PROCESS THE WPIFF(ISA,ISB,FLID) FCB	
07310 70520001			
07312 70220002			
07314 22000502			
07316 011073AA			
07318 0E70			
07319 0000			
0731A E20A0502			
0731C 0010192C			
0731E 0E70			
0731F 0000			

PAGE 145: (0000)DCAL6>BBNICH.MSD.2, 29-Dec-88 14:38:48, 24: WOLF
 MPCSC -- G-FLAG SET/CLEAR

```

(06551) * MPCSC -- G-FLAG SET/CLEAR
(06552) JJW, 18 OCT 79
(06553) *
(06554) * G-FLAG NUMBER IN 1(01)
(06555) * 0/1 = CLEAR/SET IN 2(01)
(06556) *
(06557) MPCSC: MOVNR R2,1(01)
(06558) INCX R2,4
(06559) MOVNR R3,2(01)
(06560) LLS R3,5
(06561) TORNR R2,R3,R20
(06562) MOVNR R2,SYSPFGS
(06563) RETURN
(06564) EVEN

07300 70220001
07302 2624
07303 70320002
07305 3A35
07306 56360020
07308 E021FFCE
0730A 0270
0730B 0000

JGET G-FLAG NUMBER
JADD OFFSET FOR SYSPFGS
JGET CLR/SET BIT
JSHIFT TO PROPER POSITION
JAND OFF AND PUT IN R2
JDD THE MOVE THAT SETS/CLRS IT
  
```

PAGE 146: (0000)C0C16>00M16N.H50.2, 29-Dec-88 14:38:48, Ed: WOLF
DEBUGGING "BREAKPOINT" FACILITY

(06565) "DEBUGGING "BREAKPOINT" FACILITY
(06566) ;
(06567) ; TO WALT CPU EXECUTION AND SAVE REGISTERS ON THE STACK,
(06568) ; PATCH: "CALL RI, BKPT" (AS ASSEMBLED BELOW)
(06569) ; INTO THE DESIRED BREAKPOINT LOCATION (USING MPL00K).
(06570) ;
0730C 0610730E (06571) CALL RI, BKPT
0730E 0000730E (06572) BKPT: JMP BKPT

ROIMINIAO 'SAIINCXKZ JO JOJ. (C6590)

(06573) -T
(06574) -
(06575) -
(06576) -
(06577) -
(06578) -
(06579) -

0000730 (06576)
TUE\$CUB = IL

0360 (0659) 093

ADCS:	07262	(00221)	(06245)	(06253)	(06356)	(06405)	(06406)	(06415)	(06417)	(06422)	(06420)
		(06429)	(06430)	(06431)	(06432)	(06437)	(06443)	(06444)	(06453)	(06455)	(06450)
		(06460)	(06463)	(06465)	(06470)	(06471)	(06472)	(06477)	(06470)	(06479)	(06400)
		(06481)	(06482)	(06491)	(06493)	(06490)	(06503)	(06504)			
ADCS1:	00002	(06257)	(06453)	(06455)							
ADCS10:	00022	(06300)	(06316)	(06332)	(06450)	(06460)					
ADCS10A:	00023	(06301)	(06471)								
ADCS10B:	00024	(06302)	(06400)								
ADCS11:	00026	(06305)	(06437)								
ADCS12:	00031	(06320)	(06401)								
ADCS13:	00030	(06320)	(06463)	(06465)							
ADCS13A:	00039	(06329)	(06472)								
ADCS13B:	0003A	(06330)	(06402)								
ADCS14:	0003C	(06333)									
ADCS15:	00042	(06254)	(06341)								
ADCS16:	00043	(06342)	(06491)	(06493)							
ADCS16A:	00044	(06343)	(06490)								
ADCS16B:	00045	(06344)	(06503)								
ADCS17:	00046	(06346)	(06347)	(06504)							
ADCS1A:	00003	(06250)	(06470)								
ADCS1B:	00004	(06259)	(06477)								
ADCS1B2:	00009	(06267)	(06470)								
ADCS2:	0000C	(06264)	(06200)	(06272)	(06415)	(06417)					
ADCS2A:	0000D	(06273)	(06472)								
ADCS2B:	0000E	(06274)	(06420)								
ADCS2C:	00010	(06276)	(06406)								
ADCS3:	00012	(06200)	(06443)								
ADCS4:	00014	(06202)	(06444)								
ADCS5:	00017	(06207)	(06479)								
ADCS6:	00010	(06200)	(06479)								
ADCS7:	00010	(06294)	(06430)								
ADCS8:	0001F	(06296)	(06431)								
ADCS9:	00021	(06290)	(06432)								
ADCSA:	00000	(06240)	(06354)								
ADCSB:	00005	(06245)	(06261)	(06405)							
ADCSZ:	0009C	(06247)	(06356)								
ADPA:	065FE	(00161)	(02007)	(02030)							
ADPA\$A:	0660A	(02001)	(02027)								
ADPA\$B:	0661A	(02797)	(02029)								
ADPA\$Z:	00020	(02000)	(02030)								
ADPAI:	065FE	(02000)									
ADPAO:	0000A	(02009)	(02022)								
ADPE:	065D2	(00160)	(02761)								
ADPE\$A:	00000	(02750)	(02762)	(02700)							
ADPE\$Z:	00012	(02759)	(02700)								
ADP\$ORC:	00000	(00169)	(00124)	(00129)	(00134)	(00139)	(00144)	(00149)	(00154)	(00159)	(00164)
		(00169)	(00174)	(00179)	(00184)	(00189)	(00194)	(00199)	(00204)	(00209)	(00214)
		(00219)	(00203)	(02106)	(02107)	(04704)	(04707)	(04708)	(05369)	(05372)	(05373)
		(06374)	(06377)	(06378)							
AP\$A\$S:	00245	(00023)									
AP\$ANDR:	0007A	(00024)	(03595)								
AP\$ANDR0:	00F63	(00025)	(02203)	(05075)	(05509)	(06507)					
AP\$ANDR1:	00F20	(00026)									

PAGE 149:

AP 505 JV

APCSW41:	00041 (04057) (04957) (04960) (04962)
APCSW51:	00032 (04053) (04966) (04968) (04971)
APCSW7:	00042 (04787) (04821) (04824)
APCSZ1:	00008 (04493) (04765)
APCU1:	00072 (00190) (04032)
APCU811:	00091 (04350) (04363)
APCU812:	00009 (04315) (04346)
APCU831:	00004 (04323) (04337)
APCUS0UM:	00095 (04333) (04342) (04359) (04373)
APCUS5A1:	00000 (04029) (04033) (04419)
APCUS5Z1:	000A0 (04030) (04419)
APCU\$TOP:	00015 (04083) (04399)
APSSMD1:	00004 (00030)
APSLA:	1FFC0 (00039) (01158) (01670)
APSLDC:	1FFC0 (00040)
APSMEN:	1FFC0 (00041) (02712) (04697) (05055)
APSR:	1FFC0 (00042)
BCT\$AD:	00604 (00044) (02229) (05404) (05421) (05426) (05465) (05482) (05499)
	(06421) (06426) (06476) (06502)
BCT\$AT:	00606 (00045) (03500) (02241) (02251) (02256) (02266) (02271) (04020) (04032) (04044)
BCT\$BN:	00502 (00046) (02223) (04063) (04071) (04080) (04089) (04090) (04097) (04923) (04929) (04930)
	(04947) (04956) (04965) (04981) (04987) (05000) (05006) (05019) (05025) (05034) (05039)
	(05043) (05052) (05061) (05070) (05390) (05415) (05437) (05454) (05476) (05493)
	(06414) (06452) (06457) (06462) (06490)
BITSGR:	00000 (00047)
BITSRC:	00000 (00048)
BIT\$N2:	00003 (00049) (04023)
BEPT:	07302 (06571) (06572)
CF40B\$:	01AD6 (00051) (02199) (04800) (05305) (06390)
CLR\$G61:	00792 (00052) (03591) (03700)
COS\$:	03190 (00053) (02401) (02404) (02412)
COS\$B:	00024 (02390) (00127) (00132) (00137) (00152) (00162) (00167) (00172) (00177)
CSPU\$R03:	021FC (00054) (00127) (00132) (00212) (00217)
	(00197) (00207) (00212) (00241) (02470)
CTMCL:	09000 (00239) (00240) (00241) (02402)
CTMC2:	00010 (00240) (00242) (02402)
CTMC3:	09000 (00241) (02406)
CTHDG:	099F0 (00249) (00250) (03492) (03498) (03504)
CTHGI:	09970 (00240) (00249) (03486)
CTHFI:	09830 (00242) (00243) (01596)
CTHF2:	09000 (00243) (00244)
CTHF3:	090F0 (00244) (00245)
CTHF4:	09910 (00245) (00246)
CTHF5:	09930 (00246) (00247)
CTHF6:	09950 (00247) (00248)
CTHU:	09970 (00250) (00251) (04629) (04643) (04655)
D211A5:	00020 (06393) (06312)
D211A6:	0002C (06395) (06313)
D211A7:	00020 (06297) (06314)
D211A8:	00030 (06206) (06319) (06321) (06330)
DCEMD:	00007 (06061) (06114) (06120) (06139) (06147) (06166) (06174) (06193) (06201) (06211) (06212)
DCLP:	00026 (06073) (06102)

BCL1:	00030 (06001) (06129)	
BCL2:	00054 (06009) (06156)	
BCL3:	00060 (06097) (06183)	
DCOR8:	07150 (00220) (06051)	
DCOR8Z:	00000 (06040) (06053)	(06126) (06153) (06180) (06287) (06213) (06214)
DCOR8Z:	00005 (06049) (06214)	
DCOR9:	00009 (06067) (06099)	
DCOR1:	00010 (06075)	
DCOR2:	00017 (06083)	
DCOR3:	0001E (06091)	
DEAL3:	070FA (00216)	(05937) (05942) (06014)
DEAL3A:	07132 (05940)	(06000)
DEAL3B:	0713A (05936)	(06012)
DEAL3D:	00000 (05943) (05967)	
DEAL3E:	0000C (05937) (05969)	
DEAL3Z:	00056 (05939) (06014)	
DEALB:	07020 (00215) (05913)	
DEALB3A:	00000 (05910) (05914)	(05916) (05920)
DEALB3Z:	00004 (05911) (05920)	
DMS:	00794 (00056) (01530)	(02714) (02715) (05669) (05670) (05057) (05058)
DUMPS:	06660 (00166) (02090)	(02919)
DUMPSA:	06672 (02093) (02917)	
DUMPSB:	0667A (02089) (02910)	
DUMPSZ:	0001E (02092) (02919)	
DUMPS:	00000 (02099)	
DUMPSD:	00004 (02900) (02907)	
DUMPU:	06620 (00165) (02043)	
DUMPU3A:	00000 (02040) (02044)	(02000)
DUMPU3Z:	0001C (02041) (02000)	
DVLC1:	09100 (00251) (00252)	(00253) (02479)
DVLC2:	09110 (00252) (00254)	(02403)
DVLC3:	09100 (00253) (02407)	
DVLCG:	090F0 (00261) (00262)	(03493) (03499) (03505)
DVLC:	09130 (00260) (00261)	(03407)
DVLC:	09130 (00254) (00255)	(01597)
DVLC2:	09100 (00255) (00256)	
DVLC3:	09100 (00256) (00257)	
DVLC4:	09110 (00257) (00250)	
DVLC5:	09030 (00250) (00259)	
DVLC6:	09050 (00259) (00260)	
DVLC:	090F0 (00262) (00263)	(04665)
DVLCF:	09C00 (00263)	
EAROR:	011FA (00050)	
EMID:	0000A (02760)	(02772)
FI:	00000 (03794)	(03797)
FI1A:	03110 (00060)	
FI1AL:	069F0 (00101)	(00106) (03700) (03069)
FI1AL3:	0001C (03706)	(03023)
FI1ALCA:	0614C (03777)	(03050)
FI1ALCE:	06160 (03773)	(03066)
FI1ALBZ:	00000 (03776) (03069)	
F2:	00012 (03002) (03007)	
F4:	0002D (03034) (03045)	(03051)

FDY\$1:	007E0 (00061) (00220) (00231)	
FDP\$1:	06900 (03574) (03584)	
FOP\$SM:	06906 (00102) (03579)	
FOP\$SM:	06902 (00187) (03573)	
FOS\$1:	03040 (00062)	
FOSL\$1:	06920 (00100) (00185) (03612)	
FOSL\$1:	00012 (03625) (03642)	
FOSL\$2:	00019 (03630) (03653) (03694)	
FOSL\$3:	0003A (03649) (03701) (03742)	
FOSL\$4:	00059 (03697) (03744)	
FOSL\$5:	00000 (03609) (03617) (03754) (03756)	
FOSL\$5:	00064 (03610) (03756) (03757)	
FYS\$CS:	0079A (00063)	
FLC\$CL:	00000 (00064) (03579)	
FLC\$CL:	00004 (00065) (03573) (03579)	
FLC\$CL:	00005 (00066)	
FLC\$C2:	00006 (00067)	
FLC\$C3:	00007 (00068)	
FLC\$R1:	00011 (00069)	
FLC\$SE:	00020 (00070) (03573)	
FLC\$SE:	00008 (04487) (04520)	
FLC\$SE:	06764 (00175) (03221)	
FLC\$SE:	00042 (03267) (03283)	
FLC\$SE:	00052 (03250) (03273) (03326)	
FLC\$SE:	00055 (03352) (03358)	
FLC\$SE:	00058 (03360) (03362)	
FLC\$SE:	0005F (03357) (03364)	
FLC\$SE:	00061 (03227) (03231)	
FLC\$SE:	00065 (03370) (03396)	
FLC\$SE:	00077 (03381) (03403)	
FLC\$SE:	00079 (03386) (03406)	
FLC\$SE:	00078 (03391) (03409)	
FLC\$SE:	00080 (03210) (03223)	
FLC\$SE:	00084 (03219) (03421)	
FLC\$SE:	00074 (00176) (03445)	
FLC\$SE:	00080 (03473) (03480)	
FLC\$SE:	0000C (03475) (03476)	
FLC\$SE:	060EC (03440) (03556)	
FLC\$SE:	060FA (03444) (03560)	
FLC\$SE:	00032 (03457) (03534)	
FLC\$SE:	00020 (03459) (03514)	
FLC\$SE:	00003 (03445) (03462)	
FLC\$SE:	0000F (03447) (03562)	
FLC\$SE:	0103C (00072)	
FLC\$SE:	06E42 (00196) (05140) (05217)	
FLC\$SE:	06F7C (05143) (05211)	
FLC\$SE:	06E04 (05139) (05215)	
FLC\$SE:	00017 (05146) (05193)	
FLC\$SE:	00002 (05140) (05151)	
FLC\$SE:	00056 (05142) (05217)	
FLC\$SE:	06F32 (00195) (05116)	
FLC\$SE:	00000 (05113) (05117) (05119) (05123)	
FLC\$SE:	00004 (05114) (05123)	

```

MS:
00001 (00074) (01153) (01644) (01669) (02191) (02196) (02206) (02210) (02213) (02216)
(02219) (02229) (02231) (02233) (02234) (02235) (02237) (02247) (02262) (02710)
(03500) (03500) (04693) (04700) (04731) (04792) (04800) (04812) (04816) (04816)
(04820) (04839) (04917) (04975) (04994) (05013) (05377) (05382) (05394) (05411)
(05421) (05422) (05433) (05438) (05450) (05461) (05462) (05472) (05489) (05489)
(05499) (05500) (05501) (05853) (06382) (06387) (06401) (06405) (06406) (06410)
(06421) (06422) (06440) (06469) (06470) (06471) (06472) (06486) (06497) (06498)
(06535)
06700 (00171) (03103) (03165)
00002 (03109)
00006 (03115)
00010 (03130) (03131)
00012 (03132)
00016 (03147)
0001A (03152) (03153)
0001C (03154)
06740 (03089) (03160)
06746 (03085) (03164)
00015 (03105) (03144)
0005A (03080) (03165)
06680 (00170) (02971)
00039 (03026) (03027)
00020 (02900) (03060)
00000 (02960) (02976) (03040)
00040 (02969) (03060)
00030 (02010)
073AA (06539) (06543)
00000 (00360)
06202 (01600) (01669) (01675)
0658A (02651) (02725)
065C2 (02647) (02726)
06584 (02667) (02680) (02710)
00003 (02640) (02664)
0006E (02650) (02727)
06562 (00156) (02640) (02660) (02727)
00024 (02661) (02700)
0651A (00155) (02592) (02620)
00000 (02589) (02593)
00020 (02590) (02620)
00502 (00076) (02215)
0630A (02017) (02152)
0005F (01417) (01423)
0004E (01451) (01509)
00070 (01509) (01635)
00066 (01591) (01619)
00053 (01382) (01400)
0004P (00434) (00440)
0107E (00070)
00010 (00395) (00445)
00000 (00084) (00507)
(00574) (00582) (00583) (00584) (00587) (00592) (00593) (00594) (00597) (00598)
(00600) (00601) (00602) (00603) (00604) (00605) (00606) (00607) (00608) (00609)
(00610) (00611) (00612) (00613) (00614) (00615) (00616) (00617) (00618) (00619)
(00620) (00621) (00622) (00623) (00624) (00625) (00626) (00627) (00628) (00629)
(00630) (00631) (00632) (00633) (00634) (00635) (00636) (00637) (00638) (00639)
(00640) (00641) (00642) (00643) (00644) (00645) (00646) (00647) (00648) (00649)
(00650) (00651) (00652) (00653) (00654) (00655) (00656) (00657) (00658) (00659)
(00660) (00661) (00662) (00663) (00664) (00665) (00666) (00667) (00668) (00669)
(00670) (00671) (00672) (00673) (00674) (00675) (00676) (00677) (00678) (00679)
(00680) (00681) (00682) (00683) (00684) (00685) (00686) (00687) (00688) (00689)
(00690) (00691) (00692) (00693) (00694) (00695) (00696) (00697) (00698) (00699)
(00700) (00701) (00702) (00703) (00704) (00705) (00706) (00707) (00708) (00709)
(00710) (00711) (00712) (00713) (00714) (00715) (00716) (00717) (00718) (00719)
(00720) (00721) (00722) (00723) (00724) (00725) (00726) (00727) (00728) (00729)
(00730) (00731) (00732) (00733) (00734) (00735) (00736) (00737) (00738) (00739)
(00740) (00741) (00742) (00743) (00744) (00745) (00746) (00747) (00748) (00749)
(00750) (00751) (00752) (00753) (00754) (00755) (00756) (00757) (00758) (00759)
(00760) (00761) (00762) (00763) (00764) (00765) (00766) (00767) (00768) (00769)
(00770) (00771) (00772) (00773) (00774) (00775) (00776) (00777) (00778) (00779)
(00780) (00781) (00782) (00783) (00784) (00785) (00786) (00787) (00788) (00789)
(00790) (00791) (00792) (00793) (00794) (00795) (00796) (00797) (00798) (00799)
(00800) (00801) (00802) (00803) (00804) (00805) (00806) (00807) (00808) (00809)
(00810) (00811) (00812) (00813) (00814) (00815) (00816) (00817) (00818) (00819)
(00820) (00821) (00822) (00823) (00824) (00825) (00826) (00827) (00828) (00829)
(00830) (00831) (00832) (00833) (00834) (00835) (00836) (00837) (00838) (00839)
(00840) (00841) (00842) (00843) (00844) (00845) (00846) (00847) (00848) (00849)
(00850) (00851) (00852) (00853) (00854) (00855) (00856) (00857) (00858) (00859)
(00860) (00861) (00862) (00863) (00864) (00865) (00866) (00867) (00868) (00869)
(00870) (00871) (00872) (00873) (00874) (00875) (00876) (00877) (00878) (00879)
(00880) (00881) (00882) (00883) (00884) (00885) (00886) (00887) (00888) (00889)
(00890) (00891) (00892) (00893) (00894) (00895) (00896) (00897) (00898) (00899)
(00900) (00901) (00902) (00903) (00904) (00905) (00906) (00907) (00908) (00909)
(00910) (00911) (00912) (00913) (00914) (00915) (00916) (00917) (00918) (00919)
(00920) (00921) (00922) (00923) (00924) (00925) (00926) (00927) (00928) (00929)
(00930) (00931) (00932) (00933) (00934) (00935) (00936) (00937) (00938) (00939)
(00940) (00941) (00942) (00943) (00944) (00945) (00946) (00947) (00948) (00949)
(00950) (00951) (00952) (00953) (00954) (00955) (00956) (00957) (00958) (00959)
(00960) (00961) (00962) (00963) (00964) (00965) (00966) (00967) (00968) (00969)
(00970) (00971) (00972) (00973) (00974) (00975) (00976) (00977) (00978) (00979)
(00980) (00981) (00982) (00983) (00984) (00985) (00986) (00987) (00988) (00989)
(00990) (00991) (00992) (00993) (00994) (00995) (00996) (00997) (00998) (00999)

```


NULP3A:	06100 (01441) (01650)	
NULP3AP:	06080 (00141) (01430)	(01440) (01650)
NULP3APB:	06016 (00140) (01230)	
NULP3B:	06162 (01437) (01656)	
NULP3C:	00002 (01430) (01450)	(01454)
NULP3SA:	00000 (01234) (01241)	(01430)
NULP3SZ:	00065 (01235) (01430)	
NULP3Z:	0010A (01440) (01650)	
NULP3Z:	00039 (01342) (01300)	
NULP3Z:	00051 (01326) (01307)	
NULP3Z:	00040 (01367) (01301)	
NULP3Z:	00017 (01277) (01306)	
NULP3Z:	0004P (01256) (01305)	(01305)
NULP3Z:	00010 (01250) (01267)	
NULP3Z:	0004R (01561) (01567)	
NULP3Z:	0004C (01549) (01569)	
NULP3Z:	00032 (01532) (01537)	
NULP3Z:	0002A (01499) (01527)	
NULP3Z:	001F2 (00142) (01661)	
NULP3Z:	00015 (01475) (01403)	(01405)
NULP3Z:	00012 (01476) (01400)	
NULP3Z:	00020 (01509)	
NULP3Z:	06170 (01662) (01660)	
NULP3Z:	06310 (02039) (02042)	
NULP3Z:	00002 (06229) (06261)	(06276) (06280) (06282)
NULP3Z:	00010 (00006) (00656)	(00671) (00934) (00943) (02601) (02605) (04056) (04062) (05750)
NULP3Z:	00010 (05754)	
PIT3S:	052A4 (00136) (01023)	(01040) (01179)
PIT3S:	052E2 (01101) (01116)	(01153)
PIT3S:	06007 (01026) (01174)	
PIT3S:	06000 (01022) (01170)	
PIT3S:	00026 (01043) (01143)	
PIT3S:	00001 (01023) (01046)	
PIT3S:	00070 (01025) (01179)	
PIT3S:	070C0 (05000) (05072)	
PIT3S:	070D6 (05796) (05073)	
PIT3S:	07098 (05015) (05020)	(05853)
PIT3S:	00003 (05797) (05812)	
PIT3S:	00070 (05799) (05074)	
PIT3S:	07076 (00211) (05797)	(05800) (05874)
PIT3S:	00020 (05009) (05051)	
PIT3S:	07022 (00210) (05741)	
PIT3S:	00000 (05730) (05742)	(05777)
PIT3S:	00020 (05739) (05777)	
PIT3S:	052F4 (00135) (00040)	
PIT3S:	00007 (00052) (00060)	
PIT3S:	00010 (00050) (00063)	
PIT3S:	00015 (00050) (00070)	(00082)
PIT3S:	00010 (00054) (00065)	(00075)
PIT3S:	00010 (00056) (00080)	
PIT3S:	0001P (00067) (00086)	
PIT3S:	00025 (00080) (00092)	
PIT3S:	00027 (00091) (00090)	

PRISON:

AGE 157: LHMWJ<CAL>NHP16M-AS0.2, 29-Dec-88 14:J0140, Ed: WOLF
TOP OF EXCLUSIVE DEFINITION

TMSJC:	00JKN	(05271)	(05350)
TMSJ1:	00P02	(05201)	
TMSJ2:	00M1J	(05274)	(05310)
TMSJHAI:	00M1E	(05339)	(05477)
TMSJHAI:	00W2H	(05341)	(05483)
TMSJHAI:	00D21	(05343)	(05484)
TMSJHAI:	00M4C	(05302)	(05494)
TMSJHAI:	00M00	(05303)	(05500)
TMSJHAI:	00M1F	(05310)	(05501)
TMSJHAI:	00M0B	(05304)	(05505)
TMSJHAI:	00M4F	(05306)	(05506)
TMSJHAI:	00M02	(05204)	(05416)
TMSJHAI:	00M03	(05205)	(05422)
TMSJHAI:	00M15	(05322)	(05423)
TMSJHAI:	00M04	(05206)	(05427)
TMSJHAI:	00M05	(05200)	(05428)
TMSJHAI:	00M19	(05330)	(05430)
TMSJHAI:	00M10	(05332)	(05444)
TMSJHAI:	00M1C	(05334)	(05445)
TMSJHAI:	00M07	(05293)	(05455)
TMSJHAI:	00M0H	(05294)	(05461)
TMSJHAI:	00M1A	(05331)	(05462)
TMSJHAI:	00M09	(05295)	(05466)
TMSJHAI:	00M0A	(05297)	(05467)
TMSJHAI:	00M14	(05321)	(05399)
TMSJHAI:	00M16	(05323)	(05405)
TMSJHAI:	00M17	(05325)	(05406)
TMSJHAI:	00M04	(05270)	(05353)
TMSJHAI:	00E9A	(04200)	(05243)
TMSJHAI:	00M0M	(05240)	(05244)
TMSJHAI:	00M04	(05241)	(05249)
TMSJHAI:	021PF	(00097)	
TMSJHAI:	073C0	(00113)	(06576)
TMSJHAI:	00200	(00090)	(00111)
TMSJHAI:	00304	(02001)	(02065)
TMSJHAI:	00M0P	(06222)	(06272)
TMSJHAI:	00J15	(02002)	
TMSJHAI:	00M02	(06230)	(06274)
TMSJHAI:	00M02	(03702)	(03795)
TMSJHAI:	00M0C	(06220)	(06273)
TMSJHAI:	05D06	(00126)	(00409)
TMSJHAI:	00M10	(00495)	(00562)
TMSJHAI:	05E22	(00485)	(00600)
TMSJHAI:	05E2A	(00481)	(00605)
TMSJHAI:	00M0H	(00404)	(00606)
TMSJHAI:	01C5A	(01134)	
TMSJHAI:	00J06	(02096)	(02047)
TMSJHAI:	00J32	(05223)	(06257)
TMSJHAI:	00J07	(02017)	
TMSJHAI:	00M02	(05231)	(06254)
TMSJHAI:	00J14	(03710)	(04721)
TMSJHAI:	00M20	(01751)	(04757)
TMSJHAI:	05E0C	(00609)	(05765)

PAGE 159: [0000]UCAL6>00016M.WSN.2, 29-Dec-88 14:30:40, Ed: WOLF
TOP OF EXECUTIVE DEFINITION

YOUS:	06303 (01999)	(06349) (06346)	(03047) (03048) (03049) (03050)
Y11:	00002 (06232)	(03036) (03037)	(06228) (06243)
Y01:	00002 (03783)	(02049) (02091)	(06312) (06313) (06314) (06319) (06324)
Y2:	0000F (02043)	(02049) (02091)	(06312) (06313) (06314) (06319) (06324)
ZFR0:	0070A (00107)	(02201) (04002)	(06312) (06313) (06314) (06319) (06324)
	0070A (06325)	(06326) (06392)	

LINES WITH ERRORS: 0 (MAP VERSION 000101.10) E- 0

(00002) * CSPI PATCHES TO MAP-300 SWAPII (REL 3.5) EXEC
 (00003) *
 (00004) * THIS PATCH FILE ASSUMES MAP HAS BEEN LOADED WITH
 (00005) * SWAPII EXEC, RELEASE 3.5.
 (00006) *
 (00007) * THIS PATCH ASSUMES REV. 10 (OR LATER) MICROCODE.
 (00008) *
 (00009) * DEFINE 'SAF' (KEY ADDRESS FIELD) AND 'LAP' (LOAD
 (00010) * ADDRESS FIELD) INSTRUCTIONS TO ASSEMBLER.
 (00011) * THESE INSTRUCTIONS ARE NEW CSPI INSTRUCTIONS
 (00012) * AVAILABLE WITH 'REV 10' CSPI MICROCODE.
 (00013) * THEY SET/LOAD THE LOW ORDER 17 BITS OF THE
 (00014) * REFERENCED FULL-WORD FROM/INTO THE INDICATED
 (00015) * REGISTER.
 (00016) *
 (00017) * OPADD SAF,(I -LS. 14) + (29 -LS. 0) + \$F
 (00018) * OPADD LAP,(I -LS. 14) + (29 -LS. 0) + \$F
 (00019) *
 (00020) * DEFINE 00 TO BE CURRENT TOP OF PATCH AREA
 (00021) * (UPDATE AS PATCH SPACE IS USED).
 (00022) * (SPACE FROM \$1C78-\$1C97 IS USED BY REL. 3.5)
 (00023) *
 (00024) * 00 = \$1C98
 (00025) *
 (00026) * DEFINE REL. 3.5 SYMBOLS
 (00027) *
 (00028) * AP\$WORD0 = \$0F63
 (00029) * BCT\$0A = \$0502
 (00030) * OCT\$AD = \$0604
 (00031) * AP\$DEL = \$0240
 (00032) * AP\$CSL = \$024A
 (00033) * SCD\$04H = \$0E30
 (00034) * W\$ = 1
 (00035) *
 00001C98
 00000F63
 00000502
 00000604
 00000240
 0000024A
 00000E30
 00000001

PAGE 3: (00NDJ)C0A10>00N16P.N50.1, 10-NOV-80 19:15:52, EDI KFIELD
 PROBLEM REPORT 01201 NEWSLETTER 023, PAGE 55

(00036) *PROBLEM REPORT 01201 NEWSLETTER 023, PAGE 55
 (00037) *
 (00038) *
 (00039) * PATCH PREBINDING ROUTINES TO WORK WITH PREBINDING
 (00040) * BUFFERS ABOVE \$7FFF (16K).
 (00041) * (NOTE: THIS PATCH USES PATCH SPACE.)
 (00042) *

00000260 (00043) BL = \$0E60
 00E60 023A0502 (00044) LAF R3, DCT\$0A(R5)

0000076C (00045) *
 0076C 02440507 (00046) BL = \$0F6C
 (00047) LAF R4, DCT\$0A(R2)

0000070C (00048) *
 007AC 07540502 (00049) BL = \$0F8C
 (00050) SAF R5, DCT\$0A(R2)

00000701 (00051) *
 007A1 00001C90 (00052) BL = \$0FB1
 (00053) JMP P12010
 (00054) P12010:

00000E50 (00055) *
 00E50 00001C00 (00056) BL = \$0E50
 (00057) JMP P1201C

00001C90 (00058) *
 00001C90 (00059) BL = 00
 (00060) P12010:

01C90 C060024A (00061) MOVWRL R6, AP\$CSL
 01C9A 9160FF7F (00062) ANDIR R6, \$FFFF
 01C9C 9A70FF7F (00063) ANDIR R7, \$FFFF
 01C9E 00000703 (00064) JMP P1201A

00000E50 (00065) *
 01CA0 9A30FF7F (00066) P1201C:
 (00067) ANDIR R3, \$FFFF
 01CA2 00000E30 (00068) JMP SCD\$04N
 (00069) *

00001C14 (00070) *
 00001C14 (00071) *
 (00072) *
 R0 = BL

PAGE 4: (RNDJ)DCAL16>BMMISD-MSD.1, 18-NOV-88 19115152, Ed: KFIELD
 PROBLEM REPORT #1203, NEWSLETTER #23, PAGE 59

(00073) *PROBLEM REPORT #1203, NEWSLETTER #23, PAGE 59
 (00074) *
 (00075) * PATCH STANDARD BINDING Routines TO CORRECTLY BIND
 (00076) * ARRAY FUNCTIONS RESIDING ABOVE 16K ON BUS 1.
 (00077) * (NOTE: THIS PATCH USES PATCH SPACE.)
 (00078) *

0000F1E (00079) BL = \$0F1E
 0001E 00001CA4 (00080) JMP P1203B
 (00081) P1203A:
 (00082) *

00001CA4 (00083) BL = 10
 (00084) P1203B:
 (00085) ANDIR R6,\$FFFF

01CA4 916FFFFF (00086) INCR R6,HS
 01CA6 2661 (00087) INCR R7,HS
 01CA7 2671 (00088) JMP P1203A
 01CA8 0000FF20 (00089) *

(00090) *
 00001CA4 (00091) BL = BL
 (00092) *

PAGE 5: (RNDJ)DCAL16>BMMISD-MSD.1, 18-NOV-88 19115152, Ed: KFIELD
 PROBLEM REPORT #3201, NEWSLETTER #23, PAGE 72

(00093) *PROBLEM REPORT #3201, NEWSLETTER #23, PAGE 72
 (00094) *
 (00095) * PATCH FFT SPECIAL BINDING Routines TO ALLOW FOR
 (00096) * CORRECT PRE-BINDING OF FFTM.
 (00097) *

0005534 (00098) BL = \$5534
 05514 0000FF63 (00099) JMP AP\$BNDDB
 (00100) *

(00101) *

PAGE 6: (000030CA16)00016P.MSD.1, 18-NOV-88 19:15:52, Ed: KFIELD
 PROBLEM REPORT 03202) NEWSLETTER 024, PAGE 71

(00102) *PROBLEM REPORT 03202) NEWSLETTER 024, PAGE 71
 (00103) *
 (00104) * PATCH TO MAKE PPTM PREDIMMABLE WITH PREDIMMING BUFFER
 (00105) * ABOVE \$7FFF.
 (00106) * (NOTE: THIS PATCH USES PATCH SPACE.)
 (00107) *
 (00108) *
 (00109) *
 (00110) *

000054A6 (00111) BL = \$54A6
 054A6 00001CA2 (00112) JMP P32020
 (00113) P32020
 (00114) *

00001CA2 (00115) BL = 00
 01CAA F0300240 (00116) P32020: MOVNR R3,AP0SL
 01CAC 9A30FF0F (00117) ANDIR R3,\$0FFF
 01CAE 000054A0 (00118) JMP P3202A
 (00119) *
 (00120) *

00001C00 (00121) 00 = 0L
 (00122) *

PAGE 7: (000030CA16)00016P.MSD.1, 18-NOV-88 19:15:52, Ed: KFIELD
 PROBLEM REPORT 03202) NEWSLETTER 024, PAGE 71

(00123) *PROBLEM REPORT 03202) NEWSLETTER 024, PAGE 71
 (00124) *
 (00125) * PATCH TO EXEC UTILITY "VALBUFS" TO ALLOW FOR
 (00126) * CORRECT VALIDATION OF BUFFERS WITH NEGATIVE
 (00127) * SAMPLE INCREMENTS (SI'S).
 (00128) * KFIELD 7/88
 (00129) *

00001C5E (00130) BL = \$1C5E
 01C5E F0340604 (00131) MOVNR R3,BCT\$AD(R2) LOAD VECTOR SI
 (00132) *

00001C62 (00133) BL = \$1C62
 01C62 C2340605 (00134) MULNR R3,BCT\$AD+H\$(R2) MULTI BY SIZE-1
 (00135) *
 (00136) *

01C64 END

PAGE 8: (08W0)<0CA16>08W16P.W50.1, 16-NOV-88 19:15:52, Edi RFIELD
 PROBLEM REPORT 08W16P NEWSLETTER 8-- PAGE --

AP5MOR04: 06F63 (00020) (00099)
 AP5SL1: 06740 (00031) (00117)
 AP5SL1: 06741 (00032) (00061)
 06742 (00030) (00131) (00134)
 06743 (00029) (00044) (00047) (00050)
 06744 (00034) (00086) (00087) (00134)
 06745 (00034) (00064)
 06746 (00054) (00053) (00060)
 06747 (00057) (00066)
 06748 (00061) (00080)
 06749 (00080) (00084)
 06750 (00113) (00119)
 06751 (00112) (00116)
 06752 (00033) (00060)
 06753 (00033) (00060)

LINES WITH ERRORS: 8 (MAP VERSION 00001.10) 8- 8

PAGE 1: (00000) <00000> 00000, 10-Nov-00 1915102, Ed: KFIELD
 (00001) <00001> 00001, 10-Nov-00 1915102, Ed: KFIELD

09850	F4127640 (00055)	DATA --90681341
09852	F2168DC0 (00056)	DATA --89571356
09854	F11258C0 (00057)	DATA --80337240
09856	F5162C0 (00058)	DATA --86967121
09858	E05PC1C0 (00059)	DATA --85448478
0985A	E0392D40 (00060)	DATA --83768241
0985C	E0093940 (00061)	DATA --81912912
0985E	E6306440 (00062)	DATA --79868747
09860	E35821C0 (00063)	DATA --77621973
09862	E0342840 (00064)	DATA --75159476
09864	DCC207C0 (00065)	DATA --72467132
09866	D900F840 (00066)	DATA --69534210
09868	D4ED8240 (00067)	DATA --66349818
0986A	D0840840 (00068)	DATA --62985408
0986C	C8C4F040 (00069)	DATA --59194915
0986E	C6ACF340 (00070)	DATA --55215304
09870	C13CE7C0 (00071)	DATA --50967117
09872	B0765D40 (00072)	DATA --46454968
09874	B5C5140 (00073)	DATA --41687981
09876	AFF354C0 (00074)	DATA --36680087
09878	A8419940 (00075)	DATA --31450189
0987A	A14EE40 (00076)	DATA --26022125
0987C	9124AE40 (00077)	DATA --20424442
0987E	92C09440 (00078)	DATA --14689926
09880	805595C0 (00079)	DATA --80854935
09882	8C97378F (00080)	DATA --82958537
09884	3C97413F (00081)	DATA --808054942
09886	8A559640 (00082)	DATA --814689934
09888	12C098C0 (00083)	DATA --82042450
0988A	1124AF40 (00084)	DATA --826022132
0988C	214EEC0 (00085)	DATA --831450195
0988E	204199C0 (00086)	DATA --83668895
09890	2FP35540 (00087)	DATA --841687988
09892	355C51C0 (00088)	DATA --846454975
09894	3B765DC0 (00089)	DATA --850967123
09896	413CE840 (00090)	DATA --855215311
09898	46ACF3C0 (00091)	DATA --859194921
0989A	40C4FDC0 (00092)	DATA --862905414
0989C	50840AC0 (00093)	DATA --866349822
0989E	54E08240 (00094)	DATA --869534215
098A0	5080F8C0 (00095)	DATA --872467136
098A2	5CC207C0 (00096)	DATA --875159479
098A4	60342040 (00097)	DATA --877621976
098A6	635B2040 (00098)	DATA --879868749
098A8	66306440 (00099)	DATA --881912915
098AA	68093940 (00100)	DATA --883768244
098AC	6A392DC0 (00101)	DATA --885448480
098AE	6D5FC1C0 (00102)	DATA --73101083
098B0	0091C340 (00103)	DATA --69104202
098B2	D0741040 (00104)	DATA --64634765
098B4	D28004C0 (00105)	DATA --59672950
098B6	CC61A240 (00106)	DATA --54208627
098B8	C5631540 (00107)	

CODING TABLE FOR K2 (32)

CTHK2:

0900A 0DC08DC0 (00100)	DATA --40243076
0900C 057F05C0 (00109)	DATA --41795417
0900E AC8A0740 (00110)	DATA --34896366
0900B A3510640 (00111)	DATA --27597119
09002 998F2540 (00112)	DATA --19965846
09004 0F7749C0 (00113)	DATA --12002789
09006 D208503F (00114)	DATA --04045165
09008 5208443F (00115)	DATA 0.04045156
0900A 0F774940 (00116)	DATA 0.12082781
0900C 190E24C0 (00117)	DATA 0.19965837
0900E 27530540 (00118)	DATA 0.27597111
0900B 2CAAD6C0 (00119)	DATA 0.34896358
09002 357F84C0 (00120)	DATA 0.41795409
09004 30C08CC0 (00121)	DATA 0.48243069
09006 456314C0 (00122)	DATA 0.54208621
09008 4C61A1C0 (00123)	DATA 0.59672952
0900A 520804C0 (00124)	DATA 0.64634768
0900C 50741040 (00125)	DATA 0.69104198
0900E 5D91C2C0 (00126)	DATA 0.73101079
0900B 621060C0 (00127)	DATA 0.76652206
09002 602163C0 (00128)	DATA 0.79709401
09004 69A0E240 (00129)	DATA 0.82546644
09006 6CBF6DC0 (00130)	DATA 0.84959197
09008 6F707140 (00131)	DATA 0.87061901
0900A 71C6E8C0 (00132)	DATA 0.88080300
0900C 73CD37C0 (00133)	DATA 0.90470025
0900E 750CECC0 (00134)	DATA 0.91036321
0900B DCC263C0 (00135)	DATA --72460231
09002 04A54FC0 (00136)	DATA --06129490
09004 C81C9640 (00137)	DATA --58608991
09006 C81C6240 (00138)	DATA --50086628
09008 3AE0240 (00139)	DATA --40376312
0900A A5F7B940 (00140)	DATA --29662241
0900C 9739EF40 (00141)	DATA --10145542
0900E F01AC0F (00142)	DATA --06100426
0900B 7D1AC0F (00143)	DATA 0.06100626
09002 1739EE40 (00144)	DATA 0.10145541
09004 25F7B940 (00145)	DATA 0.29662241
09006 33AF0240 (00146)	DATA 0.40376112
09008 401C6240 (00147)	DATA 0.50086619
0900A 481C9640 (00148)	DATA 0.58608990
0900C 54A54F40 (00149)	DATA 0.66129489
0900E 5CC263C0 (00150)	DATA 0.72460230
0900B C4292040 (00151)	DATA --53250506
09002 9983C0C0 (00152)	DATA --45079815
09004 A020A940 (00153)	DATA --36076742
09006 A1096740 (00154)	DATA --26347055
09008 940A48C0 (00155)	DATA --16050062
0900A EF6AB0F (00156)	DATA --05391451
0900C 626AB0F (00157)	DATA 0.05391451
0900E 140A48C0 (00158)	DATA 0.16050061
0900B 21B966C0 (00159)	DATA 0.26347053
09002 2F20A040 (00160)	DATA 0.36076741

DECODING TABLE FOR K3 (16)

DECODING TABLE FOR K4 (16)

89924	3987C0C0	(00161)	DATA	0.45879814
89926	44292040	(00162)	DATA	0.53250505
89928	4D7C8640	(00163)	DATA	0.68532677
8992A	55AC2340	(00164)	DATA	0.66931576
8992C	5CC267C0	(00165)	DATA	0.72460230
8992E	62D25C40	(00166)	DATA	0.77200470
89930	8C9966C0	(00167)	DATA	0.77343143
89932	82E097C0	(00168)	DATA	0.39772316
89934	A078A340	(00169)	DATA	0.31618157
89936	9D664CC0	(00170)	DATA	0.22968443
89938	91D7D040	(00171)	DATA	0.13940015
8993A	DFB78F3F	(00172)	DATA	0.04673719
8993C	5F878CBF	(00173)	DATA	0.04673717
8993E	11D7D040	(00174)	DATA	0.13940014
89940	1D664CC0	(00175)	DATA	0.22968442
89942	2078A340	(00176)	DATA	0.31618155
89944	32E897C0	(00177)	DATA	0.39772316
89946	3C9966C0	(00178)	DATA	0.47343142
89948	45786240	(00179)	DATA	0.54273631
8994A	4D7C8640	(00180)	DATA	0.68532677
8994C	54A54F40	(00181)	DATA	0.66129489
8994E	5AF92440	(00182)	DATA	0.71872024
89950	A9578440	(00183)	DATA	0.32290331
89952	9E8E0BC0	(00184)	DATA	0.23400363
89954	924022C0	(00185)	DATA	0.14250225
89956	E1E0DF8F	(00186)	DATA	0.04781699
89958	61E0D0BF	(00187)	DATA	0.47801697
8995A	124022C0	(00188)	DATA	0.14250224
8995C	1E0F0840	(00189)	DATA	0.23400362
8995E	29578440	(00190)	DATA	0.32290330
89960	33F43840	(00191)	DATA	0.48589048
89962	3DC65740	(00192)	DATA	0.40261537
89964	46001BC0	(00193)	DATA	0.55258511
89966	4E81640	(00194)	DATA	0.61554603
89968	55F46A40	(00195)	DATA	0.67152148
8996A	5C418940	(00196)	DATA	0.72075575
8996C	618F6940	(00197)	DATA	0.76365393
8996E	667F294E	(00198)	DATA	0.80072517
89970	41F8908A	(00199)	DATA	0.000000307255794
89972	5085E13A	(00200)	DATA	0.000000375037437
89974	6209C33A	(00201)	DATA	0.000000459727080
89976	78C3003A	(00202)	DATA	0.000000562341396
89978	893R7780	(00203)	DATA	0.000000687059929
8997A	80480330	(00204)	DATA	0.000000841395202
8997C	8D094E39	(00205)	DATA	0.000001029200590
8997E	10E5A200	(00206)	DATA	0.000001258925660
89980	1482500	(00207)	DATA	0.000001539926070
89982	19402830	(00208)	DATA	0.000001836493300
89984	1ECC0000	(00209)	DATA	0.000002304093330
89986	2503E430	(00210)	DATA	0.000002818303390
89988	2E456700	(00211)	DATA	0.000003447466740
8998A	30996230	(00212)	DATA	0.000004216965940
8998C	30996230	(00213)	DATA	0.000004216965940

JCODING TABLE FOR K5 (16)

JCODING TABLE FOR K6 (16)

JCODING TABLE FOR (OVERALL) ENERGY (64)

0990C	4530043R (00214)	DATA	0.0000005150222220
0990E	54A70000 (00215)	DATA	0.0000006309575000
0990H	67900000 (00216)	DATA	0.0000007717916330
0990J	7E050030 (00217)	DATA	0.000000940610360
0990L	89A7003C (00218)	DATA	0.0000011547022100
0990N	99D9600C (00219)	DATA	0.0000014125376900
0990P	0E7F7A0C (00220)	DATA	0.0000017202639000
0990R	110A073C (00221)	DATA	0.0000021134093600
0990T	15AFC03C (00222)	DATA	0.0000025052735200
0990V	1A06F00C (00223)	DATA	0.0000031622702000
0990X	2M72090C (00224)	DATA	0.0000038601200700
0990Z	2700D93C (00225)	DATA	0.0000047315136300
0991A	300C010C (00226)	DATA	0.0000057076205800
0991B	3B63050C (00227)	DATA	0.0000070794509000
0991C	40A4710C (00228)	DATA	0.0000086596445000
0991E	50D8003C (00229)	DATA	0.0000105925379000
0991G	6C00A03C (00230)	DATA	0.0000129560696000
0991I	004F3530 (00231)	DATA	0.0000158409336000
0991K	0A200430 (00232)	DATA	0.0000193065209000
0991M	0C6EC000 (00233)	DATA	0.0000237137406000
0991O	0F35A330 (00234)	DATA	0.0000290060133000
0991Q	179A3900 (00235)	DATA	0.0000354813456000
0991S	16C13030 (00236)	DATA	0.0000434010303000
0991U	10D56930 (00237)	DATA	0.0000530004504000
0991W	2200A030 (00238)	DATA	0.0000649301709000
0991Y	29A50030 (00239)	DATA	0.0000794320262000
0992A	32F0F000 (00240)	DATA	0.0000971628124000
0992C	3F4FC000 (00241)	DATA	0.0001108502310000
0992E	4C305030 (00242)	DATA	0.0001453704520000
0992G	5030A030 (00243)	DATA	0.0001770279600000
0992I	72001000 (00244)	DATA	0.0002175204340000
0992K	0007FA3E (00245)	DATA	0.0002660725400000
0992M	0AA2700E (00246)	DATA	0.0003254617900000
0992O	0009000E (00247)	DATA	0.0003901072000000
0992Q	07F4FA0E (00248)	DATA	0.0004069675720000
0992S	1304C60E (00249)	DATA	0.0005956622120000
0992U	17E0170E (00250)	DATA	0.0007206182370000
0992W	10345A0E (00251)	DATA	0.0008912509730000
0992Y	2309213E (00252)	DATA	0.0010901045700000
0993A	2002630E (00253)	DATA	0.0013335215500000
0993C	3573450E (00254)	DATA	0.0016111730900000
0993E	4161790E (00255)	DATA	0.0019526254000000
0993G	47F9053E (00256)	DATA	0.0024406191100000
0993I	6103343E (00257)	DATA	0.0029853027400000
0993K	77A9060E (00258)	DATA	0.0036517415100000
0993M	0925E00F (00259)	DATA	0.0044668363400000
0993O	0030A30F (00260)	DATA	0.0054630661100000
0993Q	0000C03F (00261)	DATA	0.0066034391900000
0993S	10072C0F (00262)	DATA	0.0081752307000000
0993U	147A113F (00263)	DATA	0.0099999999800000
0993W	370FC04A (00264)	DATA	0.43651583
0993Y	721402C0 (00265)	DATA	0.09125094
0994A	00400541 (00266)	DATA	1.65958609

JCODING TABLE FOR DELTA-GAIN (4)

09976 17400043 (00267)	DATA 1000.00000000		
09970 7034C740 (00268)	DATA -0.97017319		)/CODING TABLE FOR RESIDUAL (4)
09974 00000000 (00269)	DATA 0.00000000		
0997C 7034C740 (00270)	DATA 0.97017319		
0997E 17400043 (00271)	DATA 1000.00000000		
09972 (00272)			
09973 (00273)			
09974 (00274)			
09975 (00275)			
09976 (00276)			
09977 (00277)			
09978 (00278)			
09979 (00279)			
09980 (00280)			
09981 (00281)			
09982 (00282)			
09983 (00283)			
09984 (00284)			
09985 (00285)			
09986 (00286)			
09987 (00287)			
09988 (00288)			
09989 (00289)			
09990 (00290)			
09991 (00291)			
09992 (00292)			
09993 (00293)			
09994 (00294)			
09995 (00295)			
09996 (00296)			
09997 (00297)			
09998 (00298)			
09999 (00299)			
09A00 (00300)			
09A01 (00301)			
09A02 (00302)			
09A03 (00303)			
09A04 (00304)			
09A05 (00305)			
09A06 (00306)			
09A07 (00307)			
09A08 (00308)			
09A09 (00309)			
09A10 (00310)			
09A11 (00311)			
09A12 (00312)			
09A13 (00313)			
09A14 (00314)			
09A15 (00315)			
09A16 (00316)			
09A17 (00317)			
09A18 (00318)			
09A19 (00319)			

DECODING TABLES GIVE THE CORRESPONDENCE BETWEEN CODE (AN INTEGER) AND QUANTIZED VALUE (FLOATING POINT).

)/DECODING TABLE FOR C1 AND C3 (8)

)/DECODING TABLE FOR C2 (16)

)/DECODING TABLE FOR K1 (64)

DECODING TABLE FOR K2 (32)

DVLK28

09ABE B1260040 (00373)	DATA --38399474
09AC0 A00*1240 (00374)	DATA --31292941
09AC2 9F7C1C0 (00375)	DATA --23017049
09AC4 940R19C0 (00376)	DATA --16049504
09AC6 8A5A4C0 (00377)	DATA --00077100
09AC8 000H000 (00378)	DATA 0.00000000
09ACA 0A50440 (00379)	DATA 0.00077099
09ACC 14001940 (00380)	DATA 0.16049496
09ACE 1E7C1140 (00381)	DATA 0.23017041
09AD0 200F1140 (00382)	DATA 0.31292932
09AD2 31260C0 (00383)	DATA 0.38399466
09AD4 39034C0 (00384)	DATA 0.45070433
09AD6 41A61740 (00385)	DATA 0.51208119
09AD8 40F70040 (00386)	DATA 0.57003706
09ADA 4FA300C0 (00387)	DATA 0.62216294
09ADC 55A090C0 (00388)	DATA 0.66929903
09ADE 5815C340 (00389)	DATA 0.71160164
09AE0 5F0939C0 (00390)	DATA 0.74930502
09AE2 6427A0C0 (00391)	DATA 0.70270507
09AE4 67F3E440 (00392)	DATA 0.81213313
09AE6 6A4100C0 (00393)	DATA 0.83793800
09AE8 6E2300C0 (00394)	DATA 0.86047144
09AFA 70A655C0 (00395)	DATA 0.88007615
09AFC 72037940 (00396)	DATA 0.89707066
09AEE 740551C0 (00397)	DATA 0.91170370
09AF0 E059040 (00398)	DATA --75245064
09AF2 000F0E40 (00399)	DATA --69433287
09AF4 080F9340 (00400)	DATA --62547535
09AF6 C5063340 (00401)	DATA --54526930
09AF8 0A110940 (00402)	DATA --45360014
09AFA AC00C4C0 (00403)	DATA --35134251
09AFC 9E047A40 (00404)	DATA --23900272
09AFE 0F947740 (00405)	DATA --12171034
09000 0000000 (00406)	DATA 0.00000000
09002 0F947740 (00407)	DATA 0.12171033
09004 1E047A40 (00408)	DATA 0.23900271
09006 2CF0C440 (00409)	DATA 0.35134250
09008 3A1100C0 (00410)	DATA 0.45360013
0900A 45C062C0 (00411)	DATA 0.54526937
0900C 500F9340 (00412)	DATA 0.62547535
0900E 500F0E40 (00413)	DATA 0.69433286
09010 C0F70340 (00414)	DATA --57005347
09012 0F1200C0 (00415)	DATA --49273609
09014 04110A40 (00416)	DATA --40677000
09016 A00F0940 (00417)	DATA --31293977
09018 903510C0 (00418)	DATA --21257565
0901A 0DC319C0 (00419)	DATA --10751649
0901C 0000000 (00420)	DATA 0.00000000
0901E 0DC319C0 (00421)	DATA 0.10751640
09020 103510C0 (00422)	DATA 0.21257564
09022 200F0940 (00423)	DATA 0.31293977
09024 34110A40 (00424)	DATA 0.40677000
09026 3F1200C0 (00425)	DATA 0.49273609

DECODING TABLE FOR K3 (16)

DECODING TABLE FOR K4 (16)

PAGE 10: (ABNDJ)CUCALG>NBI6T.MSD.1, 10-NOV-80 19:15:02, ED: KPIZLD
CODING AND DECODING TABLES

09820 40F70340 (00426) DATA 0-57005347
09821 510065C0 (00427) DATA 0-63043908
09822 59596740 (00428) DATA 0-69080000
09823 5FE90940 (00429) DATA 0-74932021
09824 C1232640 (00430) DATA --50090009
09825 070A4040 (00431) DATA --43634920
09826 40C6F240 (00432) DATA --35763304
09827 A3019540 (00433) DATA --27340579
09828 97A00DC0 (00434) DATA --10493246
09829 00F44C0 (00435) DATA --09327063
09830 00000000 (00436) DATA 0-00000000
09831 00F04C0 (00437) DATA 0-09327062
09832 17A00DC0 (00438) DATA 0-10493246
09833 23019540 (00439) DATA 0-27340579
09834 20C6F1C0 (00440) DATA 0-35763303
09835 370A4040 (00441) DATA 0-43634920
09836 4123E640 (00442) DATA 0-50890000
09837 499605C0 (00443) DATA 0-57489004
09838 512C13C0 (00444) DATA 0-63415766
09839 57E93940 (00445) DATA 0-60600496
09840 AEB09640 (00446) DATA --36516072
09841 A3C616C0 (00447) DATA --27940269
09842 9034C8C0 (00448) DATA --10910990
09843 8C3695C0 (00449) DATA --09541500
09844 00000000 (00450) DATA 0-00000000
09845 0C3695C0 (00451) DATA 0-09541579
09846 1034C0C0 (00452) DATA 0-10910909
09847 23C616C0 (00453) DATA 0-27940267
09848 2E009640 (00454) DATA 0-36516071
09849 30F00040 (00455) DATA 0-44506935
09850 425D2540 (00456) DATA 0-51046750
09851 40F0340 (00457) DATA 0-58494609
09852 527092C0 (00458) DATA 0-64439614
09853 5935F3C0 (00459) DATA 0-69695902
09854 501909C0 (00460) DATA 0-74296685
09855 64350640 (00461) DATA 0-70200344
09856 64350640 (00462) DATA 0-70200344

DECODING TABLE FOR K5 (16)

DECODING TABLE FOR K6 (16)

DECODING TABLE FOR SQR(ENERGY) (64)

DVLC:

09870 5762F900 (00464) DATA 0-0001666767600000
09871 60A60130 (00465) DATA 0-0001843423110000
09872 6AE45530 (00466) DATA 0-0002030001790000
09873 76309930 (00467) DATA 0-0002254000310000
09874 002C030E (00468) DATA 0-0002493076930000
09875 0000000E (00469) DATA 0-0002750195510000
09876 09FF070E (00470) DATA 0-0003050520070000
09877 000F200E (00471) DATA 0-0003373044230000
09878 0C3A250E (00472) DATA 0-0003731427710000
09879 0005E73E (00473) DATA 0-0004126910390000
09880 0FF4010E (00474) DATA 0-0004564309060000
09881 100A900E (00475) DATA 0-0005000666310000
09882 1240700E (00476) DATA 0-0005503095120000
09883 1430D20E (00477) DATA 0-0006174030700000
09884 1660013E (00478) DATA 0-0006082920105000

0900E 1007FF3E (00479)	DATA	0.000755309580000
0900F 105070E (00480)	DATA	0.000835362631000
0900G 1F4633E (00481)	DATA	0.000923900121000
0900H 217P100E (00482)	DATA	0.001021021530000
0900I 250020E (00483)	DATA	0.001130121220000
0900J 207421E (00484)	DATA	0.001249899330000
0900K 204C300E (00485)	DATA	0.001302372370000
0900L 321930E (00486)	DATA	0.001520005700000
0900M 376070E (00487)	DATA	0.001690927750000
0900N 3047E73E (00488)	DATA	0.001870143910000
0900O 43C6900E (00489)	DATA	0.002060354700000
0900P 4AF50E2E (00490)	DATA	0.002287573410000
0900Q 52E7650E (00491)	DATA	0.002530026190000
0900R 5000C03E (00492)	DATA	0.002798176140000
0900S 65609C3E (00493)	DATA	0.003094746890000
0900T 7020100E (00494)	DATA	0.003422748020000
0900U 7C0R348E (00495)	DATA	0.003705515550000
0900V 0097003F (00496)	DATA	0.004106730540000
0900W 0070033F (00497)	DATA	0.004638469660000
0900X 0A7D013F (00498)	DATA	0.005121230760000
0900Y 0099433F (00499)	DATA	0.005640232200000
0900Z 0CD4500F (00500)	DATA	0.006264335030000
0901A 0E34693F (00501)	DATA	0.006920273240000
0901B 0FB1063F (00502)	DATA	0.007662508290000
0901C 1150300F (00503)	DATA	0.008474713420000
0901D 1321C3F (00504)	DATA	0.009372923060000
0901E 153AF13F (00505)	DATA	0.010366329900000
0901F 177AF90F (00506)	DATA	0.011465025700000
0901G 19F00F3F (00507)	DATA	0.012601600000000
0901H 1C0RAA0F (00508)	DATA	0.014024100500000
0901I 1FC3F43F (00509)	DATA	0.015514472100000
0901J 2321D50F (00510)	DATA	0.017154300000000
0901K 260R100F (00511)	DATA	0.018972520500000
0901L 2AF9520F (00512)	DATA	0.020983359500000
0901M 2F07510F (00513)	DATA	0.023207321100000
0901N 3490E50F (00514)	DATA	0.025666994300000
0901O 3A2327E (00515)	DATA	0.028307361000000
0901P 404C920F (00516)	DATA	0.031390051400000
0901Q 4710200F (00517)	DATA	0.034723622700000
0901R 4FA6003F (00518)	DATA	0.038403871500000
0901S 56FC030F (00519)	DATA	0.042474189000000
0901T 6034E53F (00520)	DATA	0.046975890200000
0901U 6A673C0F (00521)	DATA	0.051954722000000
0901V 75AF303F (00522)	DATA	0.057461244100000
0901W 082273C0 (00523)	DATA	0.063551381000000
0901X 00FF2A40 (00524)	DATA	0.070287000400000
0901Y 09F34540 (00525)	DATA	0.077736505300000
0901Z 00013740 (00526)	DATA	0.085975561300000
0902A 0C2006C0 (00527)	DATA	0.095087840000000
0902B 3F011240 (00528)	DATA	0.40977802
0902C 65AC0C40 (00529)	DATA	0.79432024
0902D 00F9E4C1 (00530)	DATA	1.12201844
0902E 00F0A41 (00531)	DATA	1.49623564

DECODING TABLE FOR SORT(DELTA-GAIN) (4)

PAGE 121 (00002)DC116>000167-M30.1, 10-Nov-88 19:15:02, Ed: KFIELD
CODING AND DECODING TABLES

09070 0C000541 (00532) DVL0:	DATA -1.50409251	)DECODING TABLE FOR REGIONAL (4) (CODED 1,0,2,3)
0907A 09C930C0 (00533)	DATA -0.45145300	
0907C 39C930C0 (00534)	DATA 0.45145300	
0907E 0C000541 (00535)	DATA 1.50409251	
09C00 09C930C0 (00536) DVLUP:	DATA -0.45145300	)DVL0 IN FOLDED BINARY ORDER (4) (CODED 0,1,2,3)
09C02 0C000541 (00537)	DATA -1.50409251	
09C04 39C930C0 (00538)	DATA 0.45145300	
09C06 0C000541 (00539)	DATA 1.50409251	
09C08	END	

PAGE 13: (0000)<0001>000107.MSU.1, 10-NOV-00 19:15:02, 0000
 CNOING AND DECODING TABLES

CTNC1:	0000 (0001)
CTNC2:	0010 (0022)
CTNC3:	0000 (0014)
CTNDC1:	0000 (0026)
CTNG1:	0000 (0019)
CTNR1:	0000 (0039)
CTNR2:	0000 (0010)
CTNR3:	0000 (0013)
CTNR4:	0010 (0015)
CTNR5:	0030 (0016)
CTNR6:	0050 (0018)
CTNR7:	0000 (0026)
DVLC1:	0000 (0027)
DVLC2:	0010 (0028)
DVLC3:	0000 (0027)
DVLC4:	0000 (0028)
DVLC5:	0000 (0028)
DVLC6:	0000 (0028)
DVLC7:	0000 (0028)
DVLC8:	0000 (0028)
DVLC9:	0000 (0028)
DVLC10:	0000 (0028)
DVLC11:	0000 (0028)
DVLC12:	0000 (0028)
DVLC13:	0000 (0028)
DVLC14:	0000 (0028)
DVLC15:	0000 (0028)
DVLC16:	0000 (0028)
DVLC17:	0000 (0028)
DVLC18:	0000 (0028)
DVLC19:	0000 (0028)
DVLC20:	0000 (0028)

LINES WITH ERRORS: 0 (MAP VERSION 000101.10) 2- 0

[000001<00016>000000.2, 29-Dec-88 15:44:23, Ed: WULF

T A B L E O F C O N T E N T S

BBN 16 ED/S SPEECH CODER INPUT/OUTPUT PROGRAMS	PAGE 2
SYMBOL DEFINITIONS	PAGE 3
PATCHES TO NON-ARRAY AND ARRAY FUNCTION DISPATCH TABLES	PAGE 6
PATCHES TO INTERRUPT SERVICE ROUTINES	PAGE 7
I/O BUFFER DEFINITIONS	PAGE 8
INTEGER SCALAR TABLE DEFINITIONS FOR I/O ROUTINES	PAGE 10
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 12
ADAM INTERFACE AND SYSTEM CLOCKS PROGRAMS	PAGE 15
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 17
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 19
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 21
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 23
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 26
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 29
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 31
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 38
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 45
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 47
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT	PAGE 49

SYMBOL DEFINITIONS

```

(00005) "SYMBOL DEFINITIONS
(00006) ;
00000001 (00007) H$ = 1
00000002 (00008) V$ = 2
00000003 (00009) ;
00000004 (00010) ACQTHR = 10
00000005 (00011) ;
00000006 (00012) ;
00000007 (00013) AFD$ORC = $0E0
00000008 (00014) BITS15 = 0'7777'
00000009 (00015) ;
00000010 (00016) ;BIT-MASKS USED IN THE PROPAR ROUTINE
00000011 (00017) B0 = 1
00000012 (00018) B1 = 2
00000013 (00019) B2 = 4
00000014 (00020) B3 = 8
00000015 (00021) B4 = $10
00000016 (00022) B5 = $20
00000017 (00023) B10 = $1+00
00000018 (00024) B210 = $2+$1+00
00000019 (00025) B32 = $3+$2
00000020 (00026) B321 = $3+$2+$1
00000021 (00027) B54 = $5+$4
00000022 (00028) ;BIT-MASKS SHIFTED 1 PLACE LEFT FOR USE IN THE CORPAR ROUTINE
00000023 (00029) C0 = 2
00000024 (00030) C1 = 4
00000025 (00031) C2 = 8
00000026 (00032) C3 = $10
00000027 (00033) C4 = $20
00000028 (00034) C5 = $40
00000029 (00035) C10 = C1+C0
00000030 (00036) C21 = C2+C1
00000031 (00037) C210 = C2+C1+C0
00000032 (00038) C32 = C3+C2
00000033 (00039) C321 = C3+C2+C1
00000034 (00040) C54 = C5+C4
00000035 (00041) ;
000021FC (00042) CSPU$MDS = $21FC
00000036 (00043) ;
00000037 (00044) [C00NDJ]C0CA16>BBN16T-ADDRS.MSD.2, 9-Sep-88 15:08:32, E4: KFIELD
00000038 (00045) ;SYMBOL DEFINITIONS FOR BBN16 CODING/DECODING TABLE ADDRESSES
00000039 (00046) ; TABLES RESIDE ON BUS1, STARTING AT $900.
00000040 (00047) ; TABLE CONTENTS ARE DEFINED IN MODULE "BBN16T.MSD".
00000041 (00048) ;
00000042 (00049) CTHC1=$900
00000043 (00050) CTHC2=CTHC1+$0*W$
00000044 (00051) CTHC3=CTHC1
00000045 (00052) CTHK1=CTHC2+$16*W$
00000046 (00053) CTHK2=CTHK1+$64*W$
00000047 (00054) CTHK3=CTHK2+$32*W$
00000048 (00055) CTHK4=CTHK3+$16*W$
00000049 (00056) CTHK5=CTHK4+$16*W$
00000050 (00057) CTHK6=CTHK5+$16*W$

```

00009970 (00050) CTNC =CTHC+16*W\$
00009970 (00059) CTNC=CTHC+64*W\$
00009970 (00060) CTNU =CTHC+ 4*W\$
00009970 (00061) DVLC1=CTHC + 4*W\$
00009970 (00062) DVLC2=DVLC1+ 0*W\$
00009970 (00063) DVLC3=DVLC1
00009970 (00064) DVLC4=DVLC2+16*W\$
00009970 (00065) DVLC5=DVLC1+64*W\$
00009970 (00066) DVLC6=DVLC2+32*W\$
00009970 (00067) DVLC7=DVLC3+16*W\$
00009970 (00068) DVLC8=DVLC4+16*W\$
00009970 (00069) DVLC9=DVLC5+16*W\$
00009970 (00070) DVLC =DVLC6+16*W\$
00009970 (00071) DVLC=DVLC +64*W\$
00009970 (00072) DVLU =DVLC+ 4*W\$
00009970 (00073) DVLU=DVLU + 4*W\$
00009970 (00074))

00004022 (00075))SNAP EXEC INTERRUPT ROUTINE ADDRESSES
00004022 (00076) D16\$INT1 = \$4022)MODEM SCROLL INT 1 (MODEM)
00004022 (00077) D16\$INT2 = \$4030)MODEM SCROLL INT 2 (MODEM)
00004022 (00078) D22\$INT1 = \$40FA)ADM INT 1
00004022 (00079) D23\$INT1 = \$411E)ADM INT 1
00004022 (00080))

00000700 (00081) POTS = \$700)START OF MONARRAY FUNCTION DISPATCH TABLE
00000700 (00082))CFLAG SET/CLEAR CONSTANTS

00000700 (00083) SET = 0*40
00000700 (00084) CLR = 0
00000700 (00085) C0 = 4
00000700 (00086) C1 = 5
00000700 (00087) C2 = 6
00000700 (00088) C3 = 7
00000700 (00089))

00000000 (00090))MODEM-SCROLL INPUT DATA WORD BITS
00000000 (00091) IDW\$OHK = \$00)LOCAL HANDSET HOOKSWITCH STATE
00000000 (00092) IDW\$SI = \$40)SIGNAL RATE INDICATOR
00000000 (00093) IDW\$IC = \$20)INCOMING CALL
00000000 (00094) IDW\$CS = \$10)CLEAR TO SEND
00000000 (00095) IDW\$DM = 0)DATA MODE
00000000 (00096) IDW\$RR = 4)RECEIVER READY
00000000 (00097) IDW\$SQ = 2)SIGNAL QUALITY
00000000 (00098) IDW\$RD = 1)RECEIVE DATA
00000000 (00099))

00000502 (00100) ISVTS = \$502)START OF INTEGER SCALAR TABLE IN SNAP EXEC
00000004 (00101) LOSETHR = 4)SYNC-SEARCH LOSE-SYNC THRESHOLD: NUMBER OF
00000000 (00102)) SYNC ERRORS WITHOUT 2 CONSECUTIVE GOOD-SYNC
00000000 (00103)) FRAMES NEEDED TO LOSE SYNC.

00000000 (00104))
00000000 (00105))MODEM-SCROLL OUTPUT DATA WORD BITS
00000000 (00106) ODW\$TH = 0)TERMINAL READY
00000000 (00107) ODW\$RS = 4)REQUEST TO SEND
00000000 (00108) ODW\$SR = 2)SIGNALING RATE
00000000 (00109) ODW\$SD = 1)SEND DATA
00000000 (00110))

PAGE 5: [000010C0116]000160.MSO.2, 29-Dec-88 15:44:23, Ed: WOLF
SYMBOL DEFINITIONS

```

00000000 (00111) RANDSL = $00000          LOWER AND UPPER LIMITS FOR ERRPTR
00001C77 (00112) RANDSW = $01C77          / IN MISSIN
00003PCE (00113) SVSPFLGS = $1PCE         /MAP SYSTEM FLAG REGISTER IN PSEUDO-MEMORY
0000000C (00114) TMDITS = 00000000000000 /CONTROL BITS USED FOR TRANSMIT MODEN
(00115) /
(00116) /SYNCSOP IS A "REGISTER" IN THE ADAM AND AOM USED FOR CPU-SCROLL
(00117) / COMMUNICATION.
(00118) /
(00119) /
(00120) /THE SAP (SET ADDRESS FIELD) INSTRUCTION IS ONE OF THE NEW ONES ADDED
(00121) / TO THE CPU BY THE "REV. 10" MICROCODE REVISION.
(00122) /
(00123) /
OPADD SAP,(1.LS. 14) + (29.LS. 8) + $EP

```

PAGE 6: [000010C0116]000160.MSO.2, 29-Dec-88 15:44:23, Ed: WOLF
PATCHES TO NON-ARRAY AND ARRAY FUNCTION DISPATCH TABLES

```

(00124) "PATCHES TO NON-ARRAY AND ARRAY FUNCTION DISPATCH TABLES
(00125) HL = FDIS + (WS + 121)
0000000A (00126) ADDR PRORES          /FCB 121 (SWAP FUNCTION ADDRESS)
00003CCA (00126) ADDR PRORES          /FCB 122 (PROPAR)
00003CE4 (00127) ADDR PROPAR          /FCB 123 (CORPAR)
00003F12 (00128) ADDR CORPAR          /FCB 124 (ADINT)
0000495C (00129) ADDR ADINT          /FCB 125 (TMINT)
00003000 (00130) ADDR TMODEMINT      /FCB 126 (RMINT)
00003067 (00131) ADDR RMODEMINT      /FCB 127 (DAINT)
00003C06 (00132) ADDR ADINT
(00133) /
(00134) /
00000150 (00135) HL = AFDTORG+3*WS*(188-128) /FCB 188 (DECODE)
0001F3FA (00136) ADDR DCDB(R7,1)
0001E3FA (00137) ADDR DCDB(R7,1)
0001021FC (00138) ADDR CSPUSMOS(1,0)
(00139) /
00000156 (00140) HL = AFDTORG+3*WS*(189-128) /FCB 189 (DECODE)
0001E3FA (00141) ADDR DCDB(R7,1)
0001E3FA (00142) ADDR DCDB(R7,1)
0001021FC (00143) ADDR CSPUSMOS(1,0)
(00144) /

```

PAGE 7: (00001<0C416>00016U.WS0.2, 29-Dec-88 15:44:23, Ed: WULF
 PATCHES TO INTERRUPT SERVICE ROUTINES

```

(00145) *PATCHES TO INTERRUPT SERVICE ROUTINES
(00146) ; THESE PATCHES MAKE SCROLL INTERRUPTS GO TO OUR OWN ROUTINES.
(00147) ; NOTE THAT LOADING THIS FILE WILL MAKE THE NORMAL ADAM/ADM/IOS-2 SWAP
(00148) ; FUNCTIONS INOPERATIVE!
00004022 (00149) BL = D16SINT1
00004023 (00150) BL: INCM
00004024 (00151) CALL
00004025 (00152) RET
00004026 (00153) EVEN
00004027 (00154) JMP
00004028 (00155) ;
00004029 (00156) BL = D16SINT2
00004030 (00157) BL: INCM
00004031 (00158) CALL
00004032 (00159) RET
00004033 (00160) EVEN
00004034 (00161) JMP
00004035 (00162) ;
00004036 (00163) BL = D23SINT1
00004037 (00164) BL: INCM
00004038 (00165) CALL
00004039 (00166) RET
00004040 (00167) EVEN
00004041 (00168) JMP
00004042 (00169) ;
00004043 (00170) BL = D23SINT1
00004044 (00171) BL: INCM
00004045 (00172) CALL
00004046 (00173) RET
00004047 (00174) EVEN
00004048 (00175) JMP
00004049 (00176) ;
  
```


PAGE 9: (P001)CCL16>00160.M00.2, 29-Dec-88 15:44:23, Kds VOLV
I/O BUFFER DEFINITIONS

```

00000P40 (00230) RSNKA = 47944          )RSINK A BUFFER
00000C20 (00231) RSNKB = 40160          )RSINK B BUFFER
00000C70 (00232) RSNKC = 40376          )D/A "SILENCE" DATA
00000000 (00233) RDATA = 40552          )D/A A BUFFER
00000F10 (00234) RDATA = 40000          )D/A B BUFFER
0000035 ) (00235) )
0000036 ) TWO BUFFERS FOR THE SYNCSEARCH SUBROUTINE ARE DEFINED JUST BELOW THE
0000037 ) 16K (FULLWORD) BOUNDARY ON BUS 1.
000070EC (00238) RSPF = $70EC          )SYNC SEARCH PREVIOUS FRAME ($72 HALFWORDS)
000070F6 (00239) RASS = $70F6          )SYNC SEARCH SUB-SYNC COUNTS (SAME)
0000240 ) (00240) )
0000241 ) INITIALIZE ALTERNATING SYNC BITS IN TWO-DEM BUFFERS
0000A96A (00242) BL = TWOA             )
0000A96C (00243) DATA 0 + THBITS     )
0000A074 (00244) BL = TWOAB          )
0000A96D (00245) DATA 1 + THBITS     )
0000A96E (00246) )

```



```

(00291) ; MISC. INTEGER MEMORIES (AND INIT VALUES)
(00292) ;
0000053P (00293) RUN = ISVT$+61          ; ENCODER RUN FLAG
0000053P (00294) ; (62 FREE FOR MP1TH USE)
0000053P (00295) RL=RUN
0000053P (00296) DATA 1
0000053P (00297) ;
0000054E (00298) TFCR = ISVT$+76          ; TRANSMITTER FRAME COUNTER (B)
0000054P (00299) RPCR = ISVT$+77          ; RECEIVER FRAME COUNTER (B)
00000550 (00300) RLCTR = ISVT$+78          ; RCVR LOST-SYNC COUNTER (B)
00000551 (00301) RORUK = ISVT$+79          ; LOCAL HANDSET ON-HOOK STATE (B)
00000552 (00302) RSYNC = ISVT$+80          ; STATE OF RMODEM SYNC (B)
00000553 (00303) RROFO = ISVT$+81          ; BEGINNING OF FRAME OFFSET: SYNC BIT
0000054A (00304) ; (62 FREE FOR MP1TH)
0000054A (00305) RL=TAODC
0000054A (00306) DATA 90'B
...
(00307) ;
(00308) ; INTERRUPT COUNTERS
(00309) ;
00000555 (00310) ADCR = ISVT$+83          ; COUNTS ADAM (A/D) INTERRUPTS
00000556 (00311) TMCTR = ISVT$+84          ; COUNTS TMODEM INTERRUPTS
00000557 (00312) RMCTR = ISVT$+85          ; COUNTS RMODEM INTERRUPTS
00000558 (00313) DACR = ISVT$+86          ; COUNTS ADM (D/A) INTERRUPTS
00000555 (00314) RL=ADCTR
00000555 (00315) DATA 40'B
...
(00316) ;
(00317) ; DEBUGGING/DEMONSTRATION AIDS
(00318) ;
00000570 (00319) RMCOR = ISVT$+123          ; SET TO WZ TO TELL CORPAR NOT TO
0000057P (00320) ; (124 FREE FOR MP1TH)          ; CORRECT CHANNEL ERRORS (B)
0000057P (00321) RERRS = ISVT$+125          ; SET TO N>0 TO TELL RMODEM TO
00000570 (00322) ; (126 FREE FOR MP1TH)          ; SIMULATE N CHANNEL ERRORS PER FRAME(B)
00000570 (00323) RL=RMCOR
00000570 (00324) DATA 40'B
...
(00325)

```



```

(00326) *MODFM INTERFACE AND SYSTEM CLOCKS PROGRAMS
(00327) ; JVV, 1A-SEP-79
(00328) ;
(00329) ; THERE ARE TWO ENTRY POINTS:
(00330) ; CLESET: CLOCK RATE SETTING (START ADDRESS = 0)
(00331) ; THIS ENTRY SETS THE DATA AND SAMPLE CLOCKS AND STARTS THEM.
(00332) ; THE 16-BIT CLOCK RATE VALUE IS A LITERAL IN THE INSTRUCTION AT
(00333) ; CLEGO. (TRIED BINDING AN INTEGER TABLE VALUE HERE, BUT BECAUSE
(00334) ; WE USE NO BIDS, LDS$ IN THE SNAP EXEC DOES NO BINDING AT ALL)
(00335) ;
(00336) ; RUNMOD: SET CLOCKS AND RUN MODFM TRANSMIT/RECEIVE (START ADDRESS 1)
(00337) ; THIS ENTRY SETS AND STARTS THE CLOCKS AND THEN RUNS THE MODFM
(00338) ; INTERFACE PROGRAM.
(00339) ;
(00340) ; THESE ENDING ADDRESSES ARE TRUNCATED TO 15 BITS BECAUSE THE LLIT
(00341) ; FIELD OF THE JNE INSTRUCTION IS ONLY THAT BIG.
(00342) ; RMA$END = (RNDMA+MBLNGTH-1) .AND. BITS15
(00343) ; RMB$END = (RNDMB+MBLNGTH-1) .AND. BITS15
(00344) ; RMC$END = (RNDMC+MBLNGTH-1) .AND. BITS15
(00345) ; RMA$END = (THDMA+MBLNGTH-1) .AND. BITS15
(00346) ; THB$END = (THDMB+MBLNGTH-1) .AND. BITS15
(00347) ;
(00348) ; CONSTANT FOR SETTING CLOCK RATES TO MODFM=16000 OPS, SIGNAL
(00349) ; SAMPLE RATE=6621 SAMPLES/SEC.
(00350) ; CLRRATES = $74C6
(00351) ;
(00352) ; REGISTER AND FLAG USAGE:
(00353) ; R0 - RCVR BUFFER SWITCH, SET TO MOD$RA, MOD$RB, OR MOD$RC
(00354) ; R1 - RCVR ADDRESS POINTER
(00355) ; R2 - INTR ADDRESS POINTER
(00356) ; P1 SET SIGNIFIES RCVR DATUM READY
(00357) ; P2 SET SIGNIFIES INTR READY FOR NEXT DATUM
(00358) ; P1 IS USED AS INTR BUFFER SWITCH: CLEAR=>THDMA, SET=>THDMDB
(00359) ; P2 IS USED TO REMEMBER WHICH START ADDRESS WAS USED.
(00360) ; INT1 IS USED TO SIGNAL END OF TRANSMITTER BUFFER
(00361) ; INT2 IS USED TO SIGNAL END OF RECEIVER BUFFER
(00362) ;
(00363) ; BUS 1 ADR FOR RTHODEN IDS MODULE
(00364) ;
(00365) ;
(00366) ;
(00367) ; MOD$BGM: BEGIN IN$2(RTHODEN)
(00368) ;
(00369) ;
(00370) ; CLKSET: SF2, JUMP(CLEGO)
(00371) ; RUNMOD: CF2
(00372) ; CLEGO: LOAD(R0, CLRRATES)
(00373) ; ADDL(R0, 0, TP), JIFC(MOD$INIT, P2)
(00374) ; STOP, CF2
(00375) ;
(00376) ; MOD$INIT: LOAD(R0, MOD$RA)
(00377) ; LOAD(R1, RNDMA-1(1), L)
(00378) ;
(00379) ;
(00380) ;
(00381) ;
(00382) ;
(00383) ;
(00384) ;
(00385) ;
(00386) ;
(00387) ;
(00388) ;
(00389) ;
(00390) ;
(00391) ;
(00392) ;
(00393) ;
(00394) ;
(00395) ;
(00396) ;
(00397) ;
(00398) ;
(00399) ;
(00400) ;
(00401) ;
(00402) ;
(00403) ;
(00404) ;
(00405) ;
(00406) ;
(00407) ;
(00408) ;
(00409) ;
(00410) ;
(00411) ;
(00412) ;
(00413) ;
(00414) ;
(00415) ;
(00416) ;
(00417) ;
(00418) ;
(00419) ;
(00420) ;
(00421) ;
(00422) ;
(00423) ;
(00424) ;
(00425) ;
(00426) ;
(00427) ;
(00428) ;
(00429) ;
(00430) ;
(00431) ;
(00432) ;
(00433) ;
(00434) ;
(00435) ;
(00436) ;
(00437) ;
(00438) ;
(00439) ;
(00440) ;
(00441) ;
(00442) ;
(00443) ;
(00444) ;
(00445) ;
(00446) ;
(00447) ;
(00448) ;
(00449) ;
(00450) ;
(00451) ;
(00452) ;
(00453) ;
(00454) ;
(00455) ;
(00456) ;
(00457) ;
(00458) ;
(00459) ;
(00460) ;
(00461) ;
(00462) ;
(00463) ;
(00464) ;
(00465) ;
(00466) ;
(00467) ;
(00468) ;
(00469) ;
(00470) ;
(00471) ;
(00472) ;
(00473) ;
(00474) ;
(00475) ;
(00476) ;
(00477) ;
(00478) ;
(00479) ;
(00480) ;
(00481) ;
(00482) ;
(00483) ;
(00484) ;
(00485) ;
(00486) ;
(00487) ;
(00488) ;
(00489) ;
(00490) ;
(00491) ;
(00492) ;
(00493) ;
(00494) ;
(00495) ;
(00496) ;
(00497) ;
(00498) ;
(00499) ;
(00500) ;
(00501) ;
(00502) ;
(00503) ;
(00504) ;
(00505) ;
(00506) ;
(00507) ;
(00508) ;
(00509) ;
(00510) ;
(00511) ;
(00512) ;
(00513) ;
(00514) ;
(00515) ;
(00516) ;
(00517) ;
(00518) ;
(00519) ;
(00520) ;
(00521) ;
(00522) ;
(00523) ;
(00524) ;
(00525) ;
(00526) ;
(00527) ;
(00528) ;
(00529) ;
(00530) ;
(00531) ;
(00532) ;
(00533) ;
(00534) ;
(00535) ;
(00536) ;
(00537) ;
(00538) ;
(00539) ;
(00540) ;
(00541) ;
(00542) ;
(00543) ;
(00544) ;
(00545) ;
(00546) ;
(00547) ;
(00548) ;
(00549) ;
(00550) ;
(00551) ;
(00552) ;
(00553) ;
(00554) ;
(00555) ;
(00556) ;
(00557) ;
(00558) ;
(00559) ;
(00560) ;
(00561) ;
(00562) ;
(00563) ;
(00564) ;
(00565) ;
(00566) ;
(00567) ;
(00568) ;
(00569) ;
(00570) ;
(00571) ;
(00572) ;
(00573) ;
(00574) ;
(00575) ;
(00576) ;
(00577) ;
(00578) ;
(00579) ;
(00580) ;
(00581) ;
(00582) ;
(00583) ;
(00584) ;
(00585) ;
(00586) ;
(00587) ;
(00588) ;
(00589) ;
(00590) ;
(00591) ;
(00592) ;
(00593) ;
(00594) ;
(00595) ;
(00596) ;
(00597) ;
(00598) ;
(00599) ;
(00600) ;
(00601) ;
(00602) ;
(00603) ;
(00604) ;
(00605) ;
(00606) ;
(00607) ;
(00608) ;
(00609) ;
(00610) ;
(00611) ;
(00612) ;
(00613) ;
(00614) ;
(00615) ;
(00616) ;
(00617) ;
(00618) ;
(00619) ;
(00620) ;
(00621) ;
(00622) ;
(00623) ;
(00624) ;
(00625) ;
(00626) ;
(00627) ;
(00628) ;
(00629) ;
(00630) ;
(00631) ;
(00632) ;
(00633) ;
(00634) ;
(00635) ;
(00636) ;
(00637) ;
(00638) ;
(00639) ;
(00640) ;
(00641) ;
(00642) ;
(00643) ;
(00644) ;
(00645) ;
(00646) ;
(00647) ;
(00648) ;
(00649) ;
(00650) ;
(00651) ;
(00652) ;
(00653) ;
(00654) ;
(00655) ;
(00656) ;
(00657) ;
(00658) ;
(00659) ;
(00660) ;
(00661) ;
(00662) ;
(00663) ;
(00664) ;
(00665) ;
(00666) ;
(00667) ;
(00668) ;
(00669) ;
(00670) ;
(00671) ;
(00672) ;
(00673) ;
(00674) ;
(00675) ;
(00676) ;
(00677) ;
(00678) ;
(00679) ;
(00680) ;
(00681) ;
(00682) ;
(00683) ;
(00684) ;
(00685) ;
(00686) ;
(00687) ;
(00688) ;
(00689) ;
(00690) ;
(00691) ;
(00692) ;
(00693) ;
(00694) ;
(00695) ;
(00696) ;
(00697) ;
(00698) ;
(00699) ;
(00700) ;
(00701) ;
(00702) ;
(00703) ;
(00704) ;
(00705) ;
(00706) ;
(00707) ;
(00708) ;
(00709) ;
(00710) ;
(00711) ;
(00712) ;
(00713) ;
(00714) ;
(00715) ;
(00716) ;
(00717) ;
(00718) ;
(00719) ;
(00720) ;
(00721) ;
(00722) ;
(00723) ;
(00724) ;
(00725) ;
(00726) ;
(00727) ;
(00728) ;
(00729) ;
(00730) ;
(00731) ;
(00732) ;
(00733) ;
(00734) ;
(00735) ;
(00736) ;
(00737) ;
(00738) ;
(00739) ;
(00740) ;
(00741) ;
(00742) ;
(00743) ;
(00744) ;
(00745) ;
(00746) ;
(00747) ;
(00748) ;
(00749) ;
(00750) ;
(00751) ;
(00752) ;
(00753) ;
(00754) ;
(00755) ;
(00756) ;
(00757) ;
(00758) ;
(00759) ;
(00760) ;
(00761) ;
(00762) ;
(00763) ;
(00764) ;
(00765) ;
(00766) ;
(00767) ;
(00768) ;
(00769) ;
(00770) ;
(00771) ;
(00772) ;
(00773) ;
(00774) ;
(00775) ;
(00776) ;
(00777) ;
(00778) ;
(00779) ;
(00780) ;
(00781) ;
(00782) ;
(00783) ;
(00784) ;
(00785) ;
(00786) ;
(00787) ;
(00788) ;
(00789) ;
(00790) ;
(00791) ;
(00792) ;
(00793) ;
(00794) ;
(00795) ;
(00796) ;
(00797) ;
(00798) ;
(00799) ;
(00800) ;
(00801) ;
(00802) ;
(00803) ;
(00804) ;
(00805) ;
(00806) ;
(00807) ;
(00808) ;
(00809) ;
(00810) ;
(00811) ;
(00812) ;
(00813) ;
(00814) ;
(00815) ;
(00816) ;
(00817) ;
(00818) ;
(00819) ;
(00820) ;
(00821) ;
(00822) ;
(00823) ;
(00824) ;
(00825) ;
(00826) ;
(00827) ;
(00828) ;
(00829) ;
(00830) ;
(00831) ;
(00832) ;
(00833) ;
(00834) ;
(00835) ;
(00836) ;
(00837) ;
(00838) ;
(00839) ;
(00840) ;
(00841) ;
(00842) ;
(00843) ;
(00844) ;
(00845) ;
(00846) ;
(00847) ;
(00848) ;
(00849) ;
(00850) ;
(00851) ;
(00852) ;
(00853) ;
(00854) ;
(00855) ;
(00856) ;
(00857) ;
(00858) ;
(00859) ;
(00860) ;
(00861) ;
(00862) ;
(00863) ;
(00864) ;
(00865) ;
(00866) ;
(00867) ;
(00868) ;
(00869) ;
(00870) ;
(00871) ;
(00872) ;
(00873) ;
(00874) ;
(00875) ;
(00876) ;
(00877) ;
(00878) ;
(00879) ;
(00880) ;
(00881) ;
(00882) ;
(00883) ;
(00884) ;
(00885) ;
(00886) ;
(00887) ;
(00888) ;
(00889) ;
(00890) ;
(00891) ;
(00892) ;
(00893) ;
(00894) ;
(00895) ;
(00896) ;
(00897) ;
(00898) ;
(00899) ;
(00900) ;
(00901) ;
(00902) ;
(00903) ;
(00904) ;
(00905) ;
(00906) ;
(00907) ;
(00908) ;
(00909) ;
(00910) ;
(00911) ;
(00912) ;
(00913) ;
(00914) ;
(00915) ;
(00916) ;
(00917) ;
(00918) ;
(00919) ;
(00920) ;
(00921) ;
(00922) ;
(00923) ;
(00924) ;
(00925) ;
(00926) ;
(00927) ;
(00928) ;
(00929) ;
(00930) ;
(00931) ;
(00932) ;
(00933) ;
(00934) ;
(00935) ;
(00936) ;
(00937) ;
(00938) ;
(00939) ;
(00940) ;
(00941) ;
(00942) ;
(00943) ;
(00944) ;
(00945) ;
(00946) ;
(00947) ;
(00948) ;
(00949) ;
(00950) ;
(00951) ;
(00952) ;
(00953) ;
(00954) ;
(00955) ;
(00956) ;
(00957) ;
(00958) ;
(00959) ;
(00960) ;
(00961) ;
(00962) ;
(00963) ;
(00964) ;
(00965) ;
(00966) ;
(00967) ;
(00968) ;
(00969) ;
(00970) ;
(00971) ;
(00972) ;
(00973) ;
(00974) ;
(00975) ;
(00976) ;
(00977) ;
(00978) ;
(00979) ;
(00980) ;
(00981) ;
(00982) ;
(00983) ;
(00984) ;
(00985) ;
(00986) ;
(00987) ;
(00988) ;
(00989) ;
(00990) ;
(00991) ;
(00992) ;
(00993) ;
(00994) ;
(00995) ;
(00996) ;
(00997) ;
(00998) ;
(00999) ;

```

```

A08 04012 093A0500 (00370)
A09 04014 0A02A969 (00379)
A0A 04016 0C340C9B (00380)
A0B 04018 0E000000 (00381)
A0C 0401C 2E341C00 (00382)
A0D 04020 01300000 (00383)
A0E 04022 11300600 (00384)
A0F 04024 122C001C (00387)
A10 04026 142C0022 (00388)
A11 0402C 165A0001 (00389)
A12 04030 106A2F77 (00390)
A13 04032 1800001C (00391)
A14 04034 1042A747 (00392)
A15 04036 01304000 (00393)
A16 04038 105A0001 (00395)
A17 04040 20000022 (00397)
A18 04042 2147B191 (00398)
A19 04044 013A4000 (00399)
A20 04046 235A0001 (00401)
A21 04048 246A3390 (00402)
A22 0404C 26000015 (00403)
A23 04050 2742A070 (00404)
A24 04052 2A34A000 (00406)
A25 04054 000A0000 (00407)
A26 04056 2C9A0001 (00410)
A27 04058 2E4A2073 (00411)
A28 0405C 3082A073 (00413)
A29 04062 01302100 (00415)
A30 04064 379A0001 (00416)
A31 04066 34A0207D (00417)
A32 04068 00000000 (00418)
A33 0406C 3082A969 (00419)
A34 04070 01302100 (00420)
A35 04072 00000000 (00422)
A36 04074 00000000 (00423)
A37 04076 00000000 (00424)
A38 04078 00000000 (00425)

;SET DIRECTION MAP-TO-MAP
;DISPATCH TO PROPER BUFFER
;FILLER-UPPER
;INCR RPTR AND READ DATUM
;FILLED RMODEMA BUFFER?
;YES, SET SWITCH TO 0
;AND RPTR TO RMODEM
;AND INT LINE 2 TO SAY FULL
;SAME FOR RMODEM

;AND FOR RMODEM

;SET DIRECTION MAP-TO-MODEM
;PI SET MEANS USE TMODEM
;SEND WORD FROM TMODEM
;EMPTIED TMODEM?
;YES, RESET TPTR, FLIP P1,
;INT1 TO SAY BUFFER EMPTY
;SAME FOR TMODEM

;SIZE OF SCROLL PROGRAM IN HW
;DIR=FROM MAP (PRJR MEANINGLESS)
;SCHNLS=0 : AQ. MODE = DOUBLE. (ALSO)

```

PAGE 14: (R0ND3C0C16)B0N16U.H50.2, 29-Dec-00 15140123, Ed: WULF
 MODEN INTERFACE AND SYSTEM CLOCKS PROGRAMS

04072 0000	(00426)	DATA 0	0103 : 0102
04073 0000	(00427)	DATA 0,0	NO CONTROL REGISTERS
04074 0000			
04075 7777	(00428)	DATA -1,-1,-1,-1,-1	NULL OFFSET, BUFFER CHAIN ANCHORS
04076 7777			
04077 7777			
04078 7777			
04079 7777	(00429)		

```

PAGE 15: (MWD)CCH16>BBI6W.M50.2, 29-00C-00 15:44:23, PDI: WULF
1044 SINGLE-CHANNEL DOUBLE-BUFFERED SAMPLING

(00430) "ADAM SINGLE-CHANNEL DOUBLE-BUFFERED SAMPLING
(00431) J JY, 20 SEPT 79
(00432) J
(00433) J WE SAMPLE ON CHANNEL 1, WHICH IS CONNECTED TO THE G12 SPI.
(00434) J THIS PROGRAM IS FUNCTIONALLY IDENTICAL TO ADMS0, BUT BOUND AT
(00435) J ASSEMBLY TIME.
(00436) J
(00437) AD$END = (TADBA+SLNGTH-1) .AND. BITS15
(00438) AD$END = (TADBB+SLNGTH-1) .AND. BITS15
(00439) J
(00440) ADCNTRL = 2 J INTERNAL TRIGGER, EXTERNAL SAMPLING CLOCK,
(00441) J CH1 CONTROLLED BY SRI, SHORT FLT PT
(00442) J
(00443) J REGISTER AND FLAG USAGE:
(00444) J RB - CONTROLS MUX
(00445) J RI - DATA ADDRESS POINTER
(00446) J FI - THE ADAM USES FI TO TURN SAMPLING ON/OFF
(00447) J INT1 - SIGNALS END OF BUFFER
(00448) J
(00449) RL = $48A0 JBUS 1 ADR FOR ADPROC IOS MODULE
(00450) RA = 0
(00451) EVEN
(00452) DATA AD$SZ,0 JSCROLL PGM SIZE (RW), BIDS 110
(00453) J
(00454) AD$RCH: BEGIN IOS2(ADPROC)
(00455) J
PR JSET DIRECTION ADAM-TO-MAP
LOAD(R0,$0001) JINIT MUX TO CHNL 1
ADD(R0,0,TP)
SPI JSTART SAMPLING
AD$LOOP: LOAD(R1,TADBA-1(1),L) JINIT RPTR TO A/D BUFFER A
ADD(R0,$10,TP) JRELEASE S/H
SUB(R0,$10) JREAD SAMPLE, SEND TO BUFFER
JNE(R1,1,TH) JFILLED BUFFER?
JIFS(AD$STOP,SYNCSOP),INT1 JYES: INT1 TO SAY FULL, AND CHECK
JIF CSPO SAYS TO STOP
LOAD(R1,TADBB-1(1),L) JINIT RPTR TO A/D BUFFER B
ADD(R0,$10,TP)
SUB(R0,$10)
ADD(R1,1,TH)
JNE(R2,1,AD$END)
JIFC(AD$LOOP,SYNCSOP), INT1 JSTOP SAMPLING AND HALT
AD$STOP: CFI, STOP
END
(00474) J
(00475) J
(00476) AD$SZ = RL-AD$RCH
(00477) DATA 2
(00478) DATA $0101
(00479) DATA 0
(00480) DATA 3,0
JDIR = TO MAP
JCHNLS=1 : AQ. MODE = DOUBLE
JBI03 : BI02
JLOAD 3 CONTROL RECS STARTING WITH 0

```

PAGE 161 (00007)K0C116>R0M160.N50.2, 29-Dec-88 15:44:23, Ed: WOLF
ADAM SINGLE-CHANNEL DOUBLE-BUFFERED SAMPLING

04002 0000	(00401)	DATA 0	REG. 0: SAMPLE RATE 1 (FOR EXT CLK)
04003 0000	(00402)	DATA 0	REG. 1: SAMPLE RATE 2 NOT USED
04004 0000	(00403)	DATA ADCHNL	REG. 2: ADAM CONTROL BITS
04005 0002	(00404)	DATA -1,-1,-1,-1	BULL OFFSET, BUFFER CHAIN ANCHORS
04006 7777			
04007 7777			
04008 7777			
04009 7777			
0400A 7777	(00405)		

PAGE 18: (BBND)C0CA16>00M16W.M50.2, 29-Dec-88 15:49:23, Ed: VOL7
 AOM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT

04913 0000	(00535)	DATA 0	REG. 0: SAMPLE RATE
04914 0000	(00536)	DATA 0	REG. 1: NOT USED
04915 0002	(00537)	DATA DACCTRL	REG. 2: AOM CONTROL BITS
04916 FFFF	(00538)	DATA -1,-1,-1,-1,-1	NULL OFFSET, BUFFER CHAIN ANCHORS
04917 FFFF			
04918 FFFF			
04919 FFFF			
0491A FFFF			
	(00539)		
	(00540)		

(00541) -ADAMINT: ADAM INTERRUPT SERVICE ROUTINE

PAGE 26: (00001<DC16>000160-MS0-2, 29-Dec-88 15:44:23, Ed: WOLF
ADAMINT: ADAM INTERRUPT SERVICE ROUTINE

```

(00594) ; DATA. JUST COUNT THIS FRAME AS DISCARDED.
(00595) ; EVEN
(00596) ; AD$D1SC: INCH TADDC
(00597) ; NOP AD$RTM
(00598) ;
(00599) ; ROUTINES TO COPY FROM THE A/D BUFFER POINTED TO BY (R2)+1
(00600) ; TO A YSOURCE BUFFER. A SINGLE BMOVE WOULD SUFFICE, BUT THAT WOULD
(00601) ; HOLD OFF INTERRUPTS FOR ABOUT 250 USEC, SO WE BREAK IT UP INTO A FEW,
(00602) ; WHICH IS ALMOST AS FAST, BUT MAKES FOR BETTER INTERRUPT LATENCY.
(00603) ; THE REV-18 CSPD MICROCODE REVISION MODIFIED THE BMOVE R,L,MEN SO THAT
(00604) ; (1) R IS LEFT POINTING TO THE LAST SOURCE-WORD MOVED, AND
(00605) ; (2) X IS LEFT WITH ITS ORIGINAL CONTENTS. THIS LETS US USE
(00606) ; THE BMOVES AS DONE BELOW WITHOUT RESETTNG R AND X BEFORE EACH ONE.
(00607) ; EVEN
(00608) ; AD$CPY1: MOVIR R4,SL6-1
(00609) ; BMOVE R2,R4,TSRA
(00610) ; BMOVE R2,R4,TSRA+2*SL6
(00611) ; BMOVE R2,R4,TSRA+4*SL6
(00612) ; RETURN
(00613) ;
(00614) ; EVEN
(00615) ; AD$CPY0: MOVIR R4,SL6-1
(00616) ; BMOVE R2,R4,TSRB
(00617) ; BMOVE R2,R4,TSRB+2*SL6
(00618) ; BMOVE R2,R4,TSRB+4*SL6
(00619) ; RETURN
(00620) ;

```

(BMD) (C0A16) BMD16U.MSO.2, 29-Dec-80 15:44:23, Ed: WULF
 TMODEMINT: TMODEM INTERRUPT SERVICE ROUTINE

```

(00621) TMODEMINT: TMODEM INTERRUPT SERVICE ROUTINE
(00622) J JV, 3-OCT-79) MOD FOR DCA16, 11-SEP-80
(00623)
(00624) TMODEMINT COPIES PROTECTED AND BITSTREAMED DATA FROM A TBITS
(00625) BUFFER TO A TMODEM BUFFER. IF THE TBITS BUFFER FLAG SHOWS THAT
(00626) THE BUFFER ISN'T READY YET, A BUFFER OF "FAKE" DATA IS COPIED
(00627) FROM TBTC INSTEAD.
(00628)
00003000 BL = $3000          ;PUT THE REST OF THE CSPU CODE IN $3000-$3FFF.
(00630)
(00631) ; POINTER OFFSETS:
(00632) ; TBTPD IS FOR TBITS BUFFERS AND FLAGS (-2,0) INIT TO 0 SO IT
(00633) ; LOOKS AT TBTPD THE FIRST TIME)
(00634) ; TBTPD IS FOR TMODEM BUFFERS (-2,0) INIT TO 0 SO IT GETS SWITCHED
(00635) ; TO TMDMA THE FIRST TIME)
(00636)
(00637) ;POINTERS TO THE TBITS BUFFERS/FLAGS ARE FOUND IN THE PROTECT.MODULE.
(00638) ;POINTERS TO TMODEM-COPYING ROUTINES:
(00639) EVEN
(00640) TBTPD: ADDR TBTPD          ;TBITS BUFFER FLAGS
(00641) TBTPD: ADDR TBTPD
(00642)
(00643) ADDR TMTSCPYA          ;ROUTINE TO COPY INTO TMDMA
(00644) TMTD: ADDR TMTSCPYB      ;ROUTINE TO COPY INTO TMDMB
(00645)
(00646) EVEN
(00647) TMODEMINT:
(00648) MOVLM
(00649) MOVLM R2, TBTPD
(00650) XORIR R2, -2
(00651) MOVLM R2, TBTPD
(00652) CMPNZ RUN
(00653) JMP TMTSRN, EQZ
(00654) MOVLM R3, TBTPD
(00655) OTBTPTR(R3)
(00656) TMTSCPYA, EQZ
(00657) MOVLM R1, OTBTPTR(R3)
(00658) SUBIR R1, 1
(00659) CALL R6, OTBTPTR(R2)
(00660) MOVLM OTBTPTR(R3)
(00661) XORIR R3, -2
(00662) MOVLM R3, TBTPD
(00663) TMTSRN: MOVLM CLR+R2, SYS$FLGS ;MARK END OF TMODEMINT
(00664) RETURN
(00665)
(00666) ;TBITS(X) ISN'T FULL YET, AND WE HAVE TO SEND SOMETHING TO THE
(00667) ; MODEM, SO WE HAVE HANDY A BUFFER (TBTC) OF FAKE TBITS DATA
(00668) ; (CORRESPONDING TO SILENCE), WHICH WE SIMPLY USE INSTEAD.
(00669) EVEN
(00670) TMTSCPYA: INCH TMTFC
(00671) MOVLM R1, TBTC
(00672) CALL R6, OTBTPTR(R2)
(00673) NOP
(00674)
(00675) ;RETURN WITHOUT SWITCHING TBITS

```



```

(00705) 'RMODEMINT: RMODEM INTERRUPT SERVICE MODULE
(00706) ; JMW, 25 SEPT 79
(00707) ;
(00708) RMODEMINT CHECKS SYNC ON THE BUFFER JUST INPUT BY THE RMODEM SCROLL
(00709) ; PROGRAM, AND IF SYNC IS GOOD, COPIES THE LATEST "FRAME" OF DATA
(00710) ; FROM THE RMODEM BUFFERS TO A RMTS BUFFER. A "FRAME" OF DATA IS
(00711) ; HELD IN WORDS STARTING WITH A SYNC-BIT WORD. IN GENERAL, IT DOES
(00712) ; NOT START AND END ON RMODEM BUFFER BOUNDARIES.
(00713) ; THE STATE OF FRAME SYNC IS GIVEN BY RSYNC.
(00714) ; RSYNC=0 MEANS WE KNOW NOTHING ABOUT SYNC, SO WE MUST CALL SYNCINIT
(00715) ; TO INITIALIZE SYNC-SEARCHING.
(00716) ; RSYNC=-1 MEANS WE ARE STILL IN THE PROCESS OF SCANNING INCOMING
(00717) ; BUFFERS FOR FRAME SYNC (SYNCRCH).
(00718) ; RSYNC=+1 MEANS WE HAVE SYNC, SO WE CALL SYNCUPDATE TO CHECK THE NEW
(00719) ; FRAME FOR CONTINUED SYNC. IF SYNCUPDATE SETS RSYNC TO 0, THEN WE
(00720) ; HAVE LOST SYNC AND WE MUST REINITIALIZE WITH SYNCINIT.
(00721) RMODEMINT ALSO:
(00722) ; COPIES THE ON-HOOK BIT OF THE 1ST DATUM IN EACH RMODEM
(00723) ; BUFFER INTO RMTS FOR USE BY THE ADMININT MODULE,
(00724) ; SIMULATES CHANNEL ERRORS IF RERRS IS NONZERO, AND
(00725) ; CHECKS TO BE SURE THAT THE RMTS BUFFER IS AVAILABLE; IF NOT, THEN
(00726) ; THE NEW FRAME OF RMODEM DATA IS SIMPLY DISCARDED.
(00727) ;
(00728) ; POINTER OFFSETS:
(00729) ; RMTPO IS FOR THE RMTS BUFFERS AND FLAG (-2,0) INIT TO -2
(00730) ; SO THAT IT POINTS TO RMTA THE FIRST TIME)
(00731) ; RMPD IS FOR RMODEM BUFFERS (-4,-2,0) INIT TO 0 SO IT GETS SWITCHED
(00732) ; TO RMDNA THE FIRST TIME)
(00733) ;
(00734) ; POINTERS TO RMODEM BUFFERS AND RMTS FLAGS. (POINTERS TO
(00735) ; RMTS BUFFERS ARE IN THE CORPAR MODULE.)
(00736) ; USE RMTPTTR(RMTPO) TO POINT TO THE CURRENT RMTS BUFFER AND
(00737) ; RMTPTTR(RMTPO) TO POINT TO THE CURRENT FLAG.
(00738) ;
(00739) ;
(00740) ;
(00741) ;
(00742) ;
(00743) ;
(00744) ;
(00745) ;
(00746) ;
(00747) ; NOTE THAT RMODEMINT IS TO BE ENTERED BY CALL R0,RMODEMINT,
(00748) ; SO IT SIMPLY RETURNS. THERE WILL BE ANOTHER LEVEL THAT
(00749) ; IS DISPATCHED TO BY THE INTERRUPT, SUCH AS:
(00750) ; CALL R0,RMODEMINT
(00751) ; RET
(00752) ;
(00753) ;
(00754) RMODEMINT:
(00755) ;
(00756) ;
(00757) ;
(00758) ;
(00759) ;
(00760) ;
(00761) ;
(00762) ;
(00763) ;
(00764) ;
(00765) ;
(00766) ;
(00767) ;
(00768) ;
(00769) ;
(00770) ;
(00771) ;
(00772) ;
(00773) ;
(00774) ;
(00775) ;
(00776) ;
(00777) ;
(00778) ;
(00779) ;
(00780) ;
(00781) ;
(00782) ;
(00783) ;
(00784) ;
(00785) ;
(00786) ;
(00787) ;
(00788) ;
(00789) ;
(00790) ;
(00791) ;
(00792) ;
(00793) ;
(00794) ;
(00795) ;
(00796) ;
(00797) ;
(00798) ;
(00799) ;
(00800) ;
(00801) ;
(00802) ;
(00803) ;
(00804) ;
(00805) ;
(00806) ;
(00807) ;
(00808) ;
(00809) ;
(00810) ;
(00811) ;
(00812) ;
(00813) ;
(00814) ;
(00815) ;
(00816) ;
(00817) ;
(00818) ;
(00819) ;
(00820) ;
(00821) ;
(00822) ;
(00823) ;
(00824) ;
(00825) ;
(00826) ;
(00827) ;
(00828) ;
(00829) ;
(00830) ;
(00831) ;
(00832) ;
(00833) ;
(00834) ;
(00835) ;
(00836) ;
(00837) ;
(00838) ;
(00839) ;
(00840) ;
(00841) ;
(00842) ;
(00843) ;
(00844) ;
(00845) ;
(00846) ;
(00847) ;
(00848) ;
(00849) ;
(00850) ;
(00851) ;
(00852) ;
(00853) ;
(00854) ;
(00855) ;
(00856) ;
(00857) ;
(00858) ;
(00859) ;
(00860) ;
(00861) ;
(00862) ;
(00863) ;
(00864) ;
(00865) ;
(00866) ;
(00867) ;
(00868) ;
(00869) ;
(00870) ;
(00871) ;
(00872) ;
(00873) ;
(00874) ;
(00875) ;
(00876) ;
(00877) ;
(00878) ;
(00879) ;
(00880) ;
(00881) ;
(00882) ;
(00883) ;
(00884) ;
(00885) ;
(00886) ;
(00887) ;
(00888) ;
(00889) ;
(00890) ;
(00891) ;
(00892) ;
(00893) ;
(00894) ;
(00895) ;
(00896) ;
(00897) ;
(00898) ;
(00899) ;
(00900) ;
(00901) ;
(00902) ;
(00903) ;
(00904) ;
(00905) ;
(00906) ;
(00907) ;
(00908) ;
(00909) ;
(00910) ;
(00911) ;
(00912) ;
(00913) ;
(00914) ;
(00915) ;
(00916) ;
(00917) ;
(00918) ;
(00919) ;
(00920) ;
(00921) ;
(00922) ;
(00923) ;
(00924) ;
(00925) ;
(00926) ;
(00927) ;
(00928) ;
(00929) ;
(00930) ;
(00931) ;
(00932) ;
(00933) ;
(00934) ;
(00935) ;
(00936) ;
(00937) ;
(00938) ;
(00939) ;
(00940) ;
(00941) ;
(00942) ;
(00943) ;
(00944) ;
(00945) ;
(00946) ;
(00947) ;
(00948) ;
(00949) ;
(00950) ;
(00951) ;
(00952) ;
(00953) ;
(00954) ;
(00955) ;
(00956) ;
(00957) ;
(00958) ;
(00959) ;
(00960) ;
(00961) ;
(00962) ;
(00963) ;
(00964) ;
(00965) ;
(00966) ;
(00967) ;
(00968) ;
(00969) ;
(00970) ;
(00971) ;
(00972) ;
(00973) ;
(00974) ;
(00975) ;
(00976) ;
(00977) ;
(00978) ;
(00979) ;
(00980) ;
(00981) ;
(00982) ;
(00983) ;
(00984) ;
(00985) ;
(00986) ;
(00987) ;
(00988) ;
(00989) ;
(00990) ;
(00991) ;
(00992) ;
(00993) ;
(00994) ;
(00995) ;
(00996) ;
(00997) ;
(00998) ;
(00999) ;
(01000) ;

```

```

03067 1B20 (00758) SKPL LEZ
03068 9021FFFC (00759) MOVIR R2,-4
03069 E200546 (00760) MOVIR R2,RMPO
0306C E20053F (00761) CMPHZ RUN
0306E 001038C2 (00762) JMP RMISRM,EQZ
03070 90943060 (00763) MOVIR R1,0RMPTR(R2)
03072 70220000 (00764) MOVIR R3,R1,LOWSONRE
03074 E0300551 (00765) MOVIR R3,ROHMK
03076 F040057F (00766) MOVIR R4,REARS
03078 1A10 (00767) SKPL EQZ
03079 061038CA (00768) CALL R1,RMISSIM
0307B E2000552 (00769) CMPHZ RSYNC
0307D 0000 (00770) EVEN
0307E 001038B6 (00771) JMP RMISIMITY,EQZ
03080 003038C0 (00772) JMP RMISRMCH,LIZ
03082 06103C52 (00773) CALL R1,SYNCPDATE
03084 E2000552 (00774) CMPHZ RSYNC
03086 0010380A (00775) JMP RMISLOST,EQZ
03088 F0300547 (00776) MOVIR R3,RBTP0
0308A E2063B50 (00777) CMPHZ 0RMPTR(R3)
0308C 011038C6 (00778) JMP RMISDISC,NEZ
0308E 9094305E (00779) JCOPY FRAME OF BITS (INCLUDING SYNC BIT) FROM PREVIOUS AND CURRENT
03090 FC100553 (00780) RNDMEN BUFFERS TO RBITS BUFFER. WE CPT (MLENGTH-RB0FO) BITS
03092 2711 (00781) FROM (PREV. RNDMEN + RB0FO) TO RBITS, AND THEN (RB0FO) BITS FROM
03094 90C63E8C (00782) (CURRENT RNDMEN) TO (RBITS + MLENGTH-RB0FO).
03096 EF4030A0 (00783) NOTE THAT WE JIN ADDRESSES INTO TWO MOVE INSTRUCTIONS BELOW
03098 90500209 (00784) MOVIR R1,0RMPTR-2(R2)
0309A F5500553 (00785) ADDR R1,RB0FO
0309C 9C4A0001 (00786) DECR R1,1
0309E EF4038AA (00787) EVEN
030A0 051A0000 (00788) MOVIR R4,0RMPTR(R3)
030A2 F0500553 (00789) SAY R4,RMIS0M01
030A4 001038AC (00790) MOVIR R5,MLENGTH-1
030A6 90943060 (00791) SUBMR R5,RB0FO
030A8 051A0000 (00792) ADDR R4,1(R5)
030AA 051A0000 (00793) SAY R4,RMIS0M02
030AC 25063050 (00794) RMIS0M01: MOVE R1,R5,0
030AE 9431FFFE (00795) MOVIR R5,RB0FO
030B0 00300547 (00796) JMP RMISSTF,EQZ
030B2 05000547 (00797) MOVIR R1,0RMPTR(R2)
030B4 2711 (00798) DECR R1,1
030B6 051A0000 (00799) DECR R5,1
030B8 25063050 (00800) RMIS0M02: MOVE R1,R5,0
030BA 9431FFFE (00801) RMISSTF: INCH 0RMPTR(R3)
030BC 00300547 (00802) XORIR R3,-2
030BE 05000547 (00803) MOVIR R3,RBTP0
030C0 05000547 (00804) INCH RPCR
030C2 25000547 (00805) HOP
030C4 2000 (00806)
030C6 0000 (00807) EVEN
030C8 061038EA (00808) RMISIMITY: CALL R1,SYNCPINT
030CA 2009 (00809) HOP RMISRM
030CC 0000 (00810)

```

JDO WIL IF VOCODER NOT RUNNING
 JRI NOW POINTS TO LATEST RNDMEN BUFFER
 JGET UN-WOKE BIT FROM 1ST RNDMEN DATUM
 J AND SAVE IT FOR USE BY ADAMINT ROUTINE
 JREARS=0 OF CHANNEL ERRORS PER BUFFER
 J TO BE SIMULATED
 JWHAT DO WE KNOW ABOUT SYNC?
 J0 MEANS NIL, SO INIT
 J-1 MEANS WE LACK IT, SO SEARCH
 J+1 MEANS WE HAVE IT, SO UPDATE
 JDO WE STILL HAVE IT AFTER THE UPDATE?
 JNO, SO REINIT FOR SYNC-SEARCHING
 JRI GETS RBITS POINTER OFFSET
 JIS RBITS BUFFER AVAILABLE (EMPTY)?
 JNO, SO DISCARD THE FRAME OF DATA
 JCOPY FRAME OF BITS (INCLUDING SYNC BIT) FROM PREVIOUS AND CURRENT
 RNDMEN BUFFERS TO RBITS BUFFER. WE CPT (MLENGTH-RB0FO) BITS
 FROM (PREV. RNDMEN + RB0FO) TO RBITS, AND THEN (RB0FO) BITS FROM
 (CURRENT RNDMEN) TO (RBITS + MLENGTH-RB0FO).
 NOTE THAT WE JIN ADDRESSES INTO TWO MOVE INSTRUCTIONS BELOW
 MOVIR R1,0RMPTR-2(R2)
 ADDR R1,RB0FO
 DECR R1,1
 EVEN
 MOVIR R4,0RMPTR(R3)
 SAY R4,RMIS0M01
 MOVIR R5,MLENGTH-1
 SUBMR R5,RB0FO
 ADDR R4,1(R5)
 SAY R4,RMIS0M02
 RMIS0M01: MOVE R1,R5,0
 MOVIR R5,RB0FO
 JMP RMISSTF,EQZ
 MOVIR R1,0RMPTR(R2)
 DECR R1,1
 DECR R5,1
 RMIS0M02: MOVE R1,R5,0
 RMISSTF: INCH 0RMPTR(R3)
 XORIR R3,-2
 MOVIR R3,RBTP0
 INCH RPCR
 HOP
 RMISRM

PAGE 25: (RNDJ)DCJ16>RMI160.MSU-2, 29-Dec-88 15:44:23, Ed: WOLF
RMODEMINT: RMODEM INTERRUPT SERVICE MODULE

```

0309 0000 (00011) EVEN
030A 9094305E (00012) RMISLOS: MOVIR R1,ERRPTR-2(R2) ;WE HAVE LOST SYNC!
030B 061030E1 (00013) CALL R1,SYNCRNT ;CALL SYNCRNT ON THE OLDER RMODEM
030C 90943060 (00014) MOVIR R1,ERRPTR(R2) ;BUFFER, THEN SYNCRCH ON THE NEWER ONE
030D 06103C12 (00015) RMISRCH: CALL R1,SYNCRCH
030E 06103C12 (00016) RMISRTH: MOVIR CLR+C2,SYSSFLCS ;MARK END OF RMODEMINT
030F 0000FFCE (00017) RETURN
0310 0670 (00018) ;
0311 (00019) ;THE RMTS BUFFER ISN'T READY (EMPTY) YET, SO WE HAVE NO PLACE TO
0312 (00020) ; PUT THE RMODEM DATA, AND WE HAVE TO DISCARD IT.
0313 0000 (00021) EVEN
0314 0000E50054C (00022) RMISDISC: INCR RMPDC ;COUNT RMODEM FRAME DISCARD
0315 030C 2107 (00023) NOP RMISRTH
0316 (00024) ;
0317 (00025) ;ROUTINE TO SIMULATE CHANNEL ERRORS. WE USE THE EXEC CODE AS A "RANDOM"
0318 (00026) ; NUMBER GENERATOR. WE MAINTAIN A POINTER TO THAT PART OF BUS 1. FOR
0319 (00027) ; EACH ERROR, WE POLL IN 2 WORDS, XOR, AND OFF TO 9 BITS, +10, AND USE
0320 (00028) ; THAT AS AN OFFSET INTO THE CURRENT RMODEM BUFFER, WHERE WE FLIP THE
0321 (00029) ; DATA BIT. THUS WE CAN ERROR ANY OF THE LAST 512 BITS OF THE FRAME.
0322 (00030) ; DOING IT FIVE TIMES PER FRAME GIVES 0.968 ERRORS.
0323 (00031) ;UPON ENTRY, R1 POINTS TO THE START OF THE RMODEM BUFFER, AND R4 HOLDS
0324 (00032) ; THE NUMBER OF ERRORS TO BE MADE.
0325 0000 (00033) EVEN
0326 03CA 2741 (00034) RMISINC: DEC R4,1 ;NUMBER-1 OF ERRORS
0327 0000 (00035) EVEN
0328 005F30E6 (00036) MOVIR R5,ERRPTR ;PICK UP POINTER INTO EXEC CODE
0329 90700001 (00037) MOVIR R7,IDW80 ;WILL BE USED TO FLIP DATA BIT
0330 006A0000 (00038) R1: MOVIR R6,0(R5) ;GET "RANDOM" NUMBER
0331 07002 F46A9001 (00039) INCR R6,1(R5) ;MAKE IT MORE SO
0332 0304 2663 (00040) INCR R6,3 ;AVOID BIAS FOR 0
0333 0305 2653 (00041) INCR R5,3 ;STEP RANDOM NUMBER PTR
0334 0306 9A6001FF (00042) ANDIR R6,31FF ;MAKE THE RANDOM NUMBER 9 BITS
0335 0308 266A (00043) INCR R6,10 ;GIVE IT THE RANGE 10-521
0336 0309 4C62 (00044) ADDR R6,R1 ;MAKE PTR INTO RMODEM BUFFER
0337 030A F57C0000 (00045) XORR R7,0(R6) ;FLIP DATA BIT FOR THE ERROR
0338 030C 8C403000 (00046) DJP R4,R1 ;JED BACK IF NOT ENOUGH ERRORS YET
0339 030E 92501C77 (00047) CNPIR R5,RAND50 ;CHECK IF ERRPTR NEEDS RESETING
0340 030E 1A20 (00048) SKPL LE ;RESET ERRPTR TO LOWER VALUE
0341 030F 905000E8 (00049) MOVIR R5,RAND5L
0342 030E 205030E6 (00050) MOVIR R5,ERRPTR
0343 030E 0E70 (00051) RETURN
0344 030E 00E0 (00052) ERRPTR: DATA RAND5L
0345 (00053)

```

```

(00054) - FRAME SYNCHRONIZATION ROUTINES
(00055) )
(00056) ) MISC. INTERNAL VARIABLES
(00057) ) RESYNC: DATA 0
(00058) ) LASTERR: DATA 0
(00059) ) QLSYNC: DATA 0
(00060) ) H0P0: (1ST 01)
(00061) )
(00062) )
(00063) )
(00064) ) SYNCINIT: INITIALIZE BUFFERS, ETC. FOR SYNC SEARCHING. CALL WITH
(00065) ) R1 POINTING TO RMODEM BUFFER.
(00066) ) ALSO ZEROS THE GAIN WORD OF BOTH RESOURCE BUFFERS AND ZEROS THE
(00067) ) ENTIRE "LAST" RBITS BUFFER, SINCE IF WE JUST LOST SYNC, THOSE
(00068) ) BUFFERS CONTAIN GARBAGE, AND WE DON'T WANT TO RESUME SYNTHESIZING
(00069) ) WITH THEM.
(00070) ) EVEN
(00071) ) SYNCINIT: MOVIR R2,NBLNCTN-1
(00072) ) ADDR R1,R2
(00073) ) SLOOP: MOVIR R3,R1,1
(00074) ) MOVIR R3,ASSP(R2)
(00075) ) DECR R1,1
(00076) ) DJP R2,SLOOP
(00077) ) MOVIR R2,ASYN
(00078) ) RSR+RRESID+W$
(00079) ) RSR+RRESID+W$
(00080) ) MOVIR R3,-2
(00081) ) XORIR R3,-2
(00082) ) MOVIR R4,0RBTPT(R3)
(00083) ) MOVIR R4,R$
(00084) ) INCR R4,W$
(00085) ) SIF R4,S$BMD
(00086) ) DECR R4,2*W$
(00087) ) MOVIR R2,(NBLNCTN/2)-2
(00088) ) S$BMD: MOVIR R4,R2,0
(00089) ) MOVIR R4,RSSSS-W$
(00090) ) MOVIR R4,RSSSS-W$
(00091) ) MOVIR R4,RSSSS+W$
(00092) ) RETURN
(00093) )
(00094) )
(00095) ) SYNCRCN: SEARCH INCOMING RMODEM BUFFER FOR SYNC, AND IF SYNC
(00096) ) IS FOUND, SET RSYNC TO +1 AND SET RMOV TO THE FRAME OFFSET
(00097) ) CORRESPONDING TO THE LOGICAL START OF THE FRAME.
(00098) ) CALL WITH R1 POINTING TO THE START OF THE RMODEM BUFFER TO BE
(00099) ) SEARCHED.
(00100) ) REGISTER USAGE:
(00101) ) R1 - PTR TO THE RMODEM BUFFER BEING SEARCHED
(00102) ) R2 - OFFSET INTO THE RMODEM, RSSP, RSSSS BUFFERS
(00103) ) R3 - SCRATCH AC
(00104) ) R4 - MAXCNT, THE MAX. OF RSSSS(I)
(00105) ) R5 - MAX, THE # OF INSTANCES OF MAXCNT IN RSSSS(I)
(00106) ) R6 - IMAX, THE FRAME OFFSET WHERE MAXCNT WAS FOUND

```

15370YINVA (L0600) ; VARIANTS

ADMINISTRATIVE: ADM INTERRUPT SERVICE ROUTINE
(01004) (50010) : JJA, 3 OCT 79

PAGE 30: (88ND)XDCAL6>NM16U.MSO.2, 29-Dec-88 15:44:23, E4: WOLF
 ADJINT: ADM INTERRUPT SERVICE ROUTINE

03C87 0000	(01057) ;		
03C88 904F0023	(01050)	EVEN	
03C8A 07200E20	(01059)	DISCPTB: MOVIN	R4,SL6-1
03C8C 07200E20	(01060)	MOVEL	R2,R4,RDAB
03C8E 07200E20	(01061)	MOVEL	R2,R4,RDAB+2*SL6
03C90 07200E20	(01062)	MOVEL	R2,R4,RDAB+4*SL6
03C92 0720	(01063)	RETURN	
	(01064)		

PAGE 31: (PROMD) <DCAL16> 88M16U, MS0.2, 29-DEC-88 15:44:23, ED: VOLF
 PRORES(AB), PROPAR(AB) -- ERROR-PROTECT AND BITSTREAM THE INTR FRAME.
 PRORES(AB), PROPAR(AB) -- ERROR-PROTECT AND BITSTREAM THE INTR FRAME.

```

(01065) )PRORES(AB), PROPAR(AB) -- ERROR-PROTECT AND BITSTREAM THE INTR FRAME.
(01066) )
(01067) )PROTECT TAKES QUANTIZED AND CODED ANALYSIS PARAMETERS AND RESIDUAL
(01068) ) SAMPLES FROM A TSINK BUFFER, AND FORMS A BITSTREAM, INCLUDING
(01069) ) ERROR-PROTECTING CODES FOR SOME BITS OF CERTAIN PARAMETERS, IN
(01070) ) A TSINK BUFFER.
(01071) )THE "PROTECT" MODULE HAS BEEN BROKEN INTO 2 SHORTER ONES, WHICH
(01072) ) CAN BE (MOSTLY) HIDDEN UNDER APU MODULE EXECUTIONS.
(01073) ) PRORES(AB) -- BITSTREAMS THE RESIDUAL SAMPLES (ABOUT 3.7 MS)
(01074) ) PROPAR(AB) -- ERROR-PROTECTS AND BITSTREAMS THE OTHER PARAMETERS
(01075) ) (ABOUT 0.6 MS). PROPAR ALSO ACCUMULATES HISTOGRAM
(01076) ) DATA FOR THE G, DG, P, C1-C3, K1-K6 PARAMETERS.
(01077) )
(01078) )THESE ROUTINES ARE ENTERED WITH R1 POINTING TO THE PC00. 1(R1)
(01079) ) IS THE BUFFER/FLAG/ROUTINE POINTER OFFSET, -2 FOR "A" OR 0 FOR "B".
(01080) )
(01081) )FORMAT OF INPUT (TSINK) BUFFER IS (ONE DATUM PER HALF-WORD):
(01082) ) 216 RESIDUAL SAMPLES (0/2)
(01083) ) GAIN (6/6)
(01084) ) DG1 - DG3 (2/2)
(01085) ) PITCH (7/7)
(01086) ) C1 (2/3)
(01087) ) C2 (3/4)
(01088) ) C3 (2/3)
(01089) ) K1 (5/6)
(01090) ) K2 (4/5)
(01091) ) K3 (3/4)
(01092) ) K4-K6 (2/4)
(01093) )WHERE (M/N) MEANS M HIGH-ORDER BITS PROTECTED OUT OF N TOTAL FOR PARAM.
(01094) )
(01095) )
(01096) )FORMAT OF OUTPUT (BITS) BUFFER IS (ONE BIT PER HALF-WORD):
(01097) ) POSITION FOR THE SYNC BIT (IN THE TWOSEN BUFFER)
(01098) ) RES1 - RES216 (2 BITS EACH, LSO FIRST)
(01099) ) B5432 GAIN (7 BITS, NAMING (7,4) CODE)
(01100) ) B10 GAIN, B10 DG1 (7)
(01101) ) B10 DG2, B10 DG3 (7)
(01102) ) B6543 PITCH (7)
(01103) ) B0 C1 (1)
(01104) ) B2 C1, B210 PITCH (7)
(01105) ) B0 C2 (1)
(01106) ) B321 C2, B1 C1 (7)
(01107) ) B0 C3 (1)
(01108) ) B0 K1 (1)
(01109) ) B54 K1, B21 C3 (7)
(01110) ) B0 K2 (1)
(01111) ) B4321 K2 (7)
(01112) ) B0 K3 (1)
(01113) ) B3 K3, B321 K1 (7)
(01114) ) B32 K4, B21 K3 (7)
(01115) ) B10 K4 (2)
(01116) ) B10 K5 (2)
(01117) ) B10 K6 (2)

```


PAGE 34: (00001<DC116>00M16U-MS0.2, 29-Dec-80 15:44:23, ED: WOLF
PRORES(AB), PROPAR(AB) -- ERROR-PROTECT AND BITSTREAM THE INTR FRAME.

03036 301C	(01224)	POPXI	R1,R6	JC3 (2/3)
03037 0000	(01225)	EVEN		
03038 E50C9C64	(01226)	INCM	CXNIST(R6)	
03039 507C0001	(01227)	MOVR	R7,R6,B0	JEXTRACT B0
0303C 9670000C	(01228)	TORIR	R7,TMBITS	
0303E 342E	(01229)	PUSHXI	R2,R7	JOUTPUT B0 OF C3
0303F 3C61	(01230)	LRS	R6,1	JSHIFT B21 TO B10 POSITION
03040 301A	(01231)			
03041 0000	(01232)	POPXI	R1,R5	JK1 (5/6)
03042 E50A9C6C	(01233)	EVEN		
03043 507A0001	(01234)	INCM	KXNIST(R5)	
03044 507A0001	(01235)	MOVR	R7,R5,B0	JEXTRACT B0 AND OUTPUT IT
03046 9670000C	(01236)	TORIR	R7,TMBITS	
03048 342E	(01237)	PUSHXI	R2,R7	
03049 0000	(01238)	EVEN		
0304A 507A0030	(01239)	MOVR	R7,R5,B54	JEXTRACT B54 OF K1
0304C 3C72	(01240)	LRS	R7,2	JSHIFT IT INTO B32 POSITION
0304D 0000	(01241)	EVEN		
0304E 567C0003	(01242)	TORIR	R7,R6,B10	JADD IN THE SAVED B21 OF C3
03050 8660309C	(01243)	CALL	R6,OUT7	J AND OUTPUT THEN PROTECTED
03052 301C	(01244)			JRS STILL HOLDS B321 OF K1
03053 0000	(01245)	POPXI	R1,R6	JK2 (4/5)
03054 E50C9CAC	(01246)	EVEN		
03055 507C0001	(01247)	INCM	KXNIST(R6)	
03056 507C0001	(01248)	MOVR	R7,R6,B0	JEXTRACT B0 AND OUTPUT IT
03058 9670000C	(01249)	TORIR	R7,TMBITS	
0305A 342E	(01250)	PUSHXI	R2,R7	
0305B 3C61	(01251)	LRS	R6,1	JSHIFT B4321 INTO POSITION
0305C 407C	(01252)	MOVR	R7,R6	J AND OUTPUT PROTECTED
0305D 0000	(01253)	EVEN		
0305E 8660309C	(01254)	CALL	R6,OUT7	
03060 301C	(01255)			
03061 0000	(01256)	POPXI	R1,R6	JK3 (3/4)
03062 E50C9CCC	(01257)	EVEN		
03063 507C0001	(01258)	INCM	KXNIST(R6)	
03064 507C0001	(01259)	MOVR	R7,R6,B0	JEXTRACT L50 AND OUTPUT IT
03066 9670000C	(01260)	TORIR	R7,TMBITS	
03068 342E	(01261)	PUSHXI	R2,R7	
03069 0000	(01262)	EVEN		
0306A 507A000E	(01263)	MOVR	R7,R5,B321	JSET B321 OF K1
0306C 3C71	(01264)	LRS	R7,1	JSHIFT INTO POSITION
0306D 0000	(01265)	EVEN		
0306E 567C0000	(01266)	TORIR	R7,R6,B3	JADD IN THE B3 OF K3
03070 8660309C	(01267)	CALL	R6,OUT7	JOUTPUT B3 K3, B321 K1 PROTECTED
03072 3C61	(01268)	LRS	R6,1	JSHIFT B21 OF K3 INTO B10 POSITION
03073 301A	(01269)			
03074 507A000C	(01270)	POPXI	R1,R5	JFETCH K4 (2/4)
03075 507A000C	(01271)	EVEN		
03076 507A000C	(01272)	INCM	KXNIST(R5)	
03077 567C0003	(01273)	MOVR	R7,R5,B32	JEXTRACT B32 OF K4
03078 567C0003	(01274)	TORIR	R7,R6,B10	JADD THE B21 OF K3
0307A 8660309C	(01275)	CALL	R6,OUT7	JOUTPUT PROTECTED B32 K4, B21 K3
0307C 507A0003	(01276)	MOVR	R7,R5,B10	

0307E 86603DA8 (01277)	CALL R6,OUT2	OUTPUT 010 OF K4, UNPROTECTED
03080 301C (01278)	POPRT R1,R6	K5 (2/4)
03081 080A (01279)	INCH K5HIST(R5)	
03082 558C9C5C (01280)	MOVHR R7,R6,R10	
03084 507C0003 (01281)	CALL R6,OUT2	OUTPUT 010 OF K5, UNPROTECTED
03086 86603DA8 (01282)	POPRT R1,R5	K6 (2/4)
03088 301A (01283)	INCH K6HIST(R5)	
03089 080A (01284)	MOVHR R7,R5,R10	
0308A 558A9CFC (01285)	CALL R6,OUT2	OUTPUT 010 K6, UNPROTECTED
0308C 507A0003 (01286)	MOVHR R7,R6,B32	EXTRACT B32 OF K5
0308E 86603DA8 (01287)	LHS R7,2	SHIFT INTO B10 POSITION
03090 507C000C (01288)	EVEN	
03092 3C72 (01289)	TOKR R7,R5,B32	ADD THE B32 OF K6
03093 080A (01290)	CALL R6,OUT2	OUTPUT PROTECTED B32 K6, B32 K5
03094 567A000C (01291)	MOVLM CLRCL,SYSSFLGS	
03096 86603DA8 (01292)	RETURN	
03098 8E70 (01293)		
03099 080A (01294)		
0309A 8E70 (01295)		
0309B 080A (01296)		
0309C 8E703DA4 (01297)		
0309E 4C7E (01298)		
0309F 2627 (01299)		
030A0 90FE30B2 (01300)		
030A2 90600006 (01301)		
030A4 D57C0000 (01302)		
030A6 0F7A (01303)		
030A7 080A (01304)		
030A8 4C7E (01305)		
030A9 2622 (01306)		
030AA 90FF30D2 (01307)		
030AC C075F7FE (01308)		
030AE C075F7FE (01309)		
030B0 0E70 (01310)		
030B1 080A (01311)		
030B2 080A3E01 (01312)		
030B4 080A3DD9 (01313)		
030B6 080A30F3 (01314)		
030B8 080A30F0 (01315)		
030BA 080A30E2 (01316)		
030BC 080A30D8 (01317)		
030BE 080A30E1 (01318)		

OUT7 TAKES A 4-BIT DATUM IN R7 AND OUTPUTS A 7-BIT HAMMING CODE
 NOTE THAT WE MODIFY THE INSTRUCTION AT B007 BY JAMMING THE
 17-BIT OUTPUT POINTER INTO ITS ADDRESS FIELD.
 EVEN
 OUT7: SAV R2,B007
 ADDR R7,R7
 INCR R2,7
 MOVHR R7,B007(R7)
 MOVHR R6,7-1
 MOVHR R7,R6,8
 RETURN
 B007: MOVHR R7,R6,8
 RETURN
 OUT2 OUTPUTS 2 BITS GIVEN IN R7
 EVEN
 OUT2: ADDR R7,R7
 INCR R2,2
 MOVHR R7,B007(R7)
 POPRT R7,-2(R2)
 POPRT R7,-1(R2)
 RETURN
)TABLE OF NAMING CODE SEQUENCES. INDEX WITH 2*(4-BIT VALUE)
) TO BE PROTECTED) TABLE ENTRY IS PTR-1 TO THE 7-BIT NAMING
) CODEWORD.
 EVEN
 ADDR M000-1
 ADDR M151-1
 ADDR M052-1
 ADDR M103-1
 ADDR M114-1
 ADDR M045-1
 ADDR M146-1
 HPTR:
 ADDR M000-1
 ADDR M151-1
 ADDR M052-1
 ADDR M103-1
 ADDR M114-1
 ADDR M045-1
 ADDR M146-1

I

10

```

(01395) 'COMPAR(AB): UNBITSTREAM, ERROR-CORRECT, AND DECODE PARAMETERS
(01396) )
(01397) )COMPAR TAKES A FRAME OF BITSTREAM (BITS BUFFER) FROM THE
(01398) ) RECEIVER MODEN AND TURNS IT INTO PARAMETERS,
(01399) ) DOING ERROR-CORRECTING DECODING FOR THE PROTECTED DATA BITS. EACH
(01400) ) UNBITSTREAMED AND CORRECTED VALUE IS THEN LOOKED UP IN THE
(01401) ) APPROPRIATE DECODING TABLE, SO THAT THE OUTPUT (RESOURCE) BUFFER IS
(01402) ) FULL-WORD FLOATING POINT VALUES READY FOR THE SYNTHESIS PROCESS.
(01403) ) COMPAR DOES NOT TOUCH THE MISTOOL SAMPLES; THEY GET UNBITSTREAMED
(01404) ) AND DECODED BY THE AP MODULES DECODA/DECOB.
(01405) )COMPAR NO LONGER RUNS AT INTERRUPT LEVEL, BUT IS INVOKED VIA FCB.
(01406) )
(01407) )THIS ROUTINE IS ENTERED (VIA FCB) WITH R1 POINTING TO THE
(01408) ) FCB. 1(R1) IS THE BUFFER/FLAG POINTER OFFSET, EQUAL TO -2 FOR "A"
(01409) ) OR 0 FOR "B".
(01410) )REGISTER USAGE:
(01411) ) R1 = PTR-1 TO THE INPUT (BITS) BUFFER (USED BY POPKI'S)
(01412) ) R2 = PTR TO THE OUTPUT (RESOURCE) BUFFER (USED IN PUSHMIL'S)
(01413) ) R3-R5 = SCRATCH IN PARAMETER DECODING
(01414) ) R6 - SCRATCH IN PARAM DECODING AND IN GET74, ETC.
(01415) ) R7 - GET74, ETC. RETURN VALUES IN HERE
(01416) )
(01417) )THE FORMAT OF THE INPUT (BITS) BUFFER IS IDENTICAL TO THE BITS
(01418) ) BUFFER, WHICH IS DESCRIBED ABOVE IN THE PROSES/PROPAR DOCUMENTATION.
(01419) ) THE FORMAT OF THE OUTPUT (RESOURCE) BUFFER IS LIKE THAT OF THE
(01420) ) TSINK BUFFER (ALSO DESCRIBED ABOVE), EXCEPT THAT EACH DATUM OCCUPIES
(01421) ) A FULL-WORD INSTEAD OF A HALF-WORD, BECAUSE THESE ARE DECODED
(01422) ) FLOATING POINT VALUES.
(01423) )
(01424) )POINTERS TO BUFFERS. THE BITS POINTERS ARE ALSO USED BY RHODEMINT.
(01425) ) EVEN
(01426) ) ADDR RSTA
(01427) ) RSTPTR: ADDR RSTB
(01428) )
(01429) ) ADDR RSHA+RESID*W$ RRESOURCE BUFFER PTRS TO PARAMETERS-
(01430) ) RSBPTR: ADDR RSBH+RESID*W$ )(STEPPING OVER THE RESIDUAL)
(01431) )
(01432) )
(01433) ) EVEN
(01434) ) COPPAR: MOVLM
(01435) ) WVMR
(01436) ) MOVIR
(01437) )
(01438) ) ADDR
(01439) ) MOVIR
(01440) ) CALL
(01441) ) LLS
(01442) ) MOVIR
(01443) )
(01444) )
(01445) ) CALL
(01446) ) MOVIR
(01447) ) LRS
(01448) ) TORR
(01449) )
(01450) )
(01451) )
(01452) )
(01453) )
(01454) )
(01455) )
(01456) )
(01457) )
(01458) )
(01459) )
(01460) )
(01461) )
(01462) )
(01463) )
(01464) )
(01465) )
(01466) )
(01467) )
(01468) )
(01469) )
(01470) )
(01471) )
(01472) )
(01473) )
(01474) )
(01475) )
(01476) )
(01477) )
(01478) )
(01479) )
(01480) )
(01481) )
(01482) )
(01483) )
(01484) )
(01485) )
(01486) )
(01487) )
(01488) )
(01489) )
(01490) )
(01491) )
(01492) )
(01493) )
(01494) )
(01495) )
(01496) )
(01497) )
(01498) )
(01499) )
(01500) )

```

PAGE 391

(BOUND)C0CA1G00016U.N50.2, 29-DEC-68 15:44:23, FDI WOLF
CORPAR(AB): UNLISTED, ERROR-CORRECT, AND DECODE PARAMETERS

03E26 C62A9070 (01440)	PUSHMIL R2,DVLC(R5)	DECODE AND STORE GAIN
03E28 C6209070 (01449)	PUSHMIL R2,DVLC(R4)	DECODE AND STORE DCI
03E2A 06003E20 (01451)	CALL R0,GET74	GET DC2, DC3
03E2C 506F0006 (01452)	MOVX R6,R7,C10	SAVE DC3 IN R6
03E2E 3C72 (01453)	LS R7,2	SHIFT DC2 INTO POSITION
03E2F C62F9070 (01454)	PUSHMIL R2,DVLC(R7)	DECODE AND STORE DC2
03E31 C62C9070 (01455)	PUSHMIL R2,DVLC(R6)	DECODE AND STORE DC3
03E33 06003E20 (01457)	CALL R0,GET74	GET NI 4 BITS OF PITCH
03E35 3A73 (01458)	LLS R7,3	SHIFT INTO FINAL POSITION
03E36 405F (01459)	MOVX R5,R7	AND SAVE IN R5
03E37 06003E20 (01460)	CALL R0,GET1	GET R0 C1
03E39 404F (01461)	MOVX R4,R7	SAVE IT IN R4
03E3A 06003E20 (01462)	CALL R0,GET74	GET R2 C1, LO 3 BITS OF PITCH
03E3C 505F000E (01463)	TORX R5,R7,C210	MOVE LO 3 OF PITCH IN WITH NI 4
03E3E 9C50001C (01464)	ADDR R5,142	ADD 14 TO "DECODE" (LSHIFTED 1, SO 2*)
03E40 3A56 (01465)	LLS R5,7-1	MAKE IT INTO UNIFORM FLT PT IN R5,R6
03E41 90600042 (01466)	MOVX R6,42	RESPOND
03E43 06540000 (01467)	MOVX R5,0(R2)	STORE R5,R6
03E45 2022 (01468)	TORX R2,2	STEP OUTPUT POINTER
03E46 3C71 (01469)	LS R7,1	SHIFT R2 OF C1 INTO R2 POSITION
03E47 504F0000 (01470)	TORX R4,R7,C2	R4 NOW HAS R2, R6 OF C1
03E49 06003E20 (01471)	CALL R0,GET1	GET R0 OF C2
03E4B 403F (01472)	MOVX R3,R7	SAVE IT IN R3
03E4C 06003E20 (01473)	CALL R0,GET74	GET R221 OF C2, R1 OF C1
03E4E 503F001C (01475)	TORX R3,R7,C321	ADD R221 OF C2 TO R0
03E50 4C7F (01477)	ADDR R7,R7	LEFTSHIFT TO POSITION R1 OF C1
03E51 504F0004 (01478)	TORX R4,R7,C1	OR IT IN WITH R2, R6 OF C1
03E53 C6209A00 (01479)	PUSHMIL R2,DVLC(R4)	DECODE AND STORE C1
03E55 C6269A10 (01480)	PUSHMIL R2,DVLC(R3)	DECODE AND STORE C2
03E57 06003E20 (01481)	CALL R0,GET1	GET R0 OF C3
03E59 405F (01482)	MOVX R5,R7	SAVE IT IN R5
03E5A 06003E20 (01483)	CALL R0,GET1	GET R0 K1
03E5C 404E (01484)	MOVX R4,R7	SAVE IT IN R4
03E5D 06003E20 (01485)	CALL R0,GET74	GET R54 K1, R21 C3
03E5F 4C7E (01486)	ADDR R7,R7	LEFTSHIFT TO POSITION R21 OF C3
03E60 505F000C (01487)	TORX R5,R7,C21	PUT R21 C3 IN WITH R0
03E62 C62A9A00 (01488)	PUSHMIL R2,DVLC(R5)	DECODE AND STORE C3
03E64 4C7E (01489)	ADDR R7,R7	LEFTSHIFT AGAIN TO POSITION R54 OF K1
03E65 504F0000 (01490)	TORX R4,R7,C54	SAVE THEM IN R4 WITH R0
03E67 06003E20 (01491)	CALL R0,GET1	GET R0 R2
03E69 405F (01492)	MOVX R5,R7	STASH IT IN R5
03E6A 06003E20 (01493)	CALL R0,GET74	GET 4 MSB'S OF R2
03E6C 4C7E (01494)	ADDR R7,R7	LEFT SHIFT 1 TO POSITION THEM
03E6D 405E (01495)	TORX R5,R7	ADD R4321 K2 TO R0 IN R5

[illegible]

03E0B 000E	(01607)	DATA 0'03'02	JRCVD 003
03E0C 0000	(01608)	DATA 0'00'02	JRCVD 004
03E0D 000A	(01609)	DATA 0'05'02	JRCVD 005
03E0E 001C	(01610)	DATA 0'16'02	JRCVD 006
03E0F 000E	(01611)	DATA 0'07'02	JRCVD 007
03E10 0000	(01612)	DATA 0'00'02	JRCVD 010
03E11 0012	(01613)	DATA 0'11'02	JRCVD 011
03E12 0004	(01614)	DATA 0'02'02	JRCVD 012
03E13 000E	(01615)	DATA 0'07'02	JRCVD 013
03E14 0000	(01616)	DATA 0'04'02	JRCVD 014
03E15 000F	(01617)	DATA 0'07'02	JRCVD 015
03E16 000E	(01618)	DATA 0'07'02	JRCVD 016
03E17 000E	(01619)	DATA 0'07'02	JRCVD 017
03E18 000F	(01620)	DATA 0'00'02	JRCVD 020
03E19 0012	(01621)	DATA 0'11'02	JRCVD 021
03E1A 001C	(01622)	DATA 0'16'02	JRCVD 022
03E1B 0016	(01623)	DATA 0'13'02	JRCVD 023
03E1C 001C	(01624)	DATA 0'16'02	JRCVD 024
03E1D 001A	(01625)	DATA 0'15'02	JRCVD 025
03E1E 001C	(01626)	DATA 0'16'02	JRCVD 026
03E1F 001C	(01627)	DATA 0'16'02	JRCVD 027
03E20 0012	(01628)	DATA 0'11'02	JRCVD 030
03E21 0012	(01629)	DATA 0'11'02	JRCVD 031
03E22 0014	(01630)	DATA 0'12'02	JRCVD 032
03E23 0012	(01631)	DATA 0'11'02	JRCVD 033
03E24 0010	(01632)	DATA 0'14'02	JRCVD 034
03E25 0012	(01633)	DATA 0'11'02	JRCVD 035
03E26 001C	(01634)	DATA 0'16'02	JRCVD 036
03E27 000F	(01635)	DATA 0'07'02	JRCVD 037
03E28 0000	(01636)	DATA 0'00'02	JRCVD 040
03E29 000A	(01637)	DATA 0'05'02	JRCVD 041
03E2A 0004	(01638)	DATA 0'02'02	JRCVD 042
03E2B 0016	(01639)	DATA 0'13'02	JRCVD 043
03E2C 000A	(01640)	DATA 0'05'02	JRCVD 044
03E2D 000A	(01641)	DATA 0'05'02	JRCVD 045
03E2E 000C	(01642)	DATA 0'06'02	JRCVD 046
03E2F 000A	(01643)	DATA 0'05'02	JRCVD 047
03E30 0004	(01644)	DATA 0'02'02	JRCVD 050
03E31 0002	(01645)	DATA 0'01'02	JRCVD 051
03E32 0004	(01646)	DATA 0'02'02	JRCVD 052
03E33 0004	(01647)	DATA 0'02'02	JRCVD 053
03E34 0010	(01648)	DATA 0'14'02	JRCVD 054
03E35 000A	(01649)	DATA 0'05'02	JRCVD 055
03E36 0004	(01650)	DATA 0'02'02	JRCVD 056
03E37 000E	(01651)	DATA 0'07'02	JRCVD 057
03E38 0010	(01652)	DATA 0'10'02	JRCVD 060
03E39 0016	(01653)	DATA 0'13'02	JRCVD 061
03E3A 0016	(01654)	DATA 0'13'02	JRCVD 062
03E3B 0016	(01655)	DATA 0'13'02	JRCVD 063
03E3C 0010	(01656)	DATA 0'14'02	JRCVD 064
03E3D 000A	(01657)	DATA 0'05'02	JRCVD 065
03E3E 001C	(01658)	DATA 0'16'02	JRCVD 066
03E3F 0016	(01659)	DATA 0'13'02	JRCVD 067

03F10 0018	(01660)	DATA 0'14'02	JRCVD 070
03F11 0012	(01661)	DATA 0'11'02	JRCVD 071
03F12 0004	(01662)	DATA 0'02'02	JRCVD 072
03F13 0016	(01663)	DATA 0'13'02	JRCVD 073
03F14 0010	(01664)	DATA 0'14'02	JRCVD 074
03F15 0018	(01665)	DATA 0'14'02	JRCVD 075
03F16 0010	(01666)	DATA 0'14'02	JRCVD 076
03F17 0012	(01667)	DATA 0'17'02	JRCVD 077
03F18 0008	(01668)	DATA 0'06'02	JRCVD 100
03F19 0006	(01669)	DATA 0'03'02	JRCVD 101
03F1A 0006	(01670)	DATA 0'03'02	JRCVD 102
03F1B 0006	(01671)	DATA 0'03'02	JRCVD 103
03F1C 0008	(01672)	DATA 0'04'02	JRCVD 104
03F1D 001A	(01673)	DATA 0'15'02	JRCVD 105
03F1E 000C	(01674)	DATA 0'06'02	JRCVD 106
03F1F 0006	(01675)	DATA 0'03'02	JRCVD 107
03F20 0008	(01676)	DATA 0'04'02	JRCVD 110
03F21 0002	(01677)	DATA 0'01'02	JRCVD 111
03F22 0014	(01678)	DATA 0'12'02	JRCVD 112
03F23 0006	(01679)	DATA 0'03'02	JRCVD 113
03F24 0008	(01680)	DATA 0'04'02	JRCVD 114
03F25 0008	(01681)	DATA 0'04'02	JRCVD 115
03F26 0008	(01682)	DATA 0'04'02	JRCVD 116
03F27 000E	(01683)	DATA 0'07'02	JRCVD 117
03F28 0010	(01684)	DATA 0'10'02	JRCVD 120
03F29 001A	(01685)	DATA 0'15'02	JRCVD 121
03F2A 0014	(01686)	DATA 0'12'02	JRCVD 122
03F2B 0006	(01687)	DATA 0'03'02	JRCVD 123
03F2C 001A	(01688)	DATA 0'15'02	JRCVD 124
03F2D 001A	(01689)	DATA 0'15'02	JRCVD 125
03F2E 001C	(01690)	DATA 0'16'02	JRCVD 126
03F2F 001A	(01691)	DATA 0'15'02	JRCVD 127
03F30 0014	(01692)	DATA 0'12'02	JRCVD 130
03F31 0012	(01693)	DATA 0'11'02	JRCVD 131
03F32 0014	(01694)	DATA 0'12'02	JRCVD 132
03F33 0014	(01695)	DATA 0'12'02	JRCVD 133
03F34 0008	(01696)	DATA 0'04'02	JRCVD 134
03F35 001A	(01697)	DATA 0'15'02	JRCVD 135
03F36 0014	(01698)	DATA 0'12'02	JRCVD 136
03F37 001E	(01699)	DATA 0'17'02	JRCVD 137
03F38 0010	(01700)	DATA 0'10'02	JRCVD 140
03F39 0002	(01701)	DATA 0'01'02	JRCVD 141
03F3A 000C	(01702)	DATA 0'06'02	JRCVD 142
03F3B 0006	(01703)	DATA 0'03'02	JRCVD 143
03F3C 000C	(01704)	DATA 0'06'02	JRCVD 144
03F3D 000A	(01705)	DATA 0'05'02	JRCVD 145
03F3E 000C	(01706)	DATA 0'06'02	JRCVD 146
03F3F 000C	(01707)	DATA 0'06'02	JRCVD 147
03F40 0002	(01708)	DATA 0'01'02	JRCVD 150
03F41 0002	(01709)	DATA 0'01'02	JRCVD 151
03F42 0004	(01710)	DATA 0'02'02	JRCVD 152
03F43 0002	(01711)	DATA 0'01'02	JRCVD 153
03F44 000A	(01712)	DATA 0'04'02	JRCVD 154

PAGE 44: (CONDICOCAL16>DDH16U.H50.2, 29-Dec-88 15:44:23, Ed: VULF
 COMPAR(16): UNBITSTREAM, ZERO8-CORRECT, AND DECODE PARAMETERS

03F45 0002	(01713)	DATA 0'01'02	JACVD 155
03F46 000C	(01714)	DATA 0'06'02	JACVD 156
03F47 001F	(01715)	DATA 0'17'02	JACVD 157
03F48 0010	(01716)	DATA 0'10'02	JACVD 160
03F49 0010	(01717)	DATA 0'10'02	JACVD 161
03F4A 0010	(01718)	DATA 0'10'02	JACVD 162
03F4B 0016	(01719)	DATA 0'13'02	JACVD 163
03F4C 0010	(01720)	DATA 0'10'02	JACVD 164
03F4D 001A	(01721)	DATA 0'15'02	JACVD 165
03F4E 000C	(01722)	DATA 0'06'02	JACVD 166
03F4F 001F	(01723)	DATA 0'17'02	JACVD 167
03F50 001A	(01724)	DATA 0'10'02	JACVD 170
03F51 0002	(01725)	DATA 0'01'02	JACVD 171
03F52 0014	(01726)	DATA 0'12'02	JACVD 172
03F53 001E	(01727)	DATA 0'17'02	JACVD 173
03F54 0010	(01728)	DATA 0'14'02	JACVD 174
03F55 001E	(01729)	DATA 0'17'02	JACVD 175
03F56 001F	(01730)	DATA 0'17'02	JACVD 176
03F57 001F	(01731)	DATA 0'17'02	JACVD 177
	(01732)		

PAGE 45: (R0001)OCAL16>R0016U-M50.2, 29-Dec-88 15:44:23, 2di WOLF
APU3-DECODE()

```

(01733) APU3-DECODE()          DECODE RESIDUAL
(01734) J. WOLF, 10/21/86
(01735)
(01736) BINDS TO APU3-DECODE OR APU3-DECODE
(01737)
(01738) UNBITSTREAMS AND DECODES THE RECEIVED RESIDUAL BITS FROM AN "RBITSM"
(01739) BUFFER (RDATA OR RBIT0) TO AN "RRESOURCE" BUFFER (RSRA OR RSRR).
(01740) RSRR(N) = DVLUP(RBITX(N+2),RBITX(N+1))
(01741) WE PULL IN 2 HALFWORDS FROM THE RBITX BUFFER, THE FIRST OF WHICH HAS
(01742) THE RESIDUAL LSB IN ITS RIGHTMOST BIT (R16) AND THE SECOND OF WHICH
(01743) HAS THE MSB. WE SHIFT THOSE BITS TO BIT 7, THEN XLSBT THEM
(01744) INTO T1 AND T2 RESPECTIVELY, THEN DO LOGIC ON THE T-BITS TO
(01745) SELECT THE PROPER OUTPUT VALUE FROM THE 4-LONG DVLUP TABLE.
(01746) DVLUP IS THE (LENGTH=4) DECODING TABLE FOR THE 2-BIT RESIDUAL DATA,
(01747) WHICH USES THE FOLDED BINARY CODE.
(01748)
(01749) INPUT SEQUENCE:
(01750) (2**9), DVLUP(0),...,DVLUP(3),
(01751) RBITX(1), RBITX(2), ..., RBITX(2*RESID), (T1)
(01752) OUTPUT SEQUENCE:
(01753) RSRR(0), RSRR(1), ..., RSRR(MRESID-1), (T0)
(01754)
(01755) JOURNATION:
(01756) -10 APU CYCLES = -1.0 USEC PER RESIDUAL SAMPLE
(01757) 3 NEW REFS = -1.5 USEC (BUS 1) PER RESIDUAL SAMPLE
(01758)
(01759) APU REGISTER USAGE:
(01760) LEFT RIGHT
(01761) A0 DVLUP(0) -
(01762) A1 DVLUP(1) -
(01763) A2 DVLUP(2) -
(01764) A3 DVLUP(3) -
(01765) A4 SHIFTED LSB
(01766) M0: 2**9 SHIFTED MSB
(01767) M4: LSB WORD MSB WORD
(01768)
(01769) EVEN
(01770) DATA DCDUSSA JAPU START ADDRESS
(01771) DATA DCDUSSZ JAPU MODULE SIZE
(01772)
(01773) DCDU: BEGIN APU(DCDU)
(01774) DCDUSSA: SET(UN)
(01775) MOV(IQA,M0)
(01776) MOV(IQA,M0) \ NOP
(01777) MOV(IQA,M1) \ NOP
(01778) MOV(IQA,M2) \ NOP
(01779) MOV(IQA,M3) \ NOP
(01780) MOV(IQA,M4) \ NOP
(01781) MOV(IQA,M4) \ NOP
(01782) MUL(M0,M4)
(01783) MOV(P,M4)
(01784) JMAIN LOOP
(01785) B1: ALIGN(A4)
(01786)
(01787) JBY MULT. BY (2**9) AND THEN ALIGNING.

```

PAGE 46: [M80D1<DC116>80M16U.M50.2, 29-Dec-88 15:44:23, Ed: WOLF
 1P03-DECODE()] DECODE RESIDUAL

```

A0B 03F70 00E00000 (01786)      MOV(TQA,M4) \ NOP
A0C 03F72 1A9A1194 (01787)      MOV(A4),XLSBT(A4)
A0D 03F74 000000EC (01790)      NOP \ MOV(TQA,M4)
A0E 03F76 04000400 (01789)      MUL(M0,M4)
A0F 03F78 00000000 (01790)      MOV(R, NULL)
A10 03F7A 911F0016 (01791)      JUMPS(DCD$1X,T2)
A11 03F7C 911E0014 (01792)      DCD$0X: JUMPS(DCD$01,T1)
A12 03F7E 02000000 (01793)      DCD$00: R(A0) \ NOP
A13 03F80 1000001A (01794)      JUMP(DCD$OUT)
A14 03F82 02200000 (01795)      ,
A15 03F84 1000001A (01797)      DCD$01: R(A1) \ NOP
A16 03F86 911E0019 (01799)      JUMPS(DCD$11,T1)
A17 03F88 02400000 (01800)      DCD$10: R(A2) \ NOP
A18 03F8A 1000001A (01801)      JUMP(DCD$OUT)
A19 03F8C 02600000 (01802)      ,
A1A 03F8E 00A400B4 (01804)      DCD$11: R(A3) \ NOP
A1B 03F90 00900000 (01805)      DCD$00: MOV(P,A4)
A1C 03F92 901C000A (01806)      MOV(R,00) \ NOP
A1D 03F94 20320032 (01807)      JUMPC(R1,ED)
A1E 03F96 10000000 (01808)      CLEAR(RA)
A1F 03F98 0000001F (01809)      JUMP(0)
A20 03F9A 0000001F (01810)      DCDU$SZ = 1A - DCDU$SA
A21 03F9C 0000001F (01811)      END
  
```

```

) (PICK UP THE NEXT LSB)
) (MOVE THE DATA BITS INTO T1, T2)
) (PICK UP THE NEXT MSB)
) (START THE NEXT MULTIPLY)
) (WAIT FOR XLSBT TO FINISH)
) (JUMP IF MSB=1)
) (MSB=0) JUMP IF LSB=1
) (DATUM=00) RETURN DVALUF(0)

) (DATUM=01: DVALUF(1))

) (MSB=1) JUMP IF LSB=1
) (DATUM=10: DVALUF(2))

) (DATUM=11: DVALUF(3))

) (OUTPUT DECODED VALUE)
) (LOOP IF WE'VE NOT FINISHED OUTPUT)
  
```

PAGE 47: (0000) <0C116> 29-DEC-88 15:44:23, Ed: WOLF
APS PROGRAM FOR DECODA

```

(01011) APS3-DECODA() APS PROGRAM FOR DECODA
(01012)
(01013) J. WOLF, 10/21/88
(01014)
(01015) BINDS TO APS3-DECOD
(01016)
(01017) APS PROGRAM FOR DECODING THE RESIDUAL FOR "A" BUFFERS (RSTA TO RSRA).
(01018) BECAUSE WE CAN'T SPARE BID'S FOR THE RSTA BUFFERS, WE HAVE 2
(01019) VERSIONS OF THE DECODE MODULE, ONE WITH (RSTA,RSRA) "HARD BOUND"
(01020) IN, AND ONE WITH (RSTB,RSRB).
(01021)
(01022) HEADER BLOCK
(01023)
(01024) EVEN
(01025) ADDR 0
(01026) ADDR 0
(01027) DATA 0
(01028) DATA DCDS$Z
(01029) ADDR 0
(01030) EVEN
(01031)
(01032) INPUT PROGRAM:
(01033) BR0: RSTA(M)
(01034) BR1: LOOP COUNTER (STARTS AT HRESID-1)
(01035)
(01036) DCDS$Z
(01037) DCDS$Z: BEGIN APS(DCDSA)
(01038) SET(R0)
(01039) LOAD(BR0,TWOH9(1),TF)
(01040) LOAD(BR0,DVLUF(1),TF)
(01041) ADD(BR0,$S,TF)
(01042) ADD(BR0,$S,TF)
(01043) ADD(BR0,$S,TF)
(01044) ADD(BR0,$S,TF)
(01045)
(01046) LOAD(BR0,RSTA(1))
(01047) LOAD(BR1,HRESID-1)
(01048) JMAIN LOOP
(01049) BR1:
(01050) ADD(BR0,$S,TF)
(01051) ADD(BR0,$S,TF)
(01052) SUBL(BR1,1),JUNPP(R1)
(01053) CLFAR(R1)
(01054) NOP(0)
(01055)
(01056) OUTPUT PROGRAM:
(01057) BR0: RSRA(M)
(01058) BR1: LOOP COUNTER (INIT TO HRESID-1)
(01059)
(01060) DCDS$Z: SET(RA)
(01061) LOAD(BR0,RSRA-$S(1))
(01062) LOAD(BR1,HRESID-1)
(01063) JMAIN LOOP
(01064) BR2:
(01065) ADD(RA0,$S,TF)
(01066)
(01067) JSTART APU
(01068) JINIT BR0 TO POINT TO OUTPUT BUFFER-$S
(01069) JINIT LOOP COUNTER
(01070) JGEN OUTPUT ADR

```

PAGE 40: CRRMOJ<0C41C>00016U.NSD.2, 29-Dec-88 15:44:23, Ed: NUL.F
 APS PROGRAM FOR DECODA

AP23-DECODA()

```

A12 03FCA 24111101 (01064)      SWL(0V1,1), JUMP(02)
A13 03FC6 26200030 (01065)      CLEAR(RO)
A14 03FC0 20000020 (01066)      MUP(0)
      0000002A (01067)      END
      0000002A (01068)      DCDSASZ = 0L - DCDSAI
      03FCA 4000 (01069)      )
      03FCA 4000 (01070)      FROM9: DATA $0000,$003E
      03FCB 003F
  )COMPUTE BUS 1 MODULE SIZE
  )(2---9) FOR DECODA, DECODA

```

```

PAGE
49: (DDDD)(<DCA16>88W16V.WSU.2, 29-DEC-88 15:44:23, Ed: VOLF
      APS3--RECDDB() APS PROGRAM FOR DECODE

```

```

(01871) * APS3-DECODEB() APS PROGRAM FOR DECODEB
(01872) }
(01873) } BOUNDS TO APS3-DECODEB
(01874) }
(01875) } SAME AS APS3-DECODEA, EXCEPT FOR "B" BUFFERS INSTEAD OF "M".
(01876) }
(01877) } HEADER BLOCK
(01878) }
(01879) }
(01880) }
(01881) }
(01882) }
(01883) }
(01884) }
(01885) }
(01886) }
(01887) }
(01888) }
(01889) }
(01890) }
(01891) }
(01892) }
(01893) }
(01894) }
(01895) }
(01896) }
(01897) }
(01898) }
(01899) }
(01900) }
(01901) }
(01902) }
(01903) }
(01904) }
(01905) }
(01906) }
(01907) }
(01908) }
(01909) }
(01910) }
(01911) }
(01912) }
(01913) }
(01914) }
(01915) }
(01916) }
(01917) }
(01918) }
(01919) }
(01920) }
(01921) }
(01922) }
(01923) }
(01924) }
(01925) }
(01926) }
(01927) }
(01928) }
(01929) }
(01930) }
(01931) }
(01932) }
(01933) }
(01934) }
(01935) }
(01936) }
(01937) }
(01938) }
(01939) }
(01940) }
(01941) }
(01942) }
(01943) }
(01944) }
(01945) }
(01946) }
(01947) }
(01948) }
(01949) }
(01950) }
(01951) }
(01952) }
(01953) }
(01954) }
(01955) }
(01956) }
(01957) }
(01958) }
(01959) }
(01960) }
(01961) }
(01962) }
(01963) }
(01964) }
(01965) }
(01966) }
(01967) }
(01968) }
(01969) }
(01970) }
(01971) }
(01972) }
(01973) }
(01974) }
(01975) }
(01976) }
(01977) }
(01978) }
(01979) }
(01980) }
(01981) }
(01982) }
(01983) }
(01984) }
(01985) }
(01986) }
(01987) }
(01988) }
(01989) }
(01990) }
(01991) }
(01992) }
(01993) }
(01994) }
(01995) }
(01996) }
(01997) }
(01998) }
(01999) }
(02000) }

```

PAGE 50: C00001C0C1C000160.N50.2, 29-DEC-88 15:44:23, E41 WULF
APS3-DECODE() APS PROGRAM FOR DECODE

0377E (01924) ;
(01925)

END END OF FILE

PAGE 501 (0000100116>000160.W50.2, 29-Dec-88 1514123, 241 WULF
APS3-DECODE() APS PROGRAM FOR DECODE

0377E (01924) ; END OF FILE
(01925)

PAGE 51:

ACQ 700:

CONTAB:	3300	(01572)	(01603)
CS909W08:	0217C	(00042)	(00130)
CTAC1:	0900	(00049)	(00050)
CTAC2:	0910	(00050)	(00052)
CTAC3:	0900	(00051)	
CTAC6:	0900	(00059)	(00060)
CTAC:	0970	(00050)	(00059)
CTW1:	0930	(00052)	(00053)
CTW2:	0900	(00053)	(00054)
CTW3:	0900	(00054)	(00055)
CTW4:	0910	(00055)	(00056)
CTW5:	0930	(00056)	(00057)
CTW6:	0950	(00057)	(00058)
CTW0:	0900	(00060)	(00061)
01631MT1:	0402	(00076)	(00149)
01631MT2:	0430	(00077)	(00156)
02281MT1:	0400	(00070)	(00163)
02331MT1:	041E	(00079)	(00170)
0480C:	040E	(00510)	(00530)
048CPVA:	03C8	(01619)	(01052)
048CPVB:	0300	(01020)	(01059)
048LOOP:	0001	(00512)	(00526)
048STOP:	0001S	(00510)	(00527)
0432:	002C	(00500)	(00530)
0488MB:	03E7	(00494)	(00517)
0488MB:	0377	(00495)	(00525)
040PT:	037C	(01020)	(01030)
040CTAL:	0002	(00497)	(00537)
04CTB:	0050	(00164)	(00313)
04PO:	00549	(00200)	(01029)
0C0500:	0012	(01793)	(01796)
0C0501:	0014	(01792)	
0C0501:	0001	(01792)	
0C0510:	0017	(01000)	
0C0511:	0019	(01799)	(01003)
0C0512:	0016	(01791)	(01799)
0C050WT:	0001A	(01794)	(01797)
0C05A:	03A0	(00137)	(01036)
0C05A52:	002A	(01020)	(01060)
0C05A1:	03A0	(01037)	(01060)
0C05A0:	000E	(01030)	(01059)
0C050:	0304	(00142)	(01091)
0C05052:	002A	(01003)	(01923)
0C0501:	0304	(01092)	(01923)
0C0500:	000E	(01093)	(01914)
0C00:	035A	(00136)	(00141)
0C005SA:	0000	(01770)	(01774)
0C00532:	00017	(01771)	(01009)
0C0157:	05C40	(00196)	(00157)
0VLC1:	09A00	(00061)	(00062)
0VLC2:	0910	(00062)	(00064)
0VLC3:	09A00	(00063)	(01491)
0VLC:	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)
	09C40	(00196)	(00157)
	09A00	(00061)	(00062)
	0910	(00062)	(00064)
	09A00	(00063)	(01491)
	0900	(00071)	(00072)
	0900	(01770)	(01774)

01C77	(00112)	(00047)		
00553	(00303)	(00705)	(00791)	(00795) (00950) (00977)
0039C	(00220)	(01426)	(01046)	
005A6	(00227)	(01427)	(01901)	
03E0C	(00700)	(00002)	(01427)	(01436)
0053B	(00256)	(00262)	(00739)	
0053C	(00257)	(00740)		
03050	(00740)	(00777)	(00001)	
00547	(00277)	(00776)	(00003)	(00000)
00000	(00233)	(00494)	(00513)	(01053) (01054) (01055)
00EAB	(00234)	(00495)	(00521)	(01060) (01061) (01062)
0057F	(00231)	(00766)		
00547	(00299)	(00004)		
00550	(00300)	(01001)		
02P07	(00342)	(00390)		
03191	(00343)	(00396)		
03390	(00344)	(00402)		
00557	(00157)	(00312)		
0007E	(00223)	(00342)	(00377)	(00404) (00743)
00700	(00224)	(00343)	(00392)	(00744)
00192	(00225)	(00344)	(00390)	(00742) (00745)
0054C	(00200)	(00022)		
030A0	(00709)	(00794)		
030A1	(00793)	(00000)		
030C6	(00770)	(00022)		
0300A	(00775)	(00012)		
030C2	(00762)	(00005)		
030CA	(00760)	(00034)		
03006	(00771)	(00000)		
030C0	(00772)	(00015)		
030AC	(00796)	(00001)		
03062	(00131)	(00150)	(00754)	
00546	(00276)	(00756)	(00760)	
03060	(00745)	(00763)	(00784)	(00797) (00012) (00014)
00570	(00319)	(00323)	(01570)	
00551	(00301)	(00500)	(00765)	
03C00	(01023)	(01035)		
03C04	(01025)	(01036)	(01039)	
00530	(00250)	(01024)		
0053E	(00259)	(01025)		
00040	(00230)	(01022)		
00C20	(00231)	(01023)		
00C00	(00232)	(01047)		
00540	(00209)	(01046)		
00540	(00279)	(01034)	(01041)	
00700	(00220)	(00070)	(01429)	(01060)
0097C	(00229)	(00079)	(01430)	(01915)
03F10	(01430)	(01439)		
070EC	(00230)	(00074)	(00921)	(00930)
070F6	(00239)	(00009)	(00923)	(00924) (00926) (00937)
00552	(00302)	(00769)	(00774)	(00949) (00960) (01000)
0053F	(00293)	(00295)	(00574)	(00652) (00761) (01032)
00001	(00371)			

PAGE 561

SOLGTM:

PAGE 57: (00007<0C116>00N16U.NSO.2, 29-Dec-88 15:44:23, Ed: WOLF
 APS3-BXC000() APS PROGRAM FOR DEC000

TOPHIST:	0900C (00206)	(01140) (01161)	
TSNAPTR:	03CC4 (01132)	(01131)	
TSNRA:	0A100 (00215)	(01132)	
TSNR0:	0A266 (00216)	(00600)	(00610) (00611)
TSNR1:	09700 (00213)	(00617) (00618)	
TSNR0:	0A0A0 (00214)	(00504)	
TSNAPTR:	04956 (00564)	(00566)	
TSNRA:	00534 (00252)	(00260)	
TSNR0:	00535 (00253)	(00567)	
TSNAPTR:	0495A (00567)	(00577) (00586)	
TSNRA:	00543 (00271)	(00576) (00580)	
TSNR0:	037CA (01040)	(01070) (01095)	
WS:	00002 (00000)	(00050) (00052)	(00053) (00054) (00055) (00056) (00057) (00058) (00059)
		(00060) (00061) (00062) (00063) (00064) (00065) (00066) (00067) (00068) (00069) (00070)	
		(00071) (00072) (00073) (00074) (00075) (00076) (00077) (00078) (00079) (00080) (00081)	
		(00082) (00083) (00084) (00085) (00086) (00087) (00088) (00089) (00090) (00091) (00092)	
		(00093) (00094) (00095) (00096) (00097) (00098) (00099) (00100) (00101) (00102) (00103)	
		(00104) (00105) (00106) (00107) (00108) (00109) (00110) (00111) (00112) (00113) (00114)	
		(00115) (00116) (00117) (00118) (00119) (00120) (00121) (00122) (00123) (00124) (00125)	
		(00126) (00127) (00128) (00129) (00130) (00131) (00132) (00133) (00134) (00135) (00136)	
		(00137) (00138) (00139) (00140) (00141) (00142) (00143) (00144) (00145) (00146) (00147)	
		(00148) (00149) (00150) (00151) (00152) (00153) (00154) (00155) (00156) (00157) (00158)	
		(00159) (00160) (00161) (00162) (00163) (00164) (00165) (00166) (00167) (00168) (00169)	
		(00170) (00171) (00172) (00173) (00174) (00175) (00176) (00177) (00178) (00179) (00180)	
		(00181) (00182) (00183) (00184) (00185) (00186) (00187) (00188) (00189) (00190) (00191)	
		(00192) (00193) (00194) (00195) (00196) (00197) (00198) (00199) (00200) (00201) (00202)	

LINES WITH ERRORS: 0 (MAP VERSION 00101.10) 2- 0

AD-A096 092

BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/6 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 J J WOLF, K D FIELD, W H RUSSELL DCA100-79-C-0037
BBN-4565-VOL-2 NL

UNCLASSIFIED

5 OF 5
NO A
036092

END
DATE
FILMED
4-81
DTIC

C <OCAL6>DBM16C.FOR.1 Mon 18-Nov-88 6:32PM Page 114

```

C CODING TABLE OF PITCH TAP THRESHOLDS C1 & C3
  CTHC1 = 0
  CTHC3 = CTHC1
C
C CODING TABLE OF PITCH TAP THRESHOLDS C2
  CTHC2 = 0
C
C (CTM1-6 MUST BE PHYSICALLY CONTIGUOUS IN MEMORY)
C CODING TABLE OF A1 THRESHOLDS
  CTHC1 = 0
C
C CODING TABLE OF A2 THRESHOLDS
  CTHC2 = 0
C
C CODING TABLE OF A3 THRESHOLDS
  CTHC3 = 0
C
C CODING TABLE OF A4 THRESHOLDS
  CTHC4 = 0
C
C CODING TABLE OF A5 THRESHOLDS
  CTHC5 = 0
C
C CODING TABLE OF A6 THRESHOLDS
  CTHC6 = 0
C
C CODING TABLE OF ENERGY THRESHOLDS
  CTHC = 0
C
C CODING TABLE OF DELTA GAIN THRESHOLDS
  CTHDC = 0
C
C CODING TABLE OF RESIDUAL SAMPLE THRESHOLDS (FOLDED BINARY)
  CTHN = 0
C
C HISTOGRAM BUFFER FOR A'S (MEMORICING AND SYSTEM MEASUREMENT USE)
  HISTO = 0
C
C A/D INPUT FROM ADAM
  TADMA = 0
C
C A/D INPUT FROM ADAM
  TADDB = 0
C
C A/D "ON-NOW" TONE DATA
  TADBC = 0
C
C INPUT SPEECH TO ANL2A
  TSP1 = 1
C
C INPUT SPEECH TO ANL2B
  TSPN = 2

```

C <OCAL6>DBM16C.FOR.1 Mon 18-Nov-88 6:32PM Page 115

```

C UNITY VECTOR
  TONES = 3
C
C PREEMPHASIZED SPEECH (CURRENT FRAME)
  TSPN = 4
C
C PREEMPHASIZED SPEECH (LAST FRAME AND CURRENT FRAME)
  TSP1 = 5
C
C PREEMPHASIZED SPEECH (LAST 49 OF LAST FRAME, ALL OF CURRENT FRAME)
  TSP = 6
C
C HANNING WINDOW COEFFICIENTS FOR 265 SAMPLE WINDOW
  THANN = 7
C
C WINDOWED SPEECH
  TESP = 8
C
C WINDOWED SPEECH WITH APPENDED ZERUS
  TSPZ = 9
C
C COSINE TABLE FOR FFT AND IFFT
  TCOST = 10
C
C FREQUENCY DOMAIN REPRESENTATION OF TSPZ
C (THIS BUFFER CONTAINS 256 COMPLEX POINTS, A 257TH COMPLEX
C POINT WILL BE ASSUMED PRESENT DIRECTLY FOLLOWING IT)
  TESP = 11
C
C REAL PART OF TESP, INCLUDING 257TH POINT
  TESP = 12
C
C IMAGINARY PART OF TESP, INCLUDING 257TH POINT
  TSP1 = 13
C
C COMPLEX BUFFER OF TSPSP, TSPSP1
C (THIS BUFFER CONTAINS 256 COMPLEX POINTS, A 257TH COMPLEX
C POINT WILL BE ASSUMED PRESENT DIRECTLY FOLLOWING IT)
  TSP = 14
C
C POWER SPECTRUM OF TESP, COMPRISING REAL PART OF TSPSP,
C INCLUDING 257TH POINT.
  TSPSP = 15
C
C ZERUS, COMPRISING IMAGINARY PART OF TSPSP, INCLUDING 257TH POINT
  TSP1 = 16
C
C AUTOCORRELATION COEFFS FOR PITCH DETERMINATION
  THP = 17
C

```

```

C <CAL6>BMM16C.FUN.1 Mon 18-Nov-88 6:32PM Page 1:6
C <CAL6>BMM16C.FUN.1 Mon 18-Nov-88 6:32PM Page 1:7

C PITCH PREDICTOR COEFFS (TIPS)
TC = 18
C AUTOCORRELATION COEFFS: -TEP(PITCH+M); M=-1,0,+1
TRPP = 19
C SUM AKA FOR MEL
THORE = 28
C CODED PITCH PREDICTOR COEFFS
TIC = 21
C UNANTIZED PITCH PREDICTOR COEFFS
TCM = 22
C FIRST RESIDUAL (CURRENT FRAME)
TE10 = 23
C FIRST RESIDUAL (LAST 6 SAMPLES OF CURRENT FRAME)
TE11 = 24
C FIRST RESIDUAL (LAST 6 SAMPLES OF LAST FRAME, PLUS CURRENT FRAME)
TE1 = 25
C NAMING WINDOW COEFFS FOR 216 SAMPLE WINDOW
THAN1 = 26
C WINDOWED FIRST RESIDUAL
TE1 = 27
C AUTOCORRELATION COEFFS FOR SPECTRAL ANALYSIS
TRS = 28
C LAMBDA = W(M) FOR APC MODULE
TLAU = 29
C MPC'S AUTOCORRELATION COEFFS
TRSP = 30
C REFLECTION COEFFS & ERROR TERMS FROM MNLQ6
TRE = 31
C QUANTIZED REFLECTION COEFFS
TKN = 32
C CODED REFLECTION COEFFS
TIA = 33
C PREDICTOR COEFFS A(1)-A(6) (ALSO START OF FILTER COEFF BLOCK)
TAM = 34
TILT = TAN

C <DCA16>BMM16C.FUN.1 Mon 18-Nov-88 6:32PM Page 1:7
C PREDICTOR COEFFS (REVERSE BUFFER), WITH A(0)=1 ADDED
TAHR = 35
C SECOND RESIDUAL
TE2 = 36
C FACTORS FOR NOISE SHAPING ANALYSIS
TFAC = 37
C NOISE SHAPING FILTER COEFFS
TANS = 38
C QUANTIZER SCALE FACTORS, ALTERNATING WITH THEIR RECIPROCALLS
TFAC = 39
C CODED DELTA GAINS
TIDG = 40
C UNQUANTIZED, NORMALIZED RESIDUAL (FOR DEBUGGING ONLY)
TUO = 41
C HISTORY FOR NOISE-SHAPING FILTER
TQ1 = 42
C HISTORY FOR SPECTRAL FILTER
TVN = 43
C APC PITCH FILTER HISTORY (CURRENT FRAME)
TRH = 44
C APC PITCH FILTER HISTORY (LAST FRAME PLUS CURRENT FRAME)
TRM = 45
C CODED RESIDUAL SAMPLES FROM ANLZA
C (ALSO, START OF CODED TRANSMISSION DATA BUFFER FROM ANLZA,
C WITH FORMAT T10A,T1C,T1DG,T1M,T1C,T1K)
T10A = 46
T10B = 47
T10C = 48
T10D = 49
C CODED RESIDUAL SAMPLES FROM ANLZB
C (ALSO, START OF CODED TRANSMISSION DATA BUFFER FROM ANLZB,
C WITH FORMAT T10B,T1C,T1DG,T1M,T1C,T1K)
T10B = 47
T10C = 48
T10D = 49
C T01TS A BUFFER
T01A = 0
C T01TS B BUFFER
T01B = 0

```


C <UCAL0>MMH10C.FUR.1 Mon 10-Nov-80 6:32PM

```

C
C CONSTANT RECIPROCAL OF 72
  172M = 68
C
C CONSTANT RECIPROCAL OF 3
  73M = 61
C
C RECEIVED SART(FRAME ENERGY)
  RCH = 65
C
C RECEIVED PITCH
  RMH = 66
C
C LEEMPHASIS FILTER COEFFICIENT SA0 (MUST BE CONTIGUOUS /---\ )
  HDCF0 = 67
C
C DEEMPHASIS FILTER COEFFICIENT SA1
  HDCF1 = 68
C
C DEEMPHASIS FILTER COEFFICIENT SA2
  HDCF2 = 69
C
C LEEMPHASIS FILTER COEFFICIENT SA3
  HDCF3 = 70
C
C DEEMPHASIS FILTER MEMORY V(-2) (MUST BE CONTIGUOUS /---\ )
  RDMV0 = 71
C
C DEEMPHASIS FILTER MEMORY V(-1)
  RDMV1 = 72
C
C DEEMPHASIS FILTER MEMORY U(-2)
  RDMV2 = 73
C
C DEEMPHASIS FILTER MEMORY U(-1)
  RDMV3 = 74
C
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C CONFIGURE SNAP-11 INTEGER SCALARS
C
C THIS MODULE CONFIGURES THE MAP INTEGER SCALARS.
C ALL MAP INTEGER SCALAR ID'S (SID'S) ARE
C SYMBOLICALLY NAMED, WITH NAMES (IN GENERAL) STARTING WITH
C 'T' FOR TRANSMITTER SCALARS, AND
C WITH 'R' FOR RECEIVER SCALARS.
C (FOR BUFFER STATUS FLAGS: 0=EMPTY, 1=FULL)
C

```

C <UCAL0>MMH10C.FUR.1 Mon 10-Nov-80 6:32PM

```

C
C CALL MPCLB(1000000000, ANCH, SDCM, RL, CNTG,LNG )
C CALL MPCLB(1000000000, ANCH, SDCM, RL, CNTG,LNG )
C CALL MPCLB(1000000000, ANCH, SDCM, RL, CNTG,LNG )
C CALL MPCLB(1000000000, ANCH, SDCM, RL, CNTG,LNG )
C CALL MPCLB(1000000000, ANCH, SDCM, RL, CNTG,LNG )
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C CONFIGURE SNAP-11 REAL SCALARS
C
C THIS MODULE CONFIGURES THE SYSTEM REAL SCALARS.
C ALL MAP REAL SCALAR ID'S (SID'S) ARE SYMBOLICALLY
C NAMED, WITH NAMES STARTING WITH 'T' FOR
C TRANSMITTER SCALARS, AND NAMES STARTING WITH 'R'
C FOR RECEIVER SCALARS.
C
C
C PREEMPHASIS FILTER COEFFICIENT
  TALPH = 50
C
C DEEMPHASIS FILTER HISTORY
  THST = 51
C
C NEGATIVE OF CURRENT FRAME D-C. TSDM
  TDCN = 52
C
C NEGATIVE OF RECIPROCAL OF SIZE OF PITCH EXTRACTION BUFFER
  TFSZ1 = 53
C
C (QUANTIZED) PITCH LAG (IN SAMPLES)
  THN = 54
C
C CUMUL PITCH LAG (INTEGER IN LEFT HALFWORD)
  THN = 55
C
C CONSTANT 14
  T14 = 56
C
C THRESHOLD VALUE FOR MEUPL, NUL06
  TTHR = 57
C
C CODED FRAME ENERGY (INTEGER IN LEFT HALFWORD)
  TIC = 58
C
C PITCH PREDICTION COEFF FOR SINGLE TAP CASE
  TCJP = 59

```

C <SCALE>BMB10C.FDM.1 Mon 18-Nov-88 6:32PM

C
C
C TSMA BUFFER STATUS FLAG
TSMPA = 50

C
C TSMB BUFFER STATUS FLAG
TSMPB = 51

C
C TSMA BUFFER STATUS FLAG
TSMPA = 52

C
C TSMB BUFFER STATUS FLAG
TSMPB = 53

C
C NMLP OUTPUT: NUMBER OF ITERATIONS NOT COMPLETED (TINC,TENC CONTIG.)
TINC = 54

C
C NMLP OUTPUT: FRACTIONAL ENERGY E(N)/R(0) (TINC,TENC CONTIG.)
TENC = 55

C (UNUSE)
1 = 56

C
C DATA BUFFER STATUS FLAG
MSDPA = 57

C
C DATA BUFFER STATUS FLAG
MSDPB = 58

C
C BSMKA BUFFER STATUS FLAG
BSMPA = 59

C
C BSMKB BUFFER STATUS FLAG
BSMPB = 60

C
C SYSTEM RUN FLAG (0 -> STOP)
MUN = 61

C
C (LEFT UNUSED FOR MP1TH)
1 = 62

C (UNUSE)
1 = 63

C
C TSMA/B BUFFER POINTER (0->B) -2->A)
TSMPA = 64

C
C TSMA/B BUFFER POINTER (0->B) -2->A)
TSMPB = 65

L <SCALE>BMB10C.FDM.1 Mon 18-Nov-88 6:32PM Page 1:1B

C
C TSMA/B BUFFER POINTER (0->B) -2->A)
TSMPA = 66

C
C TSMA/B BUFFER POINTER (0->B) -2->A)
TSMPB = 67

C
C TSMA/B/C BUFFER POINTER (0->C) -2->B) -4->A)
TSMPD = 68

C
C TSMA/B BUFFER POINTER (0->B) -2->A)
TSMPB = 69

C
C BSMKA/B BUFFER POINTER (0->B) -2->A)
BSMPA = 70

C
C BSMKA/B BUFFER POINTER (0->B) -2->A)
BSMPB = 71

C
C A/D FRAME DISCARD COUNTER
TAUDC = 72

C
C THODEM FAKE FRAME COUNTER
TMFEC = 73

C
C THODEM FRAME DISCARD COUNTER
TMFDC = 74

C
C BSINK DATA-NOT-READY COUNTER
BSNDR = 75

C
C TRANSMITTER FRAME COUNTER
TFCTR = 76

C
C RECEIVER FRAME COUNTER
RVCTR = 77

C
C RECEIVER LOST-SYNC COUNTER
RLCTR = 78

C
C LOCAL HANDSET ON-HOOK STATE (0->HANDSET OFF-HOOK)
KONHR = 79

C
C STATE OF RECEIVER SYNC (0->LUST SYNC, -1->SEARCHING SYNC) 1->HAVE SYNC.
RSYNC = 80

C
C BEGINNING-OF-FRAME OFFSET (SYNC BIT POSITION IN THODEM BUFFER)
MBOFU = 81

C
C (LEFT UNUSED FOR MP1TH)
1 = 82

C <LOCAL>MM16C.FOR.1 Mon 18-Nov-88 6:32PM

```

C A/D (ADM) INTERRUPT COUNTER
  ADCIN = 83
C TIMER (IOS-2) INTERRUPT COUNTER
  TMCIN = 84
C ANOUEM (IOS-2) INTERRUPT COUNTER
  ANOUEM = 85
C D/A (ADM) INTERRUPT COUNTER
  DACIN = 86
C A/D PEAK INPUT LEVEL
  TADPE = 87
C NO-ERROR-CORRECTION SWITCH (NO CORRECTIONS IF M2)
  INCOR = 123
C (LEFT UNUSED FOR MP1M)
  I = 124
C CHANNEL ERROR SIMULATOR FLAG (NON-ZERO => 8 SIM ERRORS PER FRAME)
  ERRS = 125
C (LEFT BRUSED FOR MP1M)
  I = 126
C
C
C
C
C
  RETURN
  END

```

[illegible]

```
C
      INTEGER FUNCTION KCLUSK(I,MMAY)
      DIMENSION IARRAY(1)
      TYPE 100
      100 FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
      STOP
      END

C
C
C      SUBROUTINE DELAY(I,J)
      TYPE 100
      100 FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
      STOP
      END
```

[illegible]

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C EXECUTE FILE TO FILE SYSTEM

C CONTINUE

C

C DEFINE INPUT AND OUTPUT FILES

C

C TYPE 216

C FORMAT(' READ INPUT SIGNAL FROM FILE: '\$)

C NBLKS = KOPR(KSIH,M)

C IF(NBLKS.LE.0) GO TO 203

C CALL SCOPY('SKIP HEADER BLOCK (256 WORDS)? ',MSG)

C IF (.NOT. CMPIRM(MSG)) GO TO 215

C ISTAT = KSIH(MDABLK,256,MDFRD)

C

C TYPE 228

C FORMAT(' WRITE OUTPUT SIGNAL TO FILE: '\$)

C NBLKS = KOPR(LSOT,0,0,TRUE.)

C IF(NBLKS.LE.0) GO TO 215

C

C USE DUMMY BUFFER TO READ & WRITE TABDAB,M & MDAB,M

C SINCE THEY ARE RL,SHORT FORMAT, AND MPDAB/MPDRB

C DON'T PROPERLY ACCESS RL,SHORT BUFFERS.

C USE BUFFER FROM SYSTEM WHICH IS CORRECT LENGTH (216

C SAMPLES), RL, LRG, AND IS NOT NEEDED FOR FRAME-TU-

C FRAME MEMORY.

C

C IDUM = 1481

C IADBUF = 60

C IADBUF = 61

C IADUM = 60

C IADMA = 61

C

C

C "Q" COMMAND -- QUIT (HALT SPEECH CODE)

C CALL MP1TH(RUN,0,0)

C CALL MP1TH(RUN,0,0)

C (THERE'S NO WAY TO STOP THE MUDEN SCROLL)

C

C RETURN TO COMMON CUMS

C GO TO 988

C

C "Q" COMMAND -- SET ERROR SIMULATION

C IF(B.LT.0 .OR. B.GT.50) GO TO 115

C CALL MP1TH(RESR,0,0)

C GO TO 112

C

C "Q" COMMAND -- CONTROL ERROR CORRECTION

C IF(B.NE.0) B=1

C CALL MP1TH(RCOR,M,0)

C GO TO 112

C

C "Q" COMMAND -- CAUSE THE RCVR TO LOSE SYNC BY JAMMING INTO RDOFU

C CALL MP1TH(RDOFU,50,0)

C GO TO 112

C

C "T" COMMAND -- STOP CODE, READ INTEGER SCALARS, RESET TADPA

C (PEAK INPUT LEVEL SO FAR) TO ZERO, THEN RESTART,

C AND TYPE OUT INTEGER SCALAR VALUES

C CALL MP1TH(RUN,0,0)

C CALL MP1TH(TSOPA,1ST(TSOPA),30,0)

C CALL MP1TH(TADPE,0)

C CALL MP1TH(ORHNS)

C CALL TST(1ST)

C GO TO 112

C

C "S" COMMAND -- SUSPEND THIS TASK, BUT LET MAP CONTINUE.

C (BPOB RESUMPTION OF TASK, IT WILL CONTINUE OUT OF THE PAUSE)

C PAUSE '(SPEECH CODE CONTINUING)'

C GO TO 110

C

Line	Code	Text	Page	Date	Time
210	C	ASK ABOUT FRAME TYPES	11	10-Nov-88	6:35PM
211	C	LIST--FALSE.	11	10-Nov-88	6:35PM
212	C	TYPE 216	11	10-Nov-88	6:35PM
213	C	FORMAT("FRAME TYPEOUTS? (TYPE, "8-CALARS, OR MIL): ")	11	10-Nov-88	6:35PM
214	C	ACCEPT 227,ICMD	11	10-Nov-88	6:35PM
215	C	FORMAT(1)	11	10-Nov-88	6:35PM
216	C	IF(ICMD.EQ.CHRS) LIST--TRUE.	11	10-Nov-88	6:35PM
217	C	IF(ICMD.NE.CHRS) GO TO 230	11	10-Nov-88	6:35PM
218	C	LIST--TRUE.	11	10-Nov-88	6:35PM
219	C	TYPE 228	11	10-Nov-88	6:35PM
220	C	FORMAT("FILE FOR TSINK DATA: ")	11	10-Nov-88	6:35PM
221	C	CALL ASSIGN(2,ISIG,-1)	11	10-Nov-88	6:35PM
222	C	WRITE(2,229)	11	10-Nov-88	6:35PM
223	C	FORMAT(" F40 6 D11 D12 D13 4 C1 C2 C3 K1 ",	11	10-Nov-88	6:35PM
224	C	"E2 E3 E4 E5 E6 RESIGNAL")	11	10-Nov-88	6:35PM
225	C	TYPE OF LOOP	11	10-Nov-88	6:35PM
226	C	1STAT = KSIG(ISIG,ISIG,216,MWIFRO)	11	10-Nov-88	6:35PM
227	C	IF (1STAT.EQ. IEOP) GO TO 230	11	10-Nov-88	6:35PM
228	C	DO 235 I=1,216	11	10-Nov-88	6:35PM
229	C	SIG(I) = FLOOR(ISIG(I) / 32768.	11	10-Nov-88	6:35PM
230	C	SET FLAG FOR "A" OR "M" MWIFERS	11	10-Nov-88	6:35PM
231	C	ABUFS = .FALSE.	11	10-Nov-88	6:35PM
232	C	IF(MOD(MWIFERS,2).EQ.0) ABUFS = .TRUE.	11	10-Nov-88	6:35PM
233	C	ABUFS = .NOT. ABUFS	11	10-Nov-88	6:35PM
234	C	IF (ABUFS) ITSINK=TSINKA	11	10-Nov-88	6:35PM
235	C	IF (ABUFS) ITSINK=TSINKB	11	10-Nov-88	6:35PM
236	C	IFAB SIG TO MAP	11	10-Nov-88	6:35PM
237	C	CALL MWIFR(IOWN,SIG,4,1,SIG(216))	11	10-Nov-88	6:35PM
238	C	IF (ABUFS) CALL MPCLB(1BUS1*IDABUF,ATADBA,STADBA,RL,CNTG,SHRT)	11	10-Nov-88	6:35PM
239	C	IF (ABUFS) CALL MPCLB(1BUS1*IDABUF,ATADBB,STADBB,RL,CNTG,SHRT)	11	10-Nov-88	6:35PM
240	C	CALL MPWT(PRCR,MS)	11	10-Nov-88	6:35PM
241	C	CALL VMOV(IADUUF,IOWN)	11	10-Nov-88	6:35PM
242	C	CALL MPWT(PRCR,AP)	11	10-Nov-88	6:35PM
243	C	EXECUTE 1 PASS THRU SYSTEM LOOP	11	10-Nov-88	6:35PM
244	C	CALL MPWFL(DMPTT)	11	10-Nov-88	6:35PM
245	C	MAKE MUST WAIT 12016.67 MSEC = 200 MSEC.	11	10-Nov-88	6:35PM
246	C	CALL DELAY(2,12)	11	10-Nov-88	6:35PM
247	C	END MOVE BITS FROM TRUNK BUFFER TO REMOTE BUFFER, AS A CHANNEL	11	10-Nov-88	6:35PM
248	C	DOES.	11	10-Nov-88	6:35PM
249	C	GO TO(241,242,243,244,245,246) I=MOD(MWIFERS,6)	11	10-Nov-88	6:35PM

```

C <DCA10>>BUBBLE.FOR.1 Mon 18-Nov-88 6:13PM Page 313
C IF REQUESTED, DO FRAME TYPEOUT
IF(.NOT.LTSM) GO TO 260
CALL MPFOR(LTSM,ISIG,2,0,ISIG(230))
WRITE(2,255) NFRAME,(ISIG(I), I=217,230),(ISIG(I),I=1,216)
FORMAT(15,15,1414,/,7211,/,7211,/,7211,/)
255 IF(LIST) CALL LIST
260 NFRAME=NFRAME+1
GO TO 270
C
C HERE WE END.
C
290 CALL LIST
TYPE 2%,NFRAME,MCLIPS
FORMAT(17," FRAMES=",15," OUTPUT SAMPLES WERE CLIPPED.")
LISTAT = KCLOSE(ISIG)
LISTAT = KCLOSE(LIST)
IF(LTSM) CALL CLOSE(2)
C
C RETURN TO COMMON CODE
C
COTO 900
~L
C <DCA10>>BUBBLE.FOR.1 Mon 18-Nov-88 6:13PM Page 4
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C EXECUTE TIMING SYSTEM
C
300 CONTINUE
C
C GET TIMING PARAMETERS.
C
TYPE 310
FORMAT(" ENTER ITERATION VALUE: ")
ACCENT 311,M
311 FORMAT(13)
C
C PUT SOME TIMING REASONABLE IN TAUMA, TADUMB
C (DON'T USE BID 63 SINCE VCOST USES IT IMPLICITLY.)
C
IADMBF = 60
CALL MFCLE(TAUMSI,IADMBF,ATAUMA,STAUMA,NL,CNTG,SHMT)
CALL VCOST(IADMBF,IT.)
CALL MFCLE(TADUMSI,IADMBF,ATADUMB,STADUMB,NL,CNTG,SHMT)
CALL VCOST(IADMBF,IT.)
C
C SET UP TIMING (100 USEC TICKS)
HEAD CLOCK
C
CALL SETCLK(1)
IT1 = IRUCLA(0)
C
EXECUTE DBMTIM "M" TIMES
C
CALL MPJST(QUN,1)
CALL MPFOR(1,1,B,DBMTIM)
C
WAIT ON MAP, READ CLOCK
C
CALL MPJST(1,WOME,1,1)
IT2 = IRUCLA(0)
TIME = FLNAT((OCLP(IT1,IT2)))
C
CONVERT TIME, PRINT IT
C
TIMENS = TIME * .1
TYPE 320,M,TIMENS
FORMAT(16," ITERATION(S) TOOK",F10.4," MSEC(S).")
320
C
RETURN TO COMMON CODE
C
COTO 900
~L

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      EXECUTE SCODE TUNING SYSTEM
C
400  CONTINUE
C
C      PUT SOMETHING REASONABLE IN TADMA,TADDB
C      (DON'T USE DIV 63 SINCE VCOST USES IT IMPLICITLY..)
C
      TADDB = 60
      CALL WCLB((IUBS1:TADDBF,ATADMA,STADBA,RL,COTG,SHRT)
      CALL VCOST(TADDBF,17.)
      CALL WCLB((IUBS1:TADDBF,ATADDB,STADDB,RL,COTG,SHRT)
      CALL VCOST(TADDBF,17.)
C
C      START UP INFINITE MPIDL UN BUNTIN
C
      CALL MPIDL(RUN,1)
      CALL MPIDL(RUN,16,0,BUNTIN)
C
      WAIT FOR USER TO SAY QUIT
C
      TYPE 410
      FORMAT(' TYPE ANY CHARACTER TO HALT MAP1 '3)
      ACCEPT 411,ICAR
      FORMAT(A1)
C
      STOP MPIDL
C
      CALL MPIDL(RUN,0,0)
C
      RETURN TO COMMON CUBE
C
COTO 906
C
      END

```

```

SUBROUTINE RIST
C      READ IST AND TYPE OUT CONTENTS
C      DIMENSION IST(87)
C
      CALL MPIDL(1,IST(1),87,0)
      CALL TIST(IST)
C
      RETURN
      END
C
SUBROUTINE TIST(IST)
C      TYPE OUT SELECTED VALUES FROM IST ARRAY
C
C <ENDMD3<MAP>000106-COMMON.F00.12, 31-Oct-80 23:17:03, EJS:WILF
C
C      MAP BUFFER NAMES:
C
      INTEGER DVLC1,DVLC2,DVLC3,DVLC4,DVLC5,DVLC6,DVLC7,DVLC8,DVLC9
      INTEGER DVLC10,DVLC11,DVLC12,DVLC13,DVLC14,DVLC15,DVLC16,DVLC17
      INTEGER DVLC18,DVLC19,DVLC20,DVLC21,DVLC22,DVLC23,DVLC24,DVLC25
      INTEGER DVLC26,DVLC27,DVLC28,DVLC29,DVLC30,DVLC31,DVLC32,DVLC33
      INTEGER DVLC34,DVLC35,DVLC36,DVLC37,DVLC38,DVLC39,DVLC40,DVLC41
      INTEGER DVLC42,DVLC43,DVLC44,DVLC45,DVLC46,DVLC47,DVLC48,DVLC49
      INTEGER DVLC50,DVLC51,DVLC52,DVLC53,DVLC54,DVLC55,DVLC56,DVLC57
      INTEGER DVLC58,DVLC59,DVLC60,DVLC61,DVLC62,DVLC63,DVLC64,DVLC65
      INTEGER DVLC66,DVLC67,DVLC68,DVLC69,DVLC70,DVLC71,DVLC72,DVLC73
      INTEGER DVLC74,DVLC75,DVLC76,DVLC77,DVLC78,DVLC79,DVLC80,DVLC81
      INTEGER DVLC82,DVLC83,DVLC84,DVLC85,DVLC86,DVLC87,DVLC88,DVLC89
      INTEGER DVLC90,DVLC91,DVLC92,DVLC93,DVLC94,DVLC95,DVLC96,DVLC97
      INTEGER DVLC98,DVLC99,DVLC100,DVLC101,DVLC102,DVLC103,DVLC104
      INTEGER DVLC105,DVLC106,DVLC107,DVLC108,DVLC109,DVLC110,DVLC111
      INTEGER DVLC112,DVLC113,DVLC114,DVLC115,DVLC116,DVLC117,DVLC118
      INTEGER DVLC119,DVLC120,DVLC121,DVLC122,DVLC123,DVLC124,DVLC125
      INTEGER DVLC126,DVLC127,DVLC128,DVLC129,DVLC130,DVLC131,DVLC132
      INTEGER DVLC133,DVLC134,DVLC135,DVLC136,DVLC137,DVLC138,DVLC139
      INTEGER DVLC140,DVLC141,DVLC142,DVLC143,DVLC144,DVLC145,DVLC146
      INTEGER DVLC147,DVLC148,DVLC149,DVLC150,DVLC151,DVLC152,DVLC153
      INTEGER DVLC154,DVLC155,DVLC156,DVLC157,DVLC158,DVLC159,DVLC160
      INTEGER DVLC161,DVLC162,DVLC163,DVLC164,DVLC165,DVLC166,DVLC167
      INTEGER DVLC168,DVLC169,DVLC170,DVLC171,DVLC172,DVLC173,DVLC174
      INTEGER DVLC175,DVLC176,DVLC177,DVLC178,DVLC179,DVLC180,DVLC181
      INTEGER DVLC182,DVLC183,DVLC184,DVLC185,DVLC186,DVLC187,DVLC188
      INTEGER DVLC189,DVLC190,DVLC191,DVLC192,DVLC193,DVLC194,DVLC195
      INTEGER DVLC196,DVLC197,DVLC198,DVLC199,DVLC200,DVLC201,DVLC202
      INTEGER DVLC203,DVLC204,DVLC205,DVLC206,DVLC207,DVLC208,DVLC209
      INTEGER DVLC210,DVLC211,DVLC212,DVLC213,DVLC214,DVLC215,DVLC216
      INTEGER DVLC217,DVLC218,DVLC219,DVLC220,DVLC221,DVLC222,DVLC223
      INTEGER DVLC224,DVLC225,DVLC226,DVLC227,DVLC228,DVLC229,DVLC230
      INTEGER DVLC231,DVLC232,DVLC233,DVLC234,DVLC235,DVLC236,DVLC237
      INTEGER DVLC238,DVLC239,DVLC240,DVLC241,DVLC242,DVLC243,DVLC244
      INTEGER DVLC245,DVLC246,DVLC247,DVLC248,DVLC249,DVLC250,DVLC251
      INTEGER DVLC252,DVLC253,DVLC254,DVLC255,DVLC256,DVLC257,DVLC258
      INTEGER DVLC259,DVLC260,DVLC261,DVLC262,DVLC263,DVLC264,DVLC265
      INTEGER DVLC266,DVLC267,DVLC268,DVLC269,DVLC270,DVLC271,DVLC272
      INTEGER DVLC273,DVLC274,DVLC275,DVLC276,DVLC277,DVLC278,DVLC279
      INTEGER DVLC280,DVLC281,DVLC282,DVLC283,DVLC284,DVLC285,DVLC286
      INTEGER DVLC287,DVLC288,DVLC289,DVLC290,DVLC291,DVLC292,DVLC293
      INTEGER DVLC294,DVLC295,DVLC296,DVLC297,DVLC298,DVLC299,DVLC300
      INTEGER DVLC301,DVLC302,DVLC303,DVLC304,DVLC305,DVLC306,DVLC307
      INTEGER DVLC308,DVLC309,DVLC310,DVLC311,DVLC312,DVLC313,DVLC314
      INTEGER DVLC315,DVLC316,DVLC317,DVLC318,DVLC319,DVLC320,DVLC321
      INTEGER DVLC322,DVLC323,DVLC324,DVLC325,DVLC326,DVLC327,DVLC328
      INTEGER DVLC329,DVLC330,DVLC331,DVLC332,DVLC333,DVLC334,DVLC335
      INTEGER DVLC336,DVLC337,DVLC338,DVLC339,DVLC340,DVLC341,DVLC342
      INTEGER DVLC343,DVLC344,DVLC345,DVLC346,DVLC347,DVLC348,DVLC349
      INTEGER DVLC350,DVLC351,DVLC352,DVLC353,DVLC354,DVLC355,DVLC356
      INTEGER DVLC357,DVLC358,DVLC359,DVLC360,DVLC361,DVLC362,DVLC363
      INTEGER DVLC364,DVLC365,DVLC366,DVLC367,DVLC368,DVLC369,DVLC370
      INTEGER DVLC371,DVLC372,DVLC373,DVLC374,DVLC375,DVLC376,DVLC377
      INTEGER DVLC378,DVLC379,DVLC380,DVLC381,DVLC382,DVLC383,DVLC384
      INTEGER DVLC385,DVLC386,DVLC387,DVLC388,DVLC389,DVLC390,DVLC391
      INTEGER DVLC392,DVLC393,DVLC394,DVLC395,DVLC396,DVLC397,DVLC398
      INTEGER DVLC399,DVLC400,DVLC401,DVLC402,DVLC403,DVLC404,DVLC405
      INTEGER DVLC406,DVLC407,DVLC408,DVLC409,DVLC410,DVLC411,DVLC412
      INTEGER DVLC413,DVLC414,DVLC415,DVLC416,DVLC417,DVLC418,DVLC419
      INTEGER DVLC420,DVLC421,DVLC422,DVLC423,DVLC424,DVLC425,DVLC426
      INTEGER DVLC427,DVLC428,DVLC429,DVLC430,DVLC431,DVLC432,DVLC433
      INTEGER DVLC434,DVLC435,DVLC436,DVLC437,DVLC438,DVLC439,DVLC440
      INTEGER DVLC441,DVLC442,DVLC443,DVLC444,DVLC445,DVLC446,DVLC447
      INTEGER DVLC448,DVLC449,DVLC450,DVLC451,DVLC452,DVLC453,DVLC454
      INTEGER DVLC455,DVLC456,DVLC457,DVLC458,DVLC459,DVLC460,DVLC461
      INTEGER DVLC462,DVLC463,DVLC464,DVLC465,DVLC466,DVLC467,DVLC468
      INTEGER DVLC469,DVLC470,DVLC471,DVLC472,DVLC473,DVLC474,DVLC475
      INTEGER DVLC476,DVLC477,DVLC478,DVLC479,DVLC480,DVLC481,DVLC482
      INTEGER DVLC483,DVLC484,DVLC485,DVLC486,DVLC487,DVLC488,DVLC489
      INTEGER DVLC490,DVLC491,DVLC492,DVLC493,DVLC494,DVLC495,DVLC496
      INTEGER DVLC497,DVLC498,DVLC499,DVLC500,DVLC501,DVLC502,DVLC503
      INTEGER DVLC504,DVLC505,DVLC506,DVLC507,DVLC508,DVLC509,DVLC510
      INTEGER DVLC511,DVLC512,DVLC513,DVLC514,DVLC515,DVLC516,DVLC517
      INTEGER DVLC518,DVLC519,DVLC520,DVLC521,DVLC522,DVLC523,DVLC524
      INTEGER DVLC525,DVLC526,DVLC527,DVLC528,DVLC529,DVLC530,DVLC531
      INTEGER DVLC532,DVLC533,DVLC534,DVLC535,DVLC536,DVLC537,DVLC538
      INTEGER DVLC539,DVLC540,DVLC541,DVLC542,DVLC543,DVLC544,DVLC545
      INTEGER DVLC546,DVLC547,DVLC548,DVLC549,DVLC550,DVLC551,DVLC552
      INTEGER DVLC553,DVLC554,DVLC555,DVLC556,DVLC557,DVLC558,DVLC559
      INTEGER DVLC560,DVLC561,DVLC562,DVLC563,DVLC564,DVLC565,DVLC566
      INTEGER DVLC567,DVLC568,DVLC569,DVLC570,DVLC571,DVLC572,DVLC573
      INTEGER DVLC574,DVLC575,DVLC576,DVLC577,DVLC578,DVLC579,DVLC580
      INTEGER DVLC581,DVLC582,DVLC583,DVLC584,DVLC585,DVLC586,DVLC587
      INTEGER DVLC588,DVLC589,DVLC590,DVLC591,DVLC592,DVLC593,DVLC594
      INTEGER DVLC595,DVLC596,DVLC597,DVLC598,DVLC599,DVLC600,DVLC601
      INTEGER DVLC602,DVLC603,DVLC604,DVLC605,DVLC606,DVLC607,DVLC608
      INTEGER DVLC609,DVLC610,DVLC611,DVLC612,DVLC613,DVLC614,DVLC615
      INTEGER DVLC616,DVLC617,DVLC618,DVLC619,DVLC620,DVLC621,DVLC622
      INTEGER DVLC623,DVLC624,DVLC625,DVLC626,DVLC627,DVLC628,DVLC629
      INTEGER DVLC630,DVLC631,DVLC632,DVLC633,DVLC634,DVLC635,DVLC636
      INTEGER DVLC637,DVLC638,DVLC639,DVLC640,DVLC641,DVLC642,DVLC643
      INTEGER DVLC644,DVLC645,DVLC646,DVLC647,DVLC648,DVLC649,DVLC650
      INTEGER DVLC651,DVLC652,DVLC653,DVLC654,DVLC655,DVLC656,DVLC657
      INTEGER DVLC658,DVLC659,DVLC660,DVLC661,DVLC662,DVLC663,DVLC664
      INTEGER DVLC665,DVLC666,DVLC667,DVLC668,DVLC669,DVLC670,DVLC671
      INTEGER DVLC672,DVLC673,DVLC674,DVLC675,DVLC676,DVLC677,DVLC678
      INTEGER DVLC679,DVLC680,DVLC681,DVLC682,DVLC683,DVLC684,DVLC685
      INTEGER DVLC686,DVLC687,DVLC688,DVLC689,DVLC690,DVLC691,DVLC692
      INTEGER DVLC693,DVLC694,DVLC695,DVLC696,DVLC697,DVLC698,DVLC699
      INTEGER DVLC700,DVLC701,DVLC702,DVLC703,DVLC704,DVLC705,DVLC706
      INTEGER DVLC707,DVLC708,DVLC709,DVLC710,DVLC711,DVLC712,DVLC713
      INTEGER DVLC714,DVLC715,DVLC716,DVLC717,DVLC718,DVLC719,DVLC720
      INTEGER DVLC721,DVLC722,DVLC723,DVLC724,DVLC725,DVLC726,DVLC727
      INTEGER DVLC728,DVLC729,DVLC730,DVLC731,DVLC732,DVLC733,DVLC734
      INTEGER DVLC735,DVLC736,DVLC737,DVLC738,DVLC739,DVLC740,DVLC741
      INTEGER DVLC742,DVLC743,DVLC744,DVLC745,DVLC746,DVLC747,DVLC748
      INTEGER DVLC749,DVLC750,DVLC751,DVLC752,DVLC753,DVLC754,DVLC755
      INTEGER DVLC756,DVLC757,DVLC758,DVLC759,DVLC760,DVLC761,DVLC762
      INTEGER DVLC763,DVLC764,DVLC765,DVLC766,DVLC767,DVLC768,DVLC769
      INTEGER DVLC770,DVLC771,DVLC772,DVLC773,DVLC774,DVLC775,DVLC776
      INTEGER DVLC777,DVLC778,DVLC779,DVLC780,DVLC781,DVLC782,DVLC783
      INTEGER DVLC784,DVLC785,DVLC786,DVLC787,DVLC788,DVLC789,DVLC790
      INTEGER DVLC791,DVLC792,DVLC793,DVLC794,DVLC795,DVLC796,DVLC797
      INTEGER DVLC798,DVLC799,DVLC800,DVLC801,DVLC802,DVLC803,DVLC804
      INTEGER DVLC805,DVLC806,DVLC807,DVLC808,DVLC809,DVLC810,DVLC811
      INTEGER DVLC812,DVLC813,DVLC814,DVLC815,DVLC816,DVLC817,DVLC818
      INTEGER DVLC819,DVLC820,DVLC821,DVLC822,DVLC823,DVLC824,DVLC825
      INTEGER DVLC826,DVLC827,DVLC828,DVLC829,DVLC830,DVLC831,DVLC832
      INTEGER DVLC833,DVLC834,DVLC835,DVLC836,DVLC837,DVLC838,DVLC839
      INTEGER DVLC840,DVLC841,DVLC842,DVLC843,DVLC844,DVLC845,DVLC846
      INTEGER DVLC847,DVLC848,DVLC849,DVLC850,DVLC851,DVLC852,DVLC853
      INTEGER DVLC854,DVLC855,DVLC856,DVLC857,DVLC858,DVLC859,DVLC860
      INTEGER DVLC861,DVLC862,DVLC863,DVLC864,DVLC865,DVLC866,DVLC867
      INTEGER DVLC868,DVLC869,DVLC870,DVLC871,DVLC872,DVLC873,DVLC874
      INTEGER DVLC875,DVLC876,DVLC877,DVLC878,DVLC879,DVLC880,DVLC881
      INTEGER DVLC882,DVLC883,DVLC884,DVLC885,DVLC886,DVLC887,DVLC888
      INTEGER DVLC889,DVLC890,DVLC891,DVLC892,DVLC893,DVLC894,DVLC895
      INTEGER DVLC896,DVLC897,DVLC898,DVLC899,DVLC900,DVLC901,DVLC902
      INTEGER DVLC903,DVLC904,DVLC905,DVLC906,DVLC907,DVLC908,DVLC909
      INTEGER DVLC910,DVLC911,DVLC912,DVLC913,DVLC914,DVLC915,DVLC916
      INTEGER DVLC917,DVLC918,DVLC919,DVLC920,DVLC921,DVLC922,DVLC923
      INTEGER DVLC924,DVLC925,DVLC926,DVLC927,DVLC928,DVLC929,DVLC930
      INTEGER DVLC931,DVLC932,DVLC933,DVLC934,DVLC935,DVLC936,DVLC937
      INTEGER DVLC938,DVLC939,DVLC940,DVLC941,DVLC942,DVLC943,DVLC944
      INTEGER DVLC945,DVLC946,DVLC947,DVLC948,DVLC949,DVLC950,DVLC951
      INTEGER DVLC952,DVLC953,DVLC954,DVLC955,DVLC956,DVLC957,DVLC958
      INTEGER DVLC959,DVLC960,DVLC961,DVLC962,DVLC963,DVLC964,DVLC965
      INTEGER DVLC966,DVLC967,DVLC968,DVLC969,DVLC970,DVLC971,DVLC972
      INTEGER DVLC973,DVLC974,DVLC975,DVLC976,DVLC977,DVLC978,DVLC979
      INTEGER DVLC980,DVLC981,DVLC982,DVLC983,DVLC984,DVLC985,DVLC986
      INTEGER DVLC987,DVLC988,DVLC989,DVLC990,DVLC991,DVLC992,DVLC993
      INTEGER DVLC994,DVLC995,DVLC996,DVLC997,DVLC998,DVLC999,DVLC1000
      INTEGER DVLC1001,DVLC1002,DVLC1003,DVLC1004,DVLC1005,DVLC1006
      INTEGER DVLC1007,DVLC1008,DVLC1009,DVLC1010,DVLC1011,DVLC1012
      INTEGER DVLC1013,DVLC1014,DVLC1015,DVLC1016,DVLC1017,DVLC1018
      INTEGER DVLC1019,DVLC1020,DVLC1021,DVLC1022,DVLC1023,DVLC1024
      INTEGER DVLC1025,DVLC1026,DVLC1027,DVLC1028,DVLC1029,DVLC1030
      INTEGER DVLC1031,DVLC1032,DVLC1033,DVLC1034,DVLC1035,DVLC1036
      INTEGER DVLC1037,DVLC1038,DVLC1039,DVLC1040,DVLC1041,DVLC1042
      INTEGER DVLC1043,DVLC1044,DVLC1045,DVLC1046,DVLC1047,DVLC1048
      INTEGER DVLC1049,DVLC1050,DVLC1051,DVLC1052,DVLC1053,DVLC1054
      INTEGER DVLC1055,DVLC1056,DVLC1057,DVLC1058,DVLC1059,DVLC1060
      INTEGER DVLC1061,DVLC1062,DVLC1063,DVLC1064,DVLC1065,DVLC1066
      INTEGER DVLC1067,DVLC1068,DVLC1069,DVLC1070,DVLC1071,DVLC1072
      INTEGER DVLC1073,DVLC1074,DVLC1075,DVLC1076,DVLC1077,DVLC1078
      INTEGER DVLC1079,DVLC1080,DVLC1081,DVLC1082,DVLC1083,DVLC1084
      INTEGER DVLC1085,DVLC1086,DVLC1087,DVLC1088,DVLC1089,DVLC1090
      INTEGER DVLC1091,DVLC1092,DVLC1093,DVLC1094,DVLC1095,DVLC1096
      INTEGER DVLC1097,DVLC1098,DVLC1099,DVLC1100,DVLC1101,DVLC1102
      INTEGER DVLC1103,DVLC1104,DVLC1105,DVLC1106,DVLC1107,DVLC1108
      INTEGER DVLC1109,DVLC1110,DVLC1111,DVLC1112,DVLC1113,DVLC1114
      INTEGER DVLC1115,DVLC1116,DVLC1117,DVLC1118,DVLC1119,DVLC1120
      INTEGER DVLC1121,DVLC1122,DVLC1123,DVLC1124,DVLC1125,DVLC1126
      INTEGER DVLC1127,DVLC1128,DVLC1129,DVLC1130,DVLC1131,DVLC1132
      INTEGER DVLC1133,DVLC1134,DVLC1135,DVLC1136,DVLC1137,DVLC1138
      INTEGER DVLC1139,DVLC1140,DVLC1141,DVLC1142,DVLC1143,DVLC1144
      INTEGER DVLC1145,DVLC1146,DVLC1147,DVLC1148,DVLC1149,DVLC1150
      INTEGER DVLC1151,DVLC1152,DVLC1153,DVLC1154,DVLC1155,DVLC1156
      INTEGER DVLC1157,DVLC1158,DVLC1159,DVLC1160,DVLC1161,DVLC1162
      INTEGER DVLC1163,DVLC1164,DVLC1165,DVLC1166,DVLC1167,DVLC1168
      INTEGER DVLC1169,DVLC1170,DVLC1171,DVLC1172,DVLC1173,DVLC1174
      INTEGER DVLC1175,DVLC1176,DVLC1177,DVLC1178,DVLC1179,DVLC1180
      INTEGER DVLC1181,DVLC1182,DVLC1183,DVLC1184,DVLC1185,DVLC1186
      INTEGER DVLC1187,DVLC1188,DVLC1189,DVLC1190,DVLC1191,DVLC1192
      INTEGER DVLC1193,DVLC1194,DVLC1195,DVLC1196,DVLC1197,DVLC1198
      INTEGER DVLC1199,DVLC1200,DVLC1201,DVLC1202,DVLC1203,DVLC1204
      INTEGER DVLC1205,DVLC1206,DVLC1207,DVLC1208,DVLC1209,DVLC1210
      INTEGER DVLC1211,DVLC1212,DVLC1213,DVLC1214,DVLC1215,DVLC1216
      INTEGER DVLC1217,DVLC1218,DVLC1219,DVLC1220,DVLC1221,DVLC1222
      INTEGER DVLC1223,DVLC1224,DVLC1225,DVLC1226,DVLC1227,DVLC1228
      INTEGER DVLC1229,DVLC1230,DVLC1231,DVLC1232,DVLC1233,DVLC1234
      INTEGER DVLC1235,DVLC1236,DVLC1237,DVLC1238,DVLC1239,DVLC1240
      INTEGER DVLC1241,DVLC1242,DVLC1243,DVLC1244,DVLC1245,DVLC1246
      INTEGER DVLC1247,DVLC1248,DVLC1249,DVLC1250,DVLC1251,DVLC1252
      INTEGER DVLC1253,DVLC1254,DVLC1255,DVLC1256,DVLC1257,DVLC1258
      INTEGER DVLC1259,DVLC1260,DVLC1261,DVLC1262,DVLC1263,DVLC1264
      INTEGER DVLC1265,DVLC1266,DVLC1267,DVLC1268,DVLC1269,DVLC1270
      INTEGER DVLC1271,DVLC1272,DVLC1273,DVLC1274,DVLC1275,DVLC1276
      INTEGER DVLC1277,DVLC1278,DVLC1279,DVLC1280,DVLC1281,DVLC1282
      INTEGER DVLC1283,DVLC1284,DVLC1285,DVLC1286,DVLC1287,DVLC1288
      INTEGER DVLC1289,DVLC1290,DVLC1291,DVLC1292,DVLC1293,DVLC1294
      INTEGER DVLC1295,DVLC1296,DVLC1297,DVLC1298,DVLC1299,DVLC1300
      INTEGER DVLC1301,DVLC1302,DVLC1303,DVLC1304,DVLC1305,DVLC1306
      INTEGER DVLC1307,DVLC1308,DVLC1309,DVLC1310,DVLC1311,DVLC1312
      INTEGER DVLC1313,DVLC1314,DVLC1315,DVLC1316,DVLC1317,DVLC1318
      INTEGER DVLC1319,DVLC1320,DVLC1321,DVLC1322,DVLC1323,DVLC1324
      INTEGER DVLC1325,DVLC1326,DVLC1327,DVLC1328,DVLC1329,DVLC1330
      INTEGER DVLC1331,DVLC1332,DVLC1333,DVLC1334,DVLC1335,DVLC1336
      INTEGER DVLC1337,DVLC1338,DVLC1339,DVLC1340,DVLC1341,DVLC1342
      INTEGER DVLC1343,DVLC1344,DVLC1345,DVLC1346,DVLC1347,DVLC1348
      INTEGER DVLC1349,DVLC1350,DVLC1351,DVLC1352,DVLC1353,DVLC1354
      INTEGER DVLC1355,DVLC1356,DVLC1357,DVLC1358,DVLC1359,DVLC1360
      INTEGER DVLC1361,DVLC1362,DVLC1363,DVLC1364,DVLC1365,DVLC1366
      INTEGER DVLC1367,DVLC1368,DVLC1369,DVLC1370,DVLC1371,DVLC1372
      INTEGER DVLC1373,DVLC1374,DVLC1375,DVLC1376,DVLC1377,DVLC1378
      INTEGER DVLC1379,DVLC1380,DVLC1381,DVLC1382,DVLC1383,DVLC1384
      INTEGER DVLC1385,DVLC1386,DVLC1387,DVLC1388,DVLC1389,DVLC1390
      INTEGER DVLC1391,DVLC1392,DVLC1393,DVLC1394,DVLC1395,DVLC1396
      INTEGER DVLC1397,DVLC1398,DVLC1399,DVLC1400,DVLC1401,DVLC1402
      INTEGER DVLC1403,DVLC1404,DVLC1405,DVLC1406,DVLC1407,DVLC1408
      INTEGER DVLC1409,DVLC1410,DVLC1411,DVLC1412,DVLC1413,DVLC1414
      INTEGER DVLC1415,DVLC1416,DVLC1417,DVLC1418,DVLC1419,DVLC1420
      INTEGER DVLC1421,DVLC1422,DVLC1423,DVLC1424,DVLC1425,DVLC1426
      INTEGER DVLC1427,DVLC1428,DVLC1429,DVLC1430,DVLC1431,DVLC1432
      INTEGER DVLC1433,DVLC1434,DVLC1435,DVLC1436,DVLC1437,DVLC1438
      INTEGER DVLC1439,DVLC1440,DVLC1441,DVLC1442,DVLC1443,DVLC1444
      INTEGER DVLC1445,DVLC1446,DVLC1447,DVLC1448,DVLC1449,DVLC1450
      INTEGER DVLC1451,DVLC1452,DVLC1453,DVLC1454,DVLC1455,DVLC1456
      INTEGER DVLC1457,DVLC1458,DVLC1459,DVLC1460,DVLC1461,DVLC1462
      INTEGER DVLC1463,DVLC1464,DVLC1465,DVLC1466,DVLC1467,DVLC1468
      INTEGER DVLC1469,DVLC1470,DVLC1471,DVLC1472,DVLC1473,DVLC1474
      INTEGER DVLC1475,DVLC1476,DVLC1477,DVLC1478,DVLC1479,DVLC1480
      INTEGER DVLC1481,DVLC1482,DVLC1483,DVLC1484,DVLC1485,DVLC1486
      INTEGER DVLC1487,DVLC1488,DVLC1489,DVLC1490,DVLC1491,DVLC1492
      INTEGER DVLC1493,DVLC1494,DVLC1495,DVLC1496,DVLC1497,DVLC1498
      INTEGER DVLC1499,DVLC1500,DVLC1501,DVLC1502,DVLC1503,DVLC1504
      INTEGER DVLC1505,DVLC1506,DVLC1507,DVLC1508,DVLC1509,DVLC1510
      INTEGER DVLC1511,DVLC1512,DVLC1513,DVLC1514,DVLC1515,DVLC1516
      INTEGER DVLC1517,DVLC1518,DVLC1519,DVLC1520,DVLC1521,DVLC1522
      INTEGER DVLC1523,DVLC1524,DVLC1525,DVLC1526,DVLC1527,DVLC1528
      INTEGER DVLC1529,DVLC1530,DVLC1531,DVLC1532,DVLC1533,DVLC1534
      INTEGER DVLC1535,DVLC1536,DVLC1537,DVLC1538,DVLC1539,DVLC1540
      INTEGER DVLC1541,DVLC1542,DVLC1543,DVLC1544,DVLC1545,DVLC1546
      INTEGER DVLC1547,DVLC1548,DVLC1549,DVLC1550,DVLC1551,DVLC1552
      INTEGER DVLC1553,DVLC1554,DVLC1555,DVLC1556,DVLC1557,DVLC1558
      INTEGER DVLC1559,DVLC1560,DVLC1561,DVLC1562,DVLC1563,DVLC1564
      INTEGER DVLC1565,DVLC1566,DVLC1567,DVLC1568,DVLC1569,DVLC1570
      INTEGER DVLC1571,DVLC1572,DVLC1573,DVLC1574,DVLC1575,DVLC1576
      INTEGER DVLC1577,DVLC1578,DVLC1579,DVLC1580,DVLC1581,DVLC1582
      INTEGER DVLC1583,DVLC1584,DVLC1585,DVLC1586,DVLC1587,DVLC1588
      INTEGER DVLC1589,DVLC1590,DVLC1591,DVLC1592,DVLC1593,DVLC1594
      INTEGER DVLC1595,DVLC1596,DVLC1597,DVLC1598,DVLC1599,DVLC1600
      INTEGER DVLC1601,DVLC1602,DVLC1603,DVLC1604,DVLC1605,DVLC1606
      INTEGER DVLC1607,DVLC1608,DVLC1609,DVLC1610,DVLC1611,DVLC1612
      INTEGER DVLC1613,DVLC1614,DVLC1615,DVLC1616,DVLC1617,DVLC1618
      INTEGER DVLC1619,DVLC1620,DVLC1621,DVLC1622,DVLC1623,DVLC1624
      INTEGER DVLC1625,DVLC1626,DVLC1627,DVLC1628,DVLC1629,DVLC1630
      INTEGER DVLC1631,DVLC1632,DVLC1633,DVLC1634,DVLC1635,DVLC1636
      INTEGER DVLC1637,DVLC1638,DVLC1639,DVLC1640,DVLC1641,DVLC1642
      INTEGER DVLC1643,DVLC1644,DVLC1645,DVLC1646,DVLC1647,DVLC1648
      INTEGER DVLC1649,DVLC1650,DVLC1651,DVLC1652,DVLC1653,DVLC1654
      INTEGER DVLC1655,DVLC1656,DVLC1657,DVLC1658,DVLC1659,DVLC1660
      INTEGER DVLC1661,DVLC1662,DVLC1663,DVLC1664,DVLC1665,DVLC1666
      INTEGER DVLC1667,DVLC1668,DVLC1669,DVLC1670,DVLC1671,DVLC1672
      INTEGER DVLC1673,DVLC1674,DVLC1675,DVLC1676,DVLC1677,DVLC1678
      INTEGER DVLC1679,DVLC1680,DVLC1681,DVLC1682,DVLC1683,DVLC1684
      INTEGER DVLC1685,DVLC1686,DVLC1687,DVLC1688,DVLC1689,DVLC1690
      INTEGER DVLC1691,DVLC1692,DVLC1693,DVLC1694,DVLC1695,DVLC1696
      INTEGER DVLC1697,DVLC1698,DVLC1699,DVLC1700,DVLC1701,DVLC1702
      INTEGER DVLC1703,DVLC1704,DVLC1705,DVLC1706,DVLC1707,DVLC1708
      INTEGER DVLC1709,DVLC1710,DVLC1711,DVLC1712,DVLC1713,DVLC1714
      INTEGER DVLC1715,DVLC1716,DVLC1717,DVLC1718,DVLC1719,DVLC1720
      INTEGER DVLC1721,DVLC1722,DVLC1723,DVLC1724,DVLC1725,DVLC1726
      INTEGER DVLC1727,DVLC1728,DVLC1729,DVLC1730,DVLC1731,DVLC1732
      INTEGER DVLC1733,DVLC1734,DVLC1735,DVLC1736,DVLC1737,DVLC1738
      INTEGER DVLC1739,DVLC1740,DVLC1741,DVLC1742,DVLC1743,DVLC1744
      INTEGER DVLC1745,DVLC1746,DVLC1747,DVLC1748,DVLC1749,DVLC1750
      INTEGER DVLC1751,DVLC1752,DVLC1753,DVLC1754,DVLC17
```


Page 3:1

MDRL:9 00-000-01 UOM

1. NO. 301MRR<01V30> J

1000

WJAB

FOIA-1

3

```

C      #232 CALL MMLQ6(TKE,TATHR,TAN,TRSP,TIR)
C      #233 CALL VXTOA(TAM,TAN)
C      #234 CALL DCONZ(TE2,0,YEI,TAHR)
C      #235 CALL VMUL(TAMS,1,TAN,0,TEAC,0)
C      #236 CALL GAIN(TCFAC,TIG,TIUG,T72K,T22,T3H)
C      #237 CALL APC(TIUA,TMH,TSP0,TFILT,TCFAC,TMH,TVH,TQ1)
C      #238 CALL APC(TIUB,TMH,TSP0,TFILT,TCFAC,TMH,TVD,TQ1)
C      #239 CALL GTHR(TSMA,TIG,TIDG,TIM,TIC,TIF)
C      #240 CALL GTHW(TSMA,TIG,TIUG,TIM,TIC,TIA)
C      #241 CALL DEAL(RSMA,KCH,MUGH,RMH,KCH,KKH)
C      #242 CALL DEAL(RSRB,RCH,RDCH,RMH,RCH,KKH)
C      #243 CALL VMOV(RMH,KRH0)
C      #244 CALL SCLRES(RBH,KCH,RDCH,RUHA)
C      #245 CALL SCLRES(RWH,RCH,RDCH,RUWB)
C      #246 CALL VCTSY(RVH,RVHM,KKH,KKH)
C      #247 CALL PITSYM(RBH,RMH,RVH,RCH)
C      #248 CALL DFL22(KRHDA,RUCF0,KRH0,KRMV0)
C      #249 CALL DFL22(RRHUB,RUCF0,RRH0,RDMV0)
C      #250 CALL MPTBF(0)

```

```
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C C
C CDEFINE PENDING BUFFER, AND PREPARE NECESSARY FUNCTIONS
C C
C CPUT PROBING BUFFER AT $0000 = 32768. ON BUS1
C C    WITH SIZE $1000 = 6144.
C C DON'T USE BID 63, SINCE IT IS USED BY FTPLR & PFILR.
C CNOTE: FTPLR & PFILR EACH USE 2 PCB SLOTS.
C C
C C      PMBUF = 00
C C      APMBUF = 32768.
C C      SPMBUF = 6144.
C C
C C      CALL MFCBLN(LMUSL,PMBUF,APMBUF,SPMBUF,FID,CNTG,LNG)
C C
C CPUT PROBEOUND FUNCTIONS IN FCBS 0214 - 249
C C
C C      CALL MPCGF(PMBUF,214)
C C
C C ANALYSIS:
C C 0214      CALL TWIST(TSP1,TSP0,T61,T611,TRM,TRM0)
C C 0215      CALL DPRE(TSP0,TALPH,TSRA,TPHST,TOMES)
C C 0216      CALL OPRE(TSP0,TALPH,TSRB,TPHST,TOMES)
C C 0217      CALL SSUN(TDCH,TSP,TFSZ1)
C C 0218      CALL VMUL(TMSP,1,TSP,TDCH,THAMB,0)
C C 0219 & 0220      CALL FTPLR(TXSP,1,TMSFZ,TCUST,TKSP)
C C 0221      CALL VMSQ(TPSPB,TKSP,TKSPI)
C C 0222      CALL VLIN(TPSPI)
C C 0223 & 0224      CALL PFILR(TRP,1,TPSP,TCOST,TRP)
C C 0225      CALL PITCH(TMPP,TMM,TRP,T14,T14,TC2P)
C C 0226      CALL NEWL(TC,TINC,TRPD,TATHR,TRP,T0UNK)
C C 0227      CALL STAB(TCM,TC2P,TIC,TC)
C C 0228      CALL IMVPF(YE10,YMM,TSP1,TCM)
C C 0229      CALL VMUL(TME1,1,TE10,0,THAM1,0)
C C 0230      CALL MC0BZ(TMS,0,TOK1,TME1)
C C 0231      CALL AFC(TSP,TRS,TLMU)
C C
```


C <DCL10>DBM16F.FOR.1 Mon 18-Nov-88 6:38PM

C DBM16F: REAL TIME SYSTEM LOOP FUNCTION LIST.
C
CALL MWPL(BMPL)
CALL MPIFF(TSMXA,TOTW,ANLZA)
CALL MPIFF(RSTFB,RSNFA,SYNZA)
CALL MPIFF(TSRFB,TOTFA,ANLZH)
CALL MPIFF(RSTFA,RSNFB,SYNZH)
CALL MWPL(BMPL)

~L

C <DCL10>DBM16F.FOR.1 Mon 18-Nov-88 6:38PM

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C DEFINE UPPER LEVEL SYSTEM FUNCTION LISTS:

C DBM16F, BMPL, BMPLT, BMPLH, BMPLS

C DBM16F, BMPLT, BMPLH, BMPLS

C DBM16F: REAL TIME SYSTEM STARTUP FUNCTION LIST.

C

CALL MWPL(BMPL)

CALL MPIFF(TSMXA,TOTW,ANLZA)

CALL MPIFF(RSTFB,RSNFA,SYNZA)

CALL MPIFF(TSRFB,TOTFA,ANLZH)

CALL MPIFF(RSTFA,RSNFB,SYNZH)

CALL MWPL(BMPL)

CALL MWPL(BMPL)

~L

C <DCA16>RBM16F.FOR.1 Mon 18-Nov-88 6:38PM

C BMWFT: FILE-TD-FILE SYSTEM FUNCTION LIST.

```

CALL MPWFL(BMWFT)
CALL MPWT(PCS8,AP)
CALL ADINT
CALL RMINT
CALL MPITF(TSRFA,TSTF,ANLZA)
CALL MPITF(RSTF,RSNFA,SYNZA)
CALL MPITF(TSRFB,TSTFA,ANLZH)
CALL MPITF(RSTFB,RSNFB,SYNZB)
CALL MPAT(PCSMA,AP)
CALL TMINT
CALL DAINT
CALL MPFL(BUFF)

```

-L

C <DCA16>RBM16F.FOR.1 Mon 18-Nov-88 6:38PM

C BMWST: REAL TIME SYSTEM RESTANT FUNCTION LIST.

```

CALL MPWFL(BMWST)
CALL MPIS(RUN,1)
CALL MPIM(RUN,ME,M,BMMLP)
CALL MPFL(BMWST)

```

-L


```

C* CALL INVPF(TSP1,TM,TSP1,TCN)
  CALL MPBF(228)
C
C SPECTRAL FILTER ANALYSIS...
C* CALL WUL(TM1,1,TE18,B,THAM1,0)
  CALL MPBF(229)
C* CALL DCORZ(TMS,0,TE1,TMS1)
  CALL MPBF(230)
C* CALL WFC(TSP,TRE,TMU)
  CALL MPBF(231)
C* CALL M4L6(TAE,TXNB,TM,TSP,TIK)
  CALL MPBF(232)
C
C INVERSE SPECTRAL FILTER...
C* CALL VAYO(TAN,TAN)
  CALL MPBF(233)
C* CALL DCORZ(TX2,0,TE1,TAHR)
  CALL MPBF(234)
C
C GAMMA PROTECTION ON PREVIOUS FRAME PARAMETER DATA
C (HIDDEN UNDER DCORZ FUNCTION)...
  CALL PROP(8)
C
C NOISE SHAPING FILTER CALCULATION...
C* CALL WUL(TMS,1,TAN,B,TFAC,B)
  CALL MPBF(235)
C
C GAIN FACTOR CALCULATION...
C* CALL GAIN(TGAC,TIG,TIDC,T72R,TX2,TX)
  CALL MPBF(236)
C
C APC RESIDUAL CALCULATION ('TIU' IS EQUIVALENT TO START UP 'TSUKA')...
C* CALL APC(TIU,TM,TSP,TILT,TCFAC,TM,TM,TQI)
  CALL MPBF(237)
C
C ERROR PROTECTION ON PREVIOUS FRAME RESIDUAL DATA
C (HIDDEN UNDER APC FUNCTION)...
  CALL PROP(8)
  CALL MPBF(238)
C
C TRANSMITTER DATA COLLECTION...
C* CALL CTMR(TSNA,TIG,TIDC,TIN,TIC,TIK)
  CALL MPBF(239)
C
  CALL MPBF(240)

```

```

C <DCAL0>RHH16F.FOR.1 Mon 18-Nov-88 6:30PM
C
C ANALYSIS FROM TSP -> TSUKA
C* PROTECT FROM TSNA -> TOTA. (PROPAR & PROKES)
C
  CALL MPBF(241)
C
C SET G3 FOR SCOPE SYNC AT START OF TRANSMITTER
  CALL MPBF(242)
C
C MONITOR PEAK INPUT LEVEL
  CALL ADPEAK(TSRB)
C
C TRANSMITTER FRAME HISTORY BUFFER UPDATES...
C* CALL THIST(TSP1,TSP,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(243)
C
C PREEMPHASIS...
C* CALL DPRE(TSP,TALPH,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(244)
C
C DONE WITH TSP...
  CALL MPBF(245)
  CALL MPBF(246)
C
C PITCH FILTER ANALYSIS...
C* CALL SSUM(TDCM,TSP,TFSZ1)
  CALL MPBF(247)
C* CALL WUL(TSP,TSP,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(248)
C* CALL FTLR(TSP,1,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(249)
C* CALL WMSQ(TSP,TSP,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(250)
C* CALL VCLR(TSP,TSP)
  CALL MPBF(251)
C* CALL PFIL(TSP,1,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(252)
C* CALL PITCH(TSP,TSP,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(253)
C* CALL NEWL(TC,TINC,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(254)
C* CALL STAN(TC,TSP,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(255)
C
C INVERSE PITCH FILTER...
C* CALL INVPF(TSP,TM,TSP,TSP,TSP,TSP,TSP,TSP)
  CALL MPBF(256)
C
C SPECTRAL FILTER ANALYSIS...

```

```

C <UCAL6>BUN10F.FUN.1 Mon 18-Nov-88 6:38PM Page 11
C <UCAL6>BUN10F.FUN.1 Mon 18-Nov-88 6:38PM Page 11:1

C CALL VMUL(TWK1,1,TX10,0,TWAM1,0)
C CALL MPBFL(229)
C CALL DCONZ(TNS,0,TWK1,TWK1)
C CALL MPBFL(230)
C CALL MPC(TSP,TNS,TLMU)
C CALL MPBFL(231)
C CALL MLOG(TK,TXNR,TAM,TNSP,TIK)
C CALL MPBFL(232)

C INVERSE SPECTRAL FILTER...
C CALL VADN(TAM,TEN)
C CALL MPBFL(233)
C CALL DCONZ(TZ,0,TZ1,TAMR)
C CALL MPBFL(234)

C BMM PROTECTION ON PREVIOUS FRAME PARAMETER DATA
C (HIDDEN UNDER DCONZ FUNCTION)...
C CALL PROP(A)

C NOISE SHAPING FILTER CALCULATION...
C CALL VMUL(TAM,1,TAM,0,TZAC,0)
C CALL MPBFL(235)

C GAIN FACTOR CALCULATION...
C CALL GAIN(TCFAC,TIC,TIDC,TZ2,TZ3,T3R)
C CALL MPBFL(236)

C APC RESIDUAL CALCULATION ("TUB" IS EQUIVALENT TO START OF "TSHK")...
C CALL APC(TUB,TM,TSP,TZ1,TZ2,TCFAC,TZ,TM,T3)
C CALL MPBFL(238)

C ERROR PROTECTION ON PREVIOUS FRAME RESIDUAL DATA
C (HIDDEN UNDER APC FUNCTION)...
C CALL PHRES(A)
C CALL MPBFL(TWFA,1)

C TRANSMITTED DATA COLLECTION...
C CALL GTH(TSHK,TIC,TIDC,TM,TIC,TIK)
C CALL MPBFL(240)

C CALL MPBFL(AMZB)

C
~L

C <UCAL6>BUN10F.FUN.1 Mon 18-Nov-88 6:38PM Page 11:1
C <UCAL6>BUN10F.FUN.1 Mon 18-Nov-88 6:38PM Page 11

C SYNZA: SYNTHESIS FROM RSRA -> KSKA)
C DECODE/COMPAR FROM RTU -> KSHB.
C
C CALL MPBFL(SYNZA)
C
C CALL MPBFL(103,ICLR)
C
C RECEIVER DATA DISTRIBUTION...
C CALL DEAL(RSRA,RCH,ROCH,RNH,WCH,RKH)
C CALL MPBFL(241)
C
C RECEIVER FRAME HISTORY BUFFER UPDATE...
C CALL VMOV(RRH,RRH0)
C CALL MPBFL(243)
C
C RESIDUAL SCALING ("RUHA" IS EQUIVALENT TO START OF "RSRA")...
C CALL SCLRES(RNH,RCH,ROCH,RUHA)
C CALL MPBFL(244)
C
C SPECTRAL SYNTHESIS...
C CALL VLTYS(RNH,RVNH,RKH,KMH)
C CALL MPBFL(246)
C
C DECODING OF PREVIOUS FRAME RECEIVER RESIDUAL DATA &
C ERROR CORRECTION ON PREVIOUS FRAME RECEIVER PARAMETER DATA
C (PARTIALLY HIDDEN UNDER VLTYS FUNCTION)...
C CALL DECOD
C CALL CORPAR(N)
C CALL MPWT(PDCSH,AP)
C CALL MPST(RDTF,0)
C
C PITCH SYNTHESIS...
C CALL PITSYM(RKH,RNH,RVNH,RCH)
C CALL MPBFL(247)
C
C DEEMPHASIS ("RRHDA" IS EQUIVALENT TO "RSKA")...
C CALL DFL22(RRADA,RDCF,0,RRH,0,RDMY,0)
C CALL MPBFL(248)
C
C KSKA IS FULL...
C CALL MPWT(PDCSH,AP)
C CALL MPST(RSNVA,1)
C
~L

```

C
RETURN
END

C
C SYMB: SYNTHESIS FROM RSNB -> RSNKB
C DECOMA/CORPAR FROM RSTA -> RSR4.
C
C CALL MPBFL(SYNZB)
C
C CALL MPSC(IG3,ICLN)
C
C RECEIVER DATA DISTRIBUTION...
C CALL DEAL(RSNB,RCH,RDCH,RNH,RCH,RKH)
C CALL MPBFL(242)
C
C RECEIVER FRAME MISTERY BUFFER UPDATE...
C CALL VMDW(RNH,RNH)
C CALL MPBFL(243)
C
C RESIDUAL SCALING ('RNB' IS EQUIVALENT TO START OF 'RSRB')...
C CALL SCAL(RNH,RCH,RDCH,RNH)
C CALL MPBFL(245)
C
C SPECTRAL SYNTHESIS...
C CALL VTSY(RNH,RNH,RKH,RNH)
C CALL MPBFL(246)
C
C DECODING OF PREVIOUS FRAME RECEIVER RESIDUAL DATA &
C ERROR CORRECTION ON PREVIOUS FRAME RECEIVER PARAMETER DATA
C (PARTIALLY MISTERY UNDER VTSY FUNCTION)...
C DECOMA
C CALL COMP(1A)
C CALL MPBFL(PCS5,AP)
C CALL MPBFL(RNCPA,0)
C
C PITCH SYNTHESIS...
C CALL PITSY(RNH,RNH,RNH,RNH)
C CALL MPBFL(247)
C
C RESYNTHESIS ('RNB' IS EQUIVALENT TO 'RSNB')...
C CALL DFL22(RNDB,RNCPA,RNH,RNH,RNH)
C CALL MPBFL(249)
C
C RSNB IS FULL...
C CALL MPBFL(PCS5,AP)
C CALL MPBFL(RNCPA,1)
C
C CALL MPBFL(SYNZB)

-L

```

C  <CALL>>BIBN10H.FUR.1  Mon 10-Nov-88  6:48PM
C
C
C  INTEGER FUNCTION ADINT(I)
C
C  C  FCB 8 124
C  ADINT=INTFN(124,8)
C  RETURN
C  END
C
C
C  INTEGER FUNCTION ADPEAK(U)
C
C  USAGE: IER = ADPEAK(U)
C
C  INTEGER U,FCBCH
C  ADPEAK = FCBCH(102,8,0,U,0,0,0,0,0,10)
C  RETURN
C  END
C
C
C  INTEGER FUNCTION APC(V,A,U,V,W,R,S,T)
C
C  USAGE: IER = APC(V,A,U,V,W,R,S,T)
C
C  NOTES: V,U,V,W,R,S,T ARE BUFFERS; A IS SCALAR
C  FUNCTION REQUIRES SPECIAL BINDING
C
C  INTEGER V,A,U,V,W,R,S,T,FCBCH
C  APC = FCBCH(-199,V,A,U,V,W,R,S,T,14)
C  RETURN
C  END
C
C
C  INTEGER FUNCTION CORPAR(IAB)
C
C  IAB=-2 -> CORRECT MATA INTO MSRA
C  IAB=0 -> CORRECT MATA INTO MSNA
C  FCB 8123
C
C  COMPAR=INTFN(123,IAB)
C  RETURN
C  END
C
C
C  INTEGER FUNCTION DAINT(I)
C

```

```

C <DCL16>*****FOR-1  Mon 18-Nov-88 01:40PM Page 1
C *****
C (*****DCL16>*****FUNK.1, 18-NOV-88 10:40:30, Ed: KFL1D
C *****
C C THIS FILE CONTAINS MAP-300 SHAPII MOST SUPPORT FUNCTIONS
C C USED BY THE NEW DCL16 SYSTEM.
C C FUNCTIONS ARE ORDERED ALPHABETICALLY.
C *****
C *****
C *****
C *****
C *****
C *****

```


407

C <UCALB>BUN16H.FOR.1 Mon 18-Nov-88 6:48PM

Page 2:7

C <UCALB>BUN16H.FOR.1 Mon 18-Nov-88 6:48PM

```

C
C
C      INTEGER V,A,U,B,C,D,PCBCH
C      IOUT=V
C      PITCH=PCBCH(134,V,A,B,C,D,PCBCH,C,IOUT,B,14)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION PITCH(V,A,U,V)
C
C      MOST SUPPORT FOR PITCH
C      USAGE: IERR=PITCH(V,A,U,V)
C
C      INTEGER V,A,U,V,PCBCH
C      PITCH=PCBCH(203,V,A,B,C,D,V,B,D,B,9)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION PROPAB(IAB)
C
C      IAB--2 => PROTECT (PARAMETERS) TSKA INTO TOTA
C      IAB--0 => PROTECT (PARAMETERS) TSKB INTO TOTA
C      PCB #122
C
C      PROPAB=INTFN(122,IAB)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION PROMES(IAM)
C
C      IAB--2 => PROTECT (RESIDUAL) TSKA INTO TOTA
C      IAB--0 => PROTECT (RESIDUAL) TSKB INTO TOTA
C      PCB #121
C
C      PROMES=INTFN(121,IAB)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION RMINT(I)
C
C      PCB # 126
C      RMINT=INTFN(126,I)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION SCLRES(V,A,U,V)
C
C      MOST SUPPORT FOR SCLRES
C      USAGE: IERR=SCLRES(V,A,U,V)
C
C      INTEGER V,A,U,V,PCBCH
C      SCLRES=PCBCH(202,V,A,U,B,C,D,B,D,B,9)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION STAB(V,A,U,V)
C
C      MOST SUPPORT FOR STAB
C      USAGE: IERR=STAB(V,A,U,V)
C
C      INTEGER V,A,U,V,PCBCH
C      STAB=PCBCH(150,V,A,U,B,C,D,B,D,B,9)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION THIST(V,U,V,M,R,S)
C
C      USAGE: IER = THIST(V,U,V,M,R,S)
C
C      NOTE: ALL AMGS ARE BUFFERS: SPECIAL HANDLING IS REQ'D.
C
C      INTEGER V,U,V,M,R,S,PCBCH
C      THIST = PCBCH(-241,V,U,V,M,R,S,B,D,B,3)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION TINT(I)
C
C      PCB # 125
C      TINT=INTFN(125,I)
C      RETURN
C      END
C
C
C      INTEGER FUNCTION RETOA(V,B)

```

```

C <OCAL>SUBROUTINE FOR-1 MON 18-MAY-80 6:48PM
C MUST SUPPORT FOR VETUA
C USAGE: IERR=VETOA(V,B)
C
C   INTEGER V,B,PCOCH
C
C   VETOA=PCOCH(13,V,B,B,B,B,B,B,11)
C   RETURN
C   END
C
C   INTEGER FUNCTION VETSV(V,B,V,B)
C
C   USAGE IER = VETSV(V,B,V,B)
C
C   NOTE: USE CALLING SEQUENCE 017 (V,B,V,B), WITH A=0
C
C   INTEGER PCOCH,V,B,V,B
C   VETSV = PCOCH(132,V,B,B,B,B,B,B,17)
C   RETURN
C   END

```

[illegible]

[illegible]


```

C <MAP>HIST16.FOR.7   Tue 30-Dec-80 9:27AM           Page 1

C (ENDMD)C<MAP>HIST16.FOR.7, 30-Dec-80 09:27:12, 843 MILE
C HIST16 --> PRINT OUT HISTOGRAM DATA FROM DCA16 SPEECH CODEX

      INTEGER HISTO,FID,CHTC,LNG,BUS1
      INTEGER M(260),G(64),K(64),C1(2),C2(16),C3(8)
      INTEGER K1(64),K2(32),K3(16),K4(16),K5(16),K6(16)
      EQUIVALENCE (M(1),G(1)),(M(65),G(1)),(M(89),C1(1)),
1 (M(77),C2(1)),(M(93),C3(1)),(M(101),K1(1)),(M(165),K2(1)),
2 (M(197),K3(1)),(M(213),K4(1)),(M(229),K5(1)),(M(245),K6(1))
      DATA HISTO,BUS1,FID,CHTC,LNG/1,64,2,1,8/
      DATA HISTO,SHISTO/39944.,260./

      CALL WPUH(3)
      TYPE 1
      FORMAT(' FILE (OR TT) FOR HISTOGRAM: 'S)
      CALL ASSIGN(2,M(1),-1)
      CALL WPLD(BUS1,HISTO,SHISTO,FID,CHTC,LNG)
      CALL WPMU(HISTO,M(1),2,0,M(260))
      NPM=0
      DO 5 I=1,4
        NPM=NPM+NG(I)
      WRITE(2,10) NPM
10  FORMAT(' TOTAL FRAMES = ',16//,' M 6 DG C1 C2',
1 - C3 K1 K2 K3 K4 K5 K6//)
      DO 15 I=1,4
        J=I-1
15  WRITE(2,16) J,G(1),G(1),C1(1),C2(1),C3(1),K1(1),K2(1),
1 K3(1),K4(1),K5(1),K6(1)
16  FORMAT(13,1116)
      DO 20 I=5,8
        J=I-1
20  WRITE(2,21) J,G(1),C1(1),C2(1),C3(1),K1(1),K2(1),
1 K3(1),K4(1),K5(1),K6(1)
21  FORMAT(13,16,61,916)
      DO 25 I=9,16
        J=I-1
25  WRITE(2,26) J,G(1),C2(1),K1(1),K2(1),
1 K3(1),K4(1),K5(1),K6(1)
26  FORMAT(13,16,61,16,61,716)
      DO 30 I=17,32
        J=I-1
30  WRITE(2,31) J,G(1),K1(1),K2(1)
31  FORMAT(13,16,4(61),216)
      DO 35 I=33,64
        J=I-1
35  WRITE(2,36) J,G(1),K1(1)
36  FORMAT(13,16,4(61),16)
      WRITE(2,40)
      FORMAT(//)
      CALL CLOSE(2)
      STOP
      END

```

DATE
FILMED
-8