

AD-A086 134

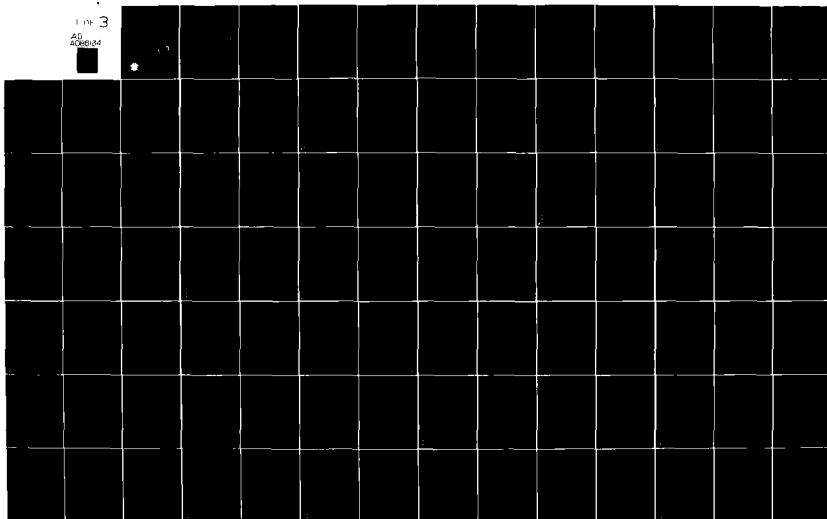
NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/0 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(!!!)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED

NL

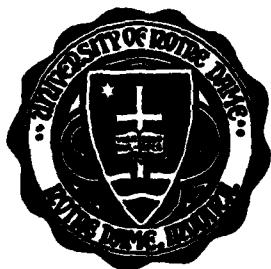
1 OF 3

AD-A086 134



ADA 086134

DDC FILE COPY



LEVEL

12

FINAL REPORT
VOLUME 2

DESIGN AND IMPLEMENTATION
OF A SPEECH CODING ALGORITHM
AT 9600 B/S

DTIC
1 1980

DTIC
ELECTE
JUL 1 1980
S C D

Department of

ELECTRICAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

This document has been approved
for public release and sale; its
distribution is unlimited.

80 6 30 191-

12

FINAL REPORT
VOLUME 2
DESIGN AND IMPLEMENTATION
OF A SPEECH CODING ALGORITHM
AT 9600 B/S

DTIC
JUL 1 1980
C

Prepared for

Defense Communications Agency
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia 22090

Contract No. DCA 100-79-C-0005

30 April 1980

This document has been approved
for public release and sale; its
distribution is unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
	AD-A086 134	
4. TITLE (and Subtitle)		5. TYPE OF REPORT & PERIOD COVERED
Design and Implementation of a Speech Coding Algorithm at 9600 B/S.		Final Report Nov 1978 - April 1980
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s)
James L. Melsa, et al.		DCA 100-79-C-0005
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Department of Electrical Engineering University of Notre Dame Notre Dame, IN 46556		
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE
Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305		April 1980
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. NUMBER OF PAGES
		519
		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 Commerce, 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, 9600 bps speech transmission, pitch extraction, adaptive residual coder, waveform reconstruction.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
→ This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system. The algorithm combines: Pitch extraction loop, Pitch compensating adaptive quantizer, Sequentially adaptive linear predictor, Adaptive source coding,		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

388467

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

✓ to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G. ✓

Accession For	
NTIS GR&I	☒
DDC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Special

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ABSTRACT

This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system.

The algorithm combines

- Pitch extraction loop
- Pitch compensating adaptive quantizer
- Sequentially adaptive linear predictor
- Adaptive source coding

to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G.

PROJECT PERSONNEL

Arvind Arora, research assistant

David L. Cohn, co-principal investigator

James M. Kresse, research assistant

James L. Melsa, principal investigator

Arun K. Pande, research assistant

Maw-lin Yeh, research assistant

FINAL REPORT
DCA CONTRACT 100-79-C-0005

	<u>Page</u>
Abstract	i
Project Personnel	ii
1. Introduction and Outline of Report	1
1.1 Introduction	1
1.2 Summary of Algorithm Requirements	3
1.3 Outline of Report	4
2. Algorithm Description	5
2.1 Introduction	5
2.2 Transmitter Buffer System	11
2.3 Adaptive Low-Pass Filter	16
2.4 Pitch Extraction	18
2.5 Adaptive Residual Coder	20
2.5.1 Adaptive Predictor	20
2.5.2 Adaptive Quantizer	22
2.6 Pitched Repetition	26
2.7 Noiseless Source Coder	28
2.8 Receiver	34
3. Synchronization	38
3.1 Sync. Algorithm Description	40
3.1.1 Synchronization Acquisitions	40
3.1.2 Synchronization Monitor	43
4. Pitch Extraction Studies	46
4.1 Introduction	46
4.2 Pitch Extraction Algorithms	47
4.3 Redundancy Removal	54
4.4 References	61
5. Tree Coding	62
5.1 Introduction	62
5.2 The (M,L) Algorithm	63
5.2.1 Description	63
5.2.2 Results	65
5.3 Adaptive Tree Coding	69
5.3.1 Description	69
5.3.2 Results	71
5.4 Conclusions and Suggestions for Further Research	75
5.5 References	77

6. Backward PARC	78
6.1 Introduction	78
6.2 System Structure	80
6.3 Performance Evaluation and Parametric Studies	85
6.4 Transmission Error Studies	90
6.5 Conclusions	95
7. Tandem Operation	96
7.1 Introduction	96
7.2 PARC in Tandem with CVSD	97
7.3 Effect of Background Noise on PARC Performance	100
7.4 References	103
8. Transmission Errors	104
8.1 Introduction	104
8.2 Simulation of Transmission Error	105
8.3 Minimizing Transmission Error Effects	108
8.4 Conclusion	112
8.5 Reference	113
9. Filtering	114
9.1 Introduction	114
9.2 Evaluation of Pre-emphasis	115
9.3 Filter Selection	118
9.4 Results	121
9.5 Low-Pass Filtering vs. Entropy	123
9.6 Results and Conclusions	125
9.7 References	128
9.A Derivation of Filter	129
10. Buffer Control	136
10.1 Introduction	136
10.2 Overflow control	137
10.3 Underflow control	138
10.4 Special considerations at the receiver	139
10.5 Conclusions and suggestions for further research	140
11. Source and Error Control Coding	141
11.1 Introduction	141
11.2 Source Coding	142
11.2.1 Quantizer levels	142
11.2.2 Side information	144
11.3 Error Control Coding	145
11.4 Conclusion and suggestions for further research	146
A. FORTRAN Simulation of Algorithm	147

B.	Segmented SNR Plots	198
B.1	Introduction	198
B.2	Listings	199
C.	Plotting Program	242
D.	PDP-11 D/A Programs	247
12.	MAP Implementation	263
12.1	Introduction	263
12.2	PARC - Transmitter	278
12.2.1	The Pitch Computation Program	281
12.2.2	The Speech Digitization Program	288
12.3	PARC - Receiver	295
12.4	Noiseless Source Coder	302
12.5	Synchronizer, Decoder	311
12.5.1	Synchronization Acquisition Module	313
12.5.2	Decoder Module	315
12.5.3	Initialization Module	318
12.5.4	Decoder Constraints	319
12.6	Program Timing and Speed	320
E.	Resampling Program	322
E.1	Operating Details	325
F.	CVSD Algorithm	344
F.1	The CVSD System	344
F.2	The Real-Time CVSD System	346
F.2.1	Design Overview	346
F.2.2	The Initialization Step	348
F.2.3	The Processing Step	351
F.3	Conclusion	355
F.4	Reference	355
G.	Program Listings for Real Time Implementation of PARC on MAP-300	370

CHAPTER 12

THE REAL-TIME IMPLEMENTATION

12.1 Introduction

In this chapter, the final form of the real-time implementation of the PARC algorithm in a full duplex mode will be presented. The whole implementation will be decomposed into four parts. Each part is described in detail in the following sections. Block diagrams, flow charts and timing diagrams are given to help reader understand the implementation. The program listings are in Appendix G.

Before presenting the implementation, the system components used will be described. A MAP-300 (Macro Arithmetic Processor) produced by Computer Signal Processing, Inc. is used to implement the speech coding algorithm (see Table 12.1). A host computer and MAP-300 system block diagram is shown in Fig. 12.1. The MAP-300 consists of six programmable processing elements as well as memory and peripherals:

- A. A Central Processing Unit (CSPU) functions as the executive controller by interpreting commands from the host computer, setting up and scheduling the other processors, and performing arithmetic and logic operations.
- B. An Arithmetic Processor (AP) consists of two subprocessors. An Arithmetic Processing Unit (APU) carries out the floating point arithmetic calculations. An Arithmetic Processing Scroll (APS) is a data addressing device for the APU.
- C. An Input/Output Scroll (IOS-2) transfers bit streams into or out of a modem.
- D. An Analog Data Acquisition Module (ADAM) samples and quantizes input analog signals into digital signals and stores them into main memory.

Table 12.1
MAP-300 Speech Processing System

Item No.	Qty.	Model Number	Description
1	1	1030	MAP-300 Processor
2	1	2030	8K x 32 MOS Memory
3	1	2050	16K x 32 MOS Memory
4	1	2203	8K x 32 MOS Memory
5	1	2120	2K x 32 Bipolar Memory
6	1	3100	PDP-11 Interface
7	1	4020	Model 2SM/ I/O Scroll
8	2	4040	Bus Switch
9	1	5120	Analog Data Acquisition Module
10	1	5130	Analog Output Module
11	1	6100	Expansion Chassis
12	1	6200	Auxiliary Power Supply
13	1	03-1360585	GTE Speech Processing Interface

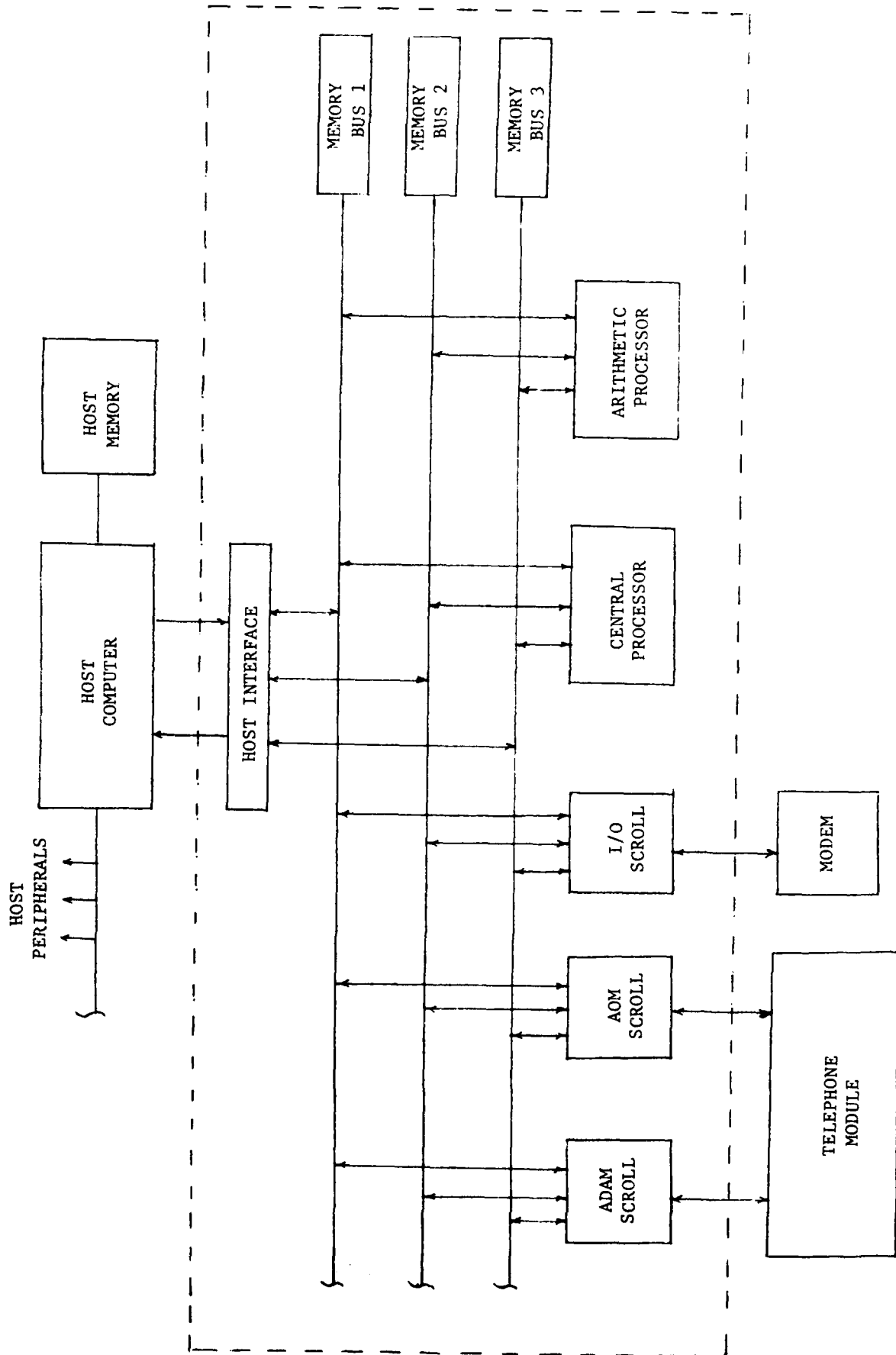


Fig. 12.1 MAP-300 System Block Diagram

- E. An Analog Output Module (AOM) converts input digital signals into analog signals and outputs them to a telephone module.
- F. A Host Interface Scroll (HIS) acts as a communication and data transfer medium between the host computer and the MAP-300 system.

This array processor system provides two classes of software. The first class is the software designed specifically for array processing. It is an executive-driver-subroutine package resident in the MAP-300 and in the host computer which permits direct calling of routines to perform arithmetic operation, to manage MAP memory, to define even higher level routines. A MAP user can execute array processing operations with simple Fortran calls in the host computer. The sophisticated programmer even can implement his own processing algorithms in MAP assembly languages. To implement the PARC digitization system, some new array and non-array functions, which are described later, are employed. The second class consists of utility programs including a cross-assembler and a cross-simulator. These may be used to add new routines to the array processing library.

The array processor can be used as a peripheral to a host computer, using host peripherals for data storage and interface for data and commands transfer between them. However, after initialization, it can stand alone by using proper control functions in the host.

The underlying design principles of the real-time implementation are as follows: First, in order to study system performance, the flexibility of changing system parameters such as speech algorithm parameters, buffer sizes and buffer starting locations must be provided. Second, in order to be in the real-time mode, the arithmetic execution time and the overhead time have to be reduced. The arithmetic execution time can be

decreased by efficiently utilizing the arithmetic processors. The overhead time can be decreased by utilizing all processors as asynchronously and as simultaneously as possible. However, the necessary synchronization between processors has to be carefully considered.

After several months of study, the final real-time PARC communication system consists of the following programs:

1. Host programs: Fortran PARC main program
Fortran PARC host support program
2. AP programs: PARC-transmitter pitch computation program
PARC-transmitter speech digitization program
PARC-receiver
3. CSPU programs: Encoder program
Decoder program
Synchronization program
Transmitter buffer pointers update program
Receiver buffer pointers update program
4. IOS programs: ADAM program
ADM program
IOS-2 program

The main function of the host Fortran programs is to initialize the MAP-300 system with the real-time PARC algorithm, to set up the proper command sequences and to initiate them. After that, the MAP-300 system can execute the PARC without any further commands from the host computer. The two PARC-transmitter programs and the PARC-receiver program require sophisticated arithmetic operations and will be executed in the AP. Those programs requiring logic and bit operations include encoder, decoder, synchronizer and address pointers update programs. These will

be executed in the CSPU. A/D converting, D/A converting and bit-transfer will be executed in the ADAM, the AOM and the IOS-2 respectively. These five processors operate in parallel. The host Fortran modules will be described below; they present the whole picture of the PARC. The rest of the programs will be presented in the following sections.

The flow chart of the real-time PARC main program is shown in the Fig. 12.2. The module can be divided into two steps: An initialization step and a processing step. In each step, each processor has its own role.

1. The initialization step:

In this step, the host computer will initialize the whole algorithm by reading in system parameters, configuring logical buffers, initializing logical buffers, loading ADAM and AOM, and creating appropriate command sequences called function lists. The logical buffers and the function lists, which are fairly complicated, will be described later. In this step, only three processors in the MAP-300 system have been used as follows:

- A. The CSPU acts as the resource controller. Responding to a host command, it loads an A/D program to the ADAM, loads a D/A program to the AOM, configures the logical buffers, and loads appropriate arithmetic functions into the AP.
- B. The AP executes these arithmetic functions which will reset the logical buffers.
- C. The HIS acts as an interface between the host computer and the MAP-300.

The Fortran host support program, which is the host support module for those new array and non-array functions in the real-time PARC

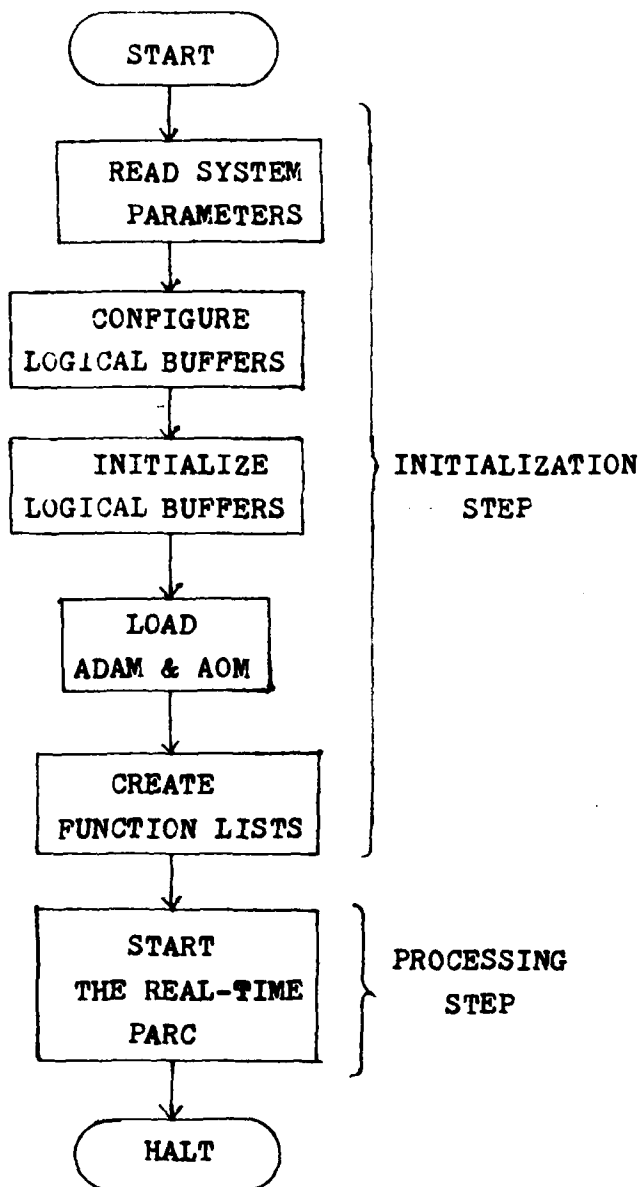


Fig. 12.2 The Flow Chart of the Main Program

algorithm, interfaces the main program to the host/MAP driver module.

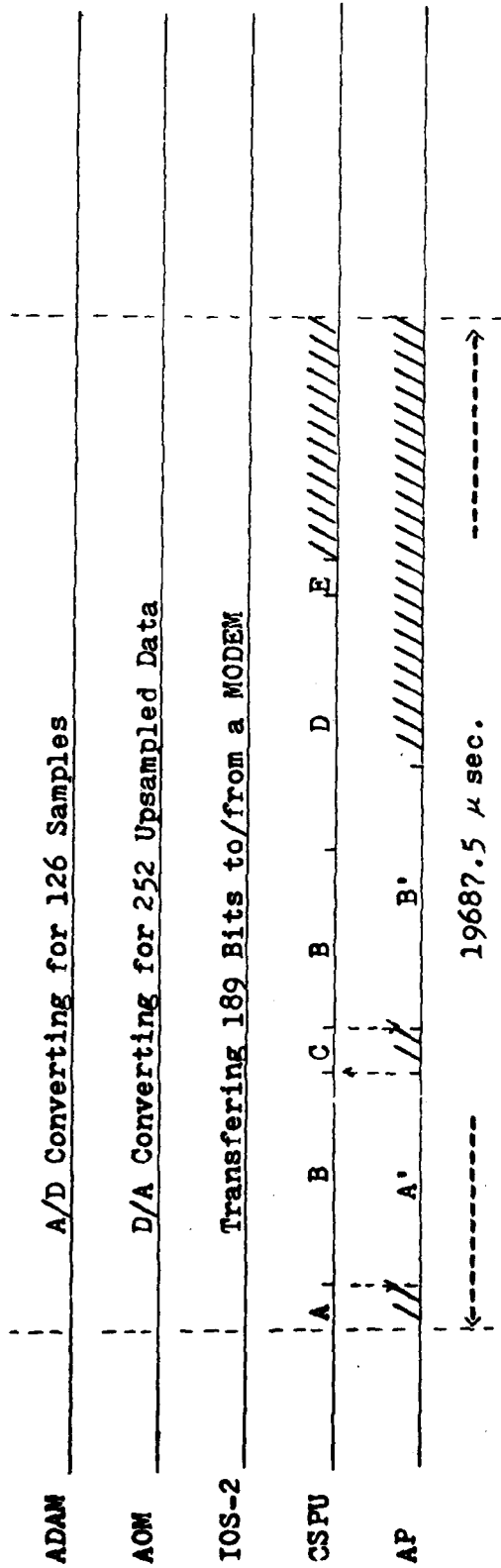
The main functions of this support program are as follows:

- 1) Check each argument for the proper range and return error code.
- 2) Pack arguments for the MAP in a Function Control Block (FCB)
- 3) Determine any host absolute addresses that must be evaluated while constructing the Data Control Block (DCB).
- 4) Pass DCB to the host/MAP driver.

2. The processing step:

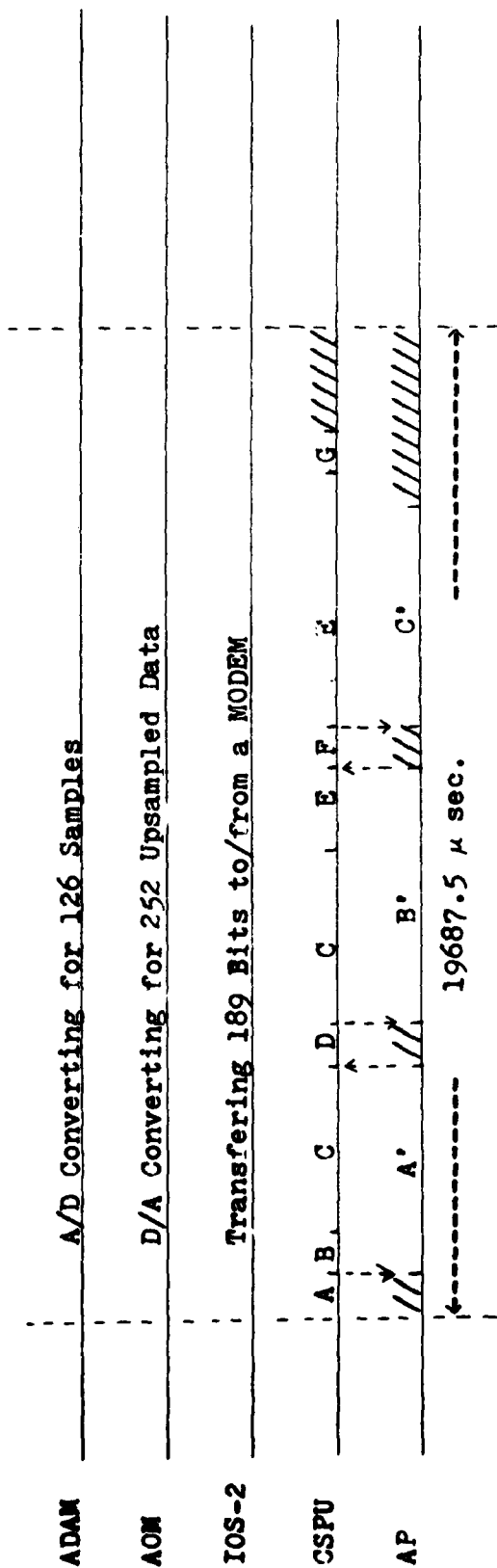
After initialization, the MAP-300 can execute the real time PARC algorithm without any further commands from the host computer. Because of the fixed delay for the whole system (refer to Section 2.2), a fixed time slot has been chosen as a main parameter to synchronize all the processors in the MAP-300 system as shown in Fig. 12.3 and Fig. 12.4. In each time slot, a particular function list, which is described below, will be executed. This fixed time slot is set to be 19687.5 microseconds which is precisely the period for ADAM to process 126 samples, the period for AOM to process 252 upsampled data, and the period for IOS-2 to process 189 bits.

The function lists, whose relationships are shown in Fig. 12.5, are eight sets of command sequences. When the real-time PARC is initiated, the first function list, which starts the ADAM, the AOM and the IOS-2, is executed. Afterward, the main control function list (the second function list) is executed by the CSPU. This function list assures the synchronization of the receiver. The execution of the list is as follows:



- A: Load the APU and the APS with the Pitch Computation Program
- B: Encoder
- C: Load the APU and the APS with the Speech Digitization Program
- D: The Channel Synchronization Program
- E: Update the PARC-Transmitter Address Pointers
- A': The Pitch Computation Program
- B': The Speech Digitization Program

Fig. 12.3 The Timing Diagram for the Synchronization Operation



- A: Load the APU and the APS with the Pitch Computation Program
- B: Update the PARC-Receiver Address Pointers
- C: Encoder
- D: Load the APU and the APS with the Speech Digitization Program
- E: Decoder
- F: Load the APU and the APS with the PARC-Receiver Program
- G: Update the PARC-transmitter Address Pointers
- A': The Pitch Computation Program
- B': The Speech Digitization Program
- C': The PARC-Receiver Program

Fig. 12.4 The Timing Diagram for the Normal PARC Operation

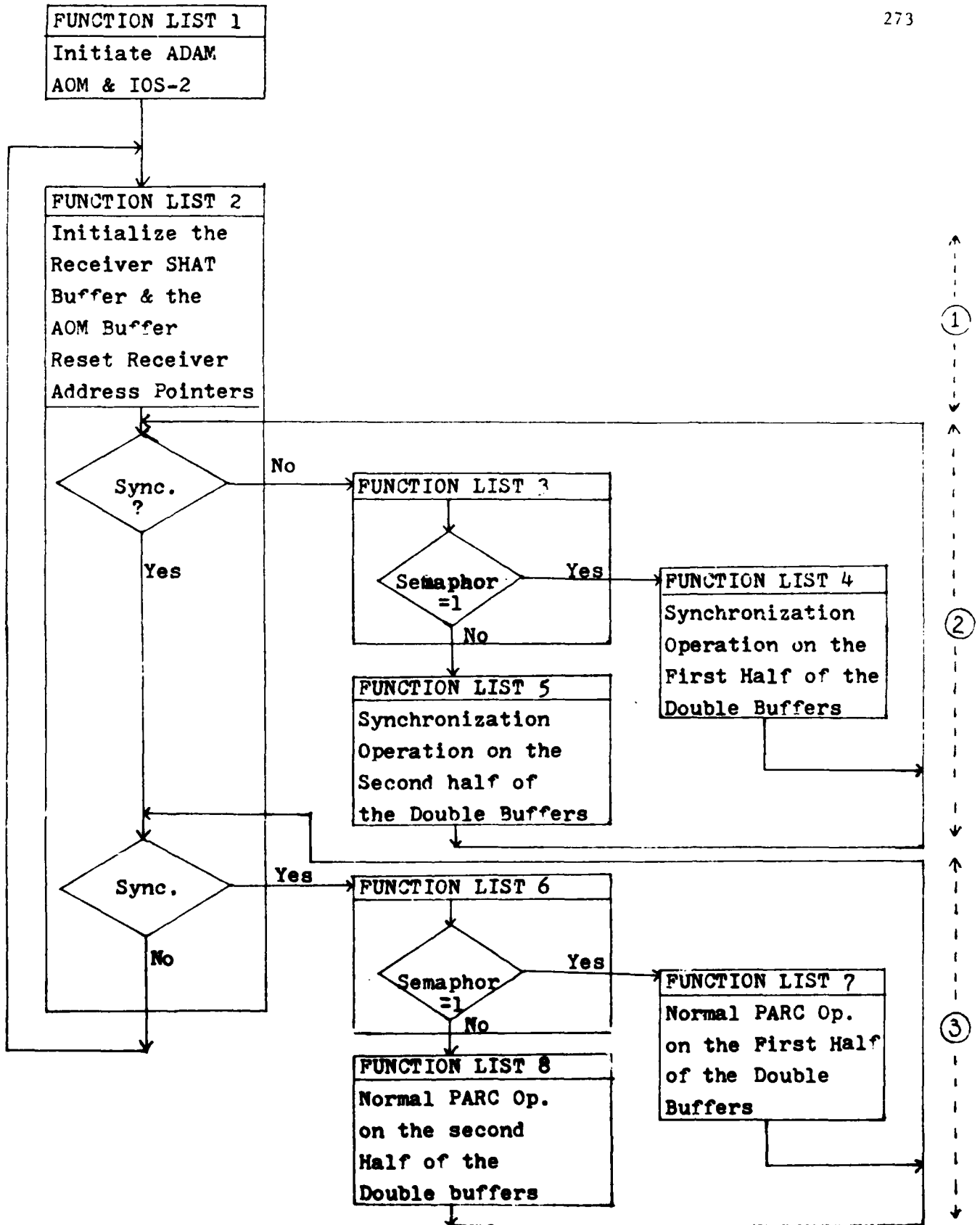


Fig. 12.5

The Flow Chart of Function Lists

1. It initializes the receiver SHAT buffer and the AOM buffer.
Also, it resets the receiver address pointers.
2. If the channel synchronization does not exist, it keeps executing the synchronization operation until a channel synchronization is set. The synchronization operation consists of the PARC-transmitter, the encoder and the synchronizer.
3. If the channel synchronization exists, it keeps executing the normal PARC operation until a channel synchronization is lost.
The normal PARC operation consists of the PARC-transmitter, the encoder, the decoder and the PARC-receiver.
4. Go to 1.

Because of parallel data processing which speeds up the whole system (refer to the Section 2.2), the double buffering technique is employed. Therefore, the synchronization operation and the normal PARC operation each consists of three function lists: A control function list, a function list which operates on the first half of the double buffers and a function list which operates on the second half of the double buffers. The control function list checks a semaphore which resides in the MAP memory and then executes a function list which will operate on the proper buffers. The reason for this is to have a continuity of data flow for the PARC-transmitter. For instance, suppose the receiver loses the channel synchronization after the normal PARC operation on the first half of the double buffers. Then the main control function list executes the synchronization operation. The control function of the synchronization operation checks the semaphore and then executes the synchronization operation on the second half of the double buffers. So, the transmitter

gets a continuous data flow. The main control function list will keep executing until the MAP-300 system is terminated.

In the synchronization operation, which consists of the function lists 3, 4 and 5 (shown in Fig. 12.3), the role of each processor in a time slot is described as follows:

- 1) The ADAM samples and quantizes the input analog signal from the telephone module and stores these samples into MAP main memory. In a time slot, it produces 126 quantized samples.
- 2) The AOM acts as a D/A converter and outputs 252 silence samples since no information flows into the receiver.
- 3) The IOS-2 acts as an input/output interface between the MAP-300 and the channel modem. In a time slot, it transfers 189 bits out of the encoder bit buffer and reads 189 bits into the decoder bit buffer.
- 4) The CSPU functions as follows in sequence:
 - A: It loads the AP with the pitch computation program.
 - B: It executes the encoder. However, upon request from an APU interrupt, it suspends the current operation and
 - C: Loads the AP with the speech digitization program.
It restarts the encoder after the loading.
 - D. It executes the channel synchronization program.
 - E. It updates the PARC-transmitter address pointers.
- 5) The AP executes the arithmetic part of the PARC-transmitter. After finishing each program, it will interrupt the CSPU to indicate that it is free for the next operation, if any. In this operation, the AP has two jobs:

A: The pitch computation program.

B: The speech digitization program.

These five processors function in parallel. The ADAM, AOM and IOS-2 operate continuously without any interrupt. However, the CSPU and the AP have to finish their own job inside the time slot in order to be in real-time.

The normal PARC operation (shown in Fig. 12.4) which consists of the function lists 6, 7 and 8, has the same ADAM and IOS-2 operation as the synchronization operation. However, the CSPU, the AOM and the AP have different operations as follows:

- 1) The ADAM samples and quantizes the input analog signal from telephone module and stores these samples into MAP main memory. In a time slot, it produces 126 quantized samples.
- 2) The AOM acts as a D/A converter and outputs 252 unsampled reconstructed speech samples to the telephone module.
- 3) The IOS-2 acts as an input/output interface between the MAP-300 and the channel modem. In a time slot, it transfers 189 bits out of the encoder bit buffer and reads 189 bits into the decoder bit buffer.
- 4) The CSPU functions as follows in sequence:
 - A: It loads the AP with the pitch computation program.
 - B: It updates the receiver address pointers.
 - C: It executes the encoder. However, upon request from an APU interrupt, it suspends the current operation and
 - D: Loads the AP with the speech digitization program. It restarts the encoder after the loading.

- E. It executes the decoder. Upon request from an APU interrupt, it suspends the current operation and
 - F. Loads the AP with the PARC-receiver program. It restarts the decoder after the loading.
 - G. It updates the transmitter address pointers.
- 5) The AP executes the PARC-transmitter and the PARC-receiver. After finishing each program, it will interrupt the CSPU to indicate that it is free for the next processing, if any. In this operation, the AP has three jobs:
- A: The pitch computation program.
 - B: The speech digitization program.
 - C: The PARC-receiver program.

In this operation, the timing control is very important which will be discussed in the next section.

The subsequent sections will describe each of the subsystems in the real-time PARC system. The next section explains the design of the PARC-transmitter. The complicated buffer system, buffer control and the speech digitization will also be described. The corresponding noiseless source encoder will be explained in Section 12.4.

Section 12.3 describes the design of the PARC-receiver. Although some parts of the system are the same as in the transmitter, the different design of the buffering and upsampling will be depicted. The corresponding noiseless source decoder will be explained in Section 3.5.

12.2 The PARC-Transmitter

In this section, the main line of the real-time PARC-Transmitter, which resides in the AP, will be presented. The corresponding encoder will be discussed in Section 12.4. The block diagram of the PARC algorithm is shown in Fig. 2.2 and discussed in Chapter 2.

Because of the limit on the size of the AP memory, the whole PARC-Transmitter algorithm cannot be implemented as a single AP program. So, the PARC-Transmitter has to be cut into two subprograms, namely, the pitch computation program and the speech digitization program, as shown in Fig. 12.6. Those portions of the algorithm which operate on blocks of speech samples will be put in the pitch computation program. Those portions include: input gain factor calculation, system noise reducer, adaptive filter and pitch extractor. These elements of the algorithm which process speech sample-by-sample will be combined into the speech digitization program. These elements are the adaptive quantizer, the inverse quantizer, the adaptive predictor and the pitch extraction.

Before any further description of the design of this PARC-Transmitter, the buffer system has to be depicted. There are seven buffers employed by the PARC-Transmitter as shown in Table 12.2.

Among them, ADAM, LEVEL and BIT buffers are double buffers which allow one processor to fill one half of the buffer while another processor processes on the other half. The SAMPLE, VHAT and SHAT buffers are circular buffers which allow continuous access. However, the additional address pointers, which indicate the starting locations of those circular buffers at the beginning of a time slot, have to be considered. The PARAMETER buffer is a single buffer which stores of system parameters such as predictor coefficients, quantizer output

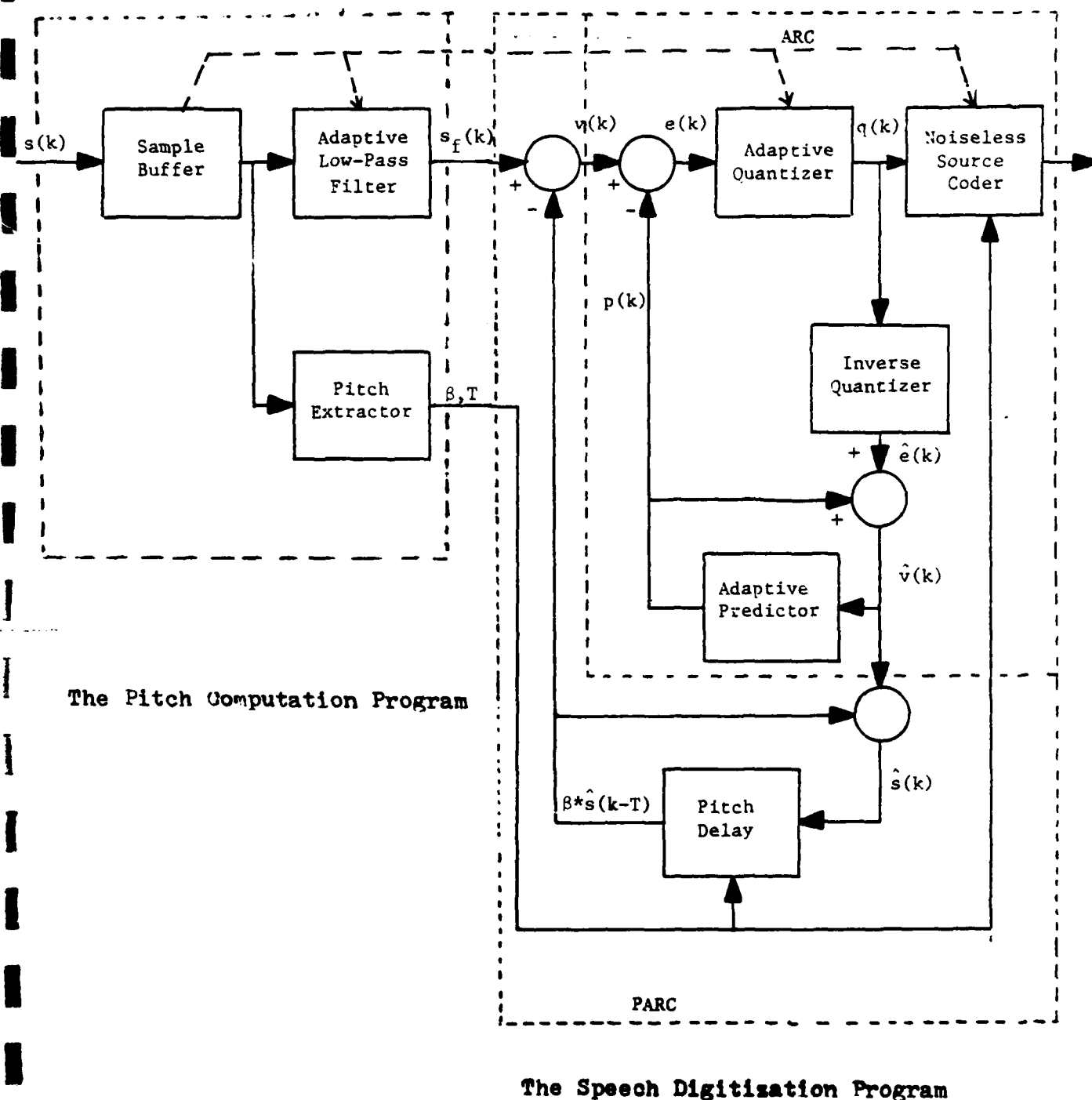


Fig. 12.6 THE BLOCK DIAGRAM OF THE PARC

Table 12.2 Buffer Configuration in the PARC-Transmitter

Name	Bus #	Starting Location	Ending Location	Buffer Size	Buffer Type
ADA1	3	3072 3198	3197 3323	126 126	Double Buffers, Short Floating Point
SAMPLE	3	0	1023	1024	Circular Buffer, Short Floating Point
VHAT	2	0	1023	1024	Circular Buffer, Short Floating Point
SHAT	3	1024	2047	1024	Circular Buffer, Short Floating Point
PARAMETER	2	3072	3231	20	Single Buffer, Long Floating Point
LEVEL	1	25000 27000	25599 27599	600 600	Double Buffer, Long Fixed Point
BIT	1	25700 27700	25899 27899	200 200	Double Buffers, Long Fixed Point

scaling factors, quantizer expansion factors, and the number of bits associated to each quantizer level.

The PARC-Transmitter employs four processors: The ADAM, the AP, the CSPU and the IOS-2. In order to achieve parallel processing, the double buffering technique is used. The function of these buffers associated to the processors at a time slot is shown in Fig. 12.7. At a time slot, the following processes are executed in parallel:

1. While the ADAM is filling a half of the ADAM buffer, the AP is accessing the other half.
2. While the AP is filling a half of LEVEL buffer, the CSPU is accessing the other half.
3. While the CSPU is filling a half of the BIT buffer, the IOS-2 is accessing the other half.

In the next two subsections, the pitch computation program and the speech digitization program will be described. The complicated buffer controller will also be presented.

12.2.1 The Pitch Computation Program

The pitch computation program contains the gain controller, a noise reducer, a buffer controller, the adaptive filter and the pitch extractor. The APU flow chart and the corresponding APS flow chart, including the controlling flows are depicted in Figs. 12.8 & 12.9. The detail functions are described as follows and the parallel processing is shown:

1. In response to the function list, the CSPU loads the APU and the APS modules of the pitch computation program into the APU and the APS respectively.
2. The CSPU then sets flag RI which will initiate the APS module.

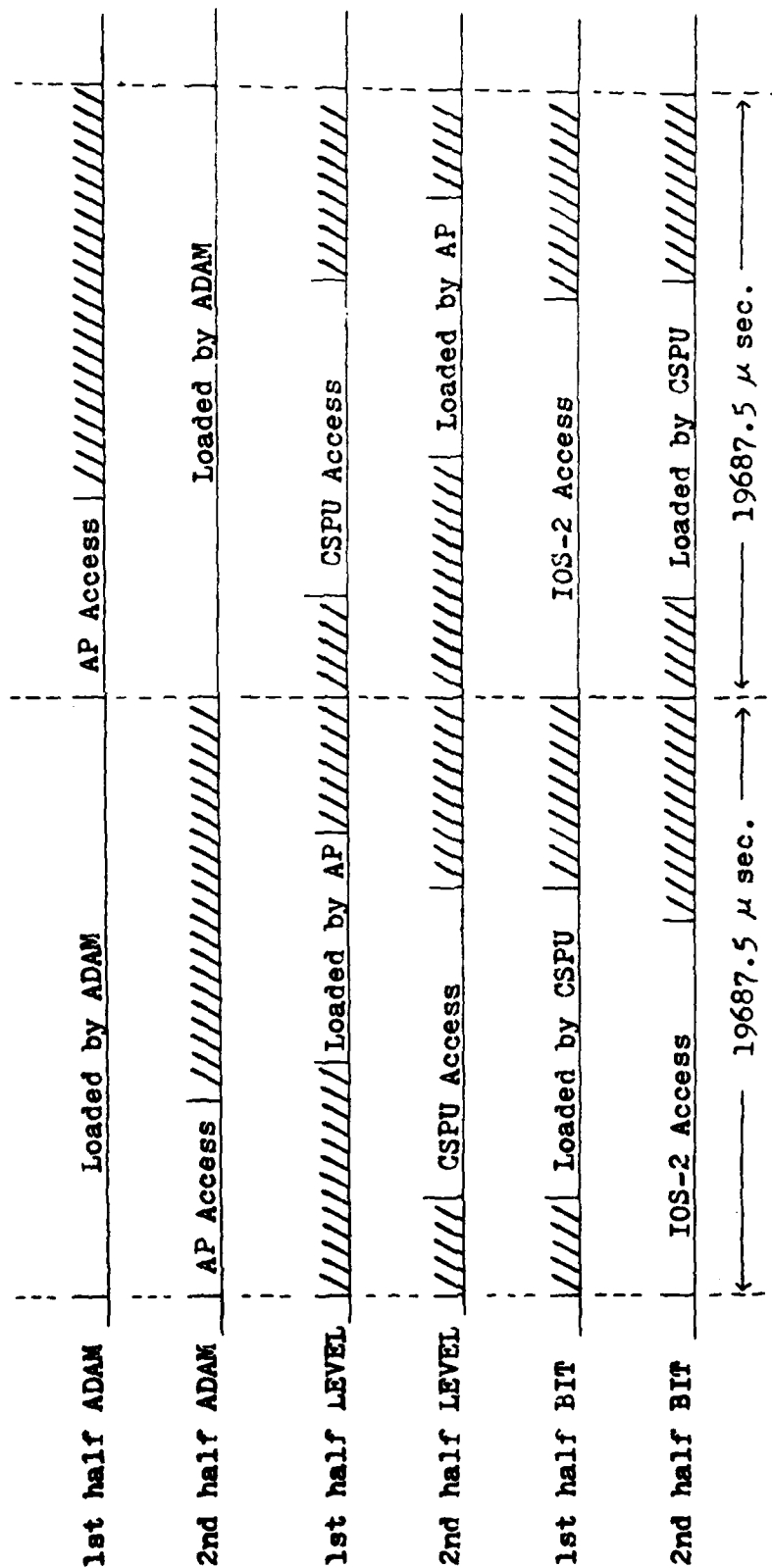


Fig. 12.7 The Double Buffering in the PARC-transmitter

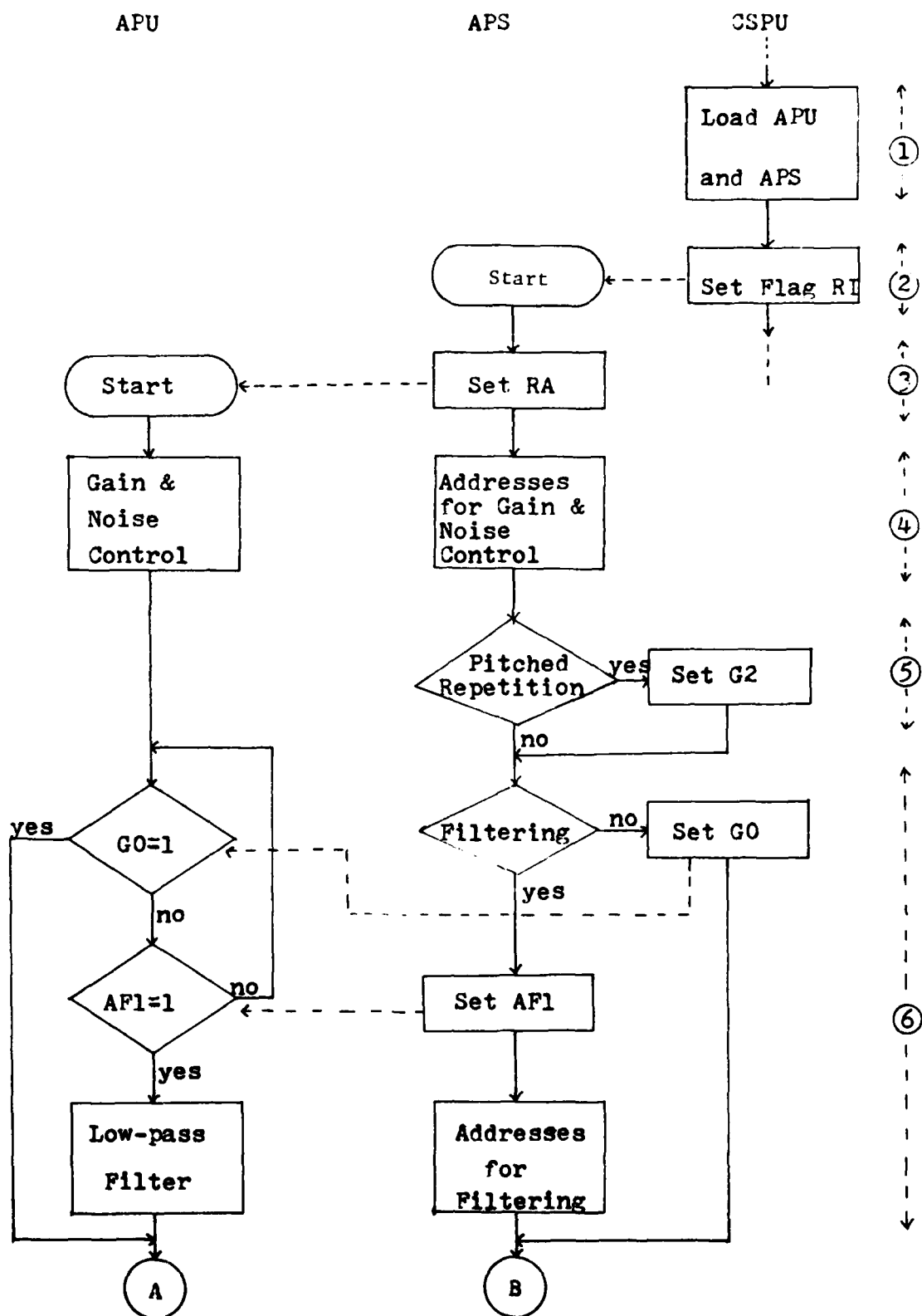


Fig. 12.8a The Flow Chart of the Pitch Computation Program

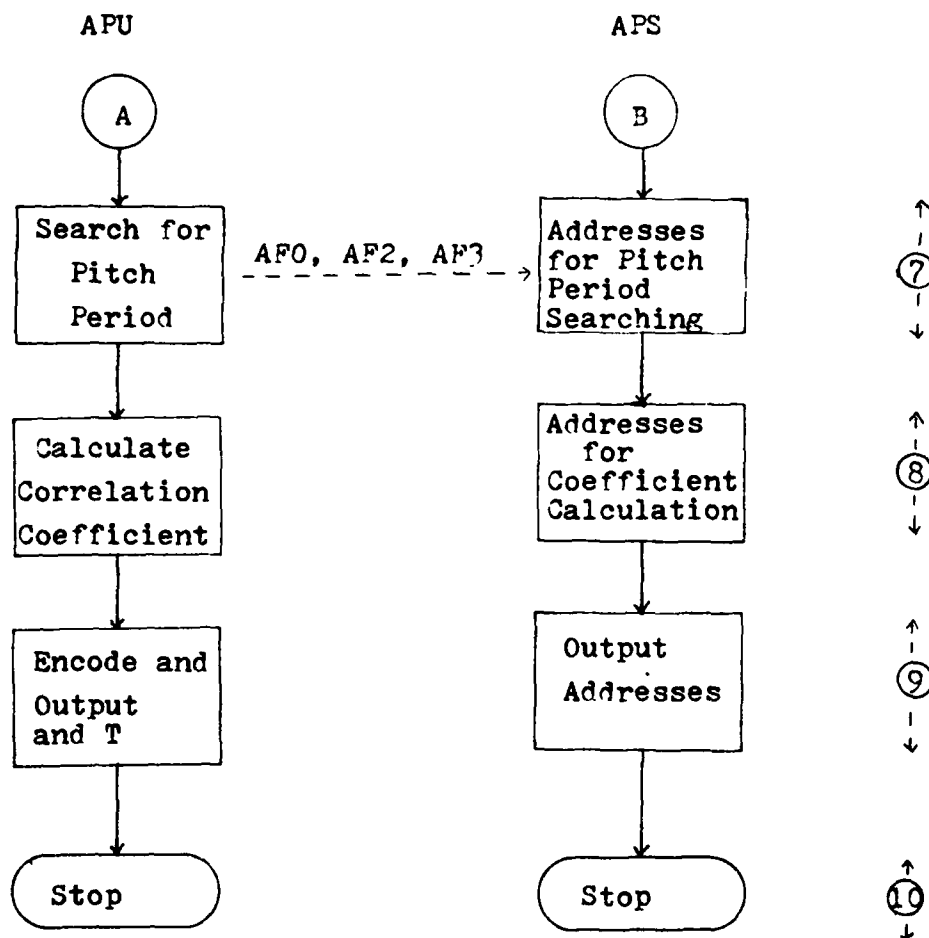


Fig. 12.8b The Flow Chart of the Pitch Computation Program

3. The APS module sets flag RA which will start the APU module.
4. The APS produces the addresses of the noise reducing parameter, the gain factor, 126 input data addresses from the ADAM buffer and 126 output data addresses to the SAMPLE buffer.

The APU reads 126 input samples whose addresses are given by the APS, reduces their noise, multiplies by the gain factor and outputs to the SAMPLE buffer. The input samples from the ADAM are 11-bit precision plus a sign bit in a range of $[-2047/2048, 2048/2048]$. After being read by the AP, these are converted into 32-bit floating point numbers.

5. The APS checks the fullness of the SAMPLE buffer, sets the flag G2 if it is over the pitch repetition threshold.
6. The APS again checks the fullness of the SAMPLE buffer. If it is under the filtering threshold, the APS sets the flag GO and jumps to Step 7. If it is over the threshold, the APS sets the flag AF1 and produces addresses for the filtering. The APU waits for the signals from the APS. If the flag GO is set, it goes to Step 7. Otherwise, it executes the filtering process which is described in Section 2.3.

7. The APS produces addresses for the data used to compute the pitch period. The APU computes the pitch period using the AMDF technique (refer to Section 2.4). In order to speed up the whole real-time system, all processors have to be utilized as asynchronously and as simultaneously as possible., i.e., the minimum use of the communication flags between the APU and the APS. However, because of the limit on the number of the registers in the APS, three flags are used in this portion to search for the pitch period as shown in Fig. 3.9. In this block, the APU determines how many iterations are left to be executed. The APS has to wait for the information, the flags AFO and AF1, at the end of each iteration.
8. The APS produces addresses for the data used to calculate the pitch correlation coefficient. If the flag G2, which indicates the pitched repetition, is set, it produces addresses starting a pitched repetition block behind. The APU calculates the pitch correlation coefficient.
9. The APS produces output addresses. The APU encodes the pitch period and the pitch correlation coefficient and outputs them to the MAP memory.
10. The AP interrupts the CSPU to indicate that it is finished.

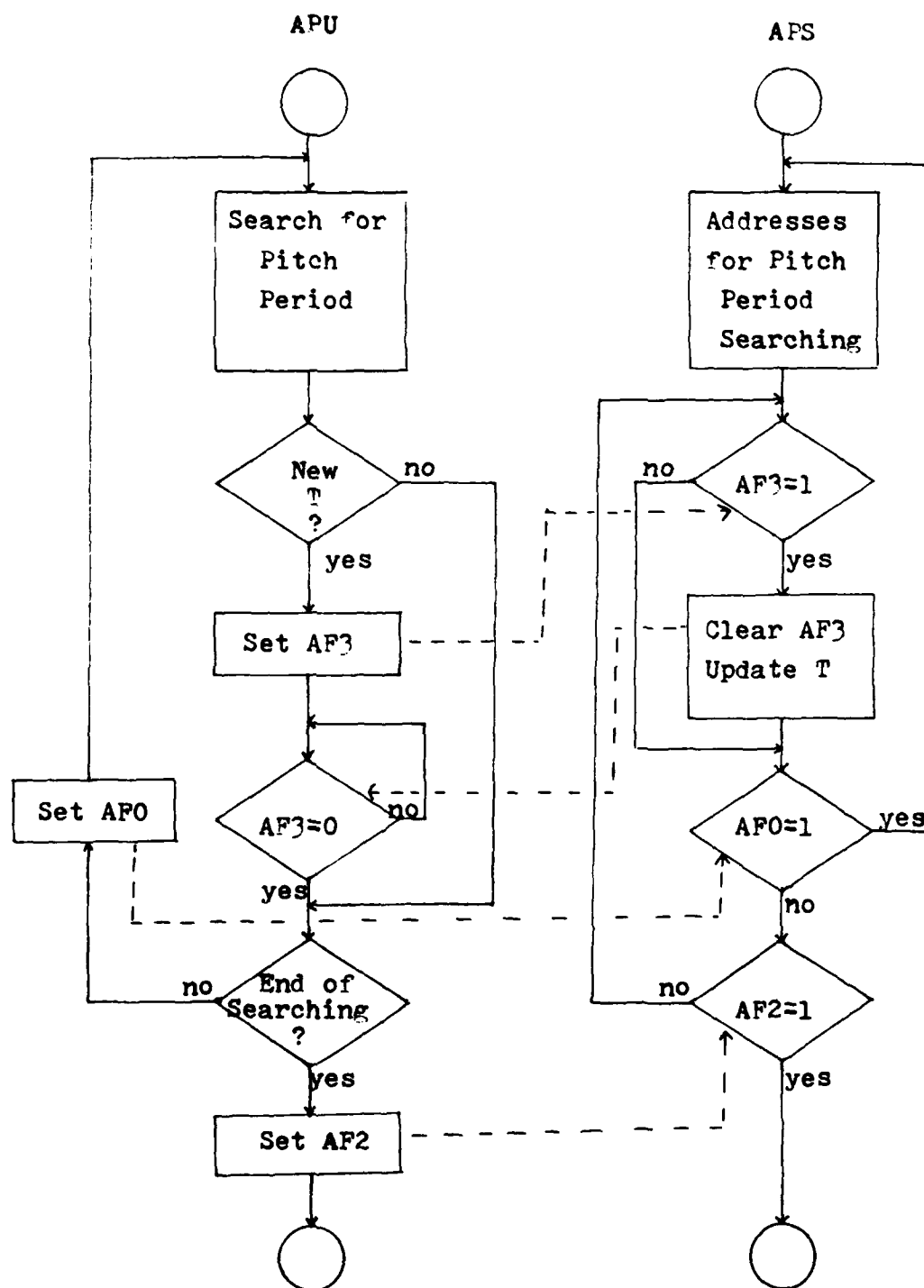


Fig. 12.9 The Flow Chart of the Pitch Period Searching Loop

12.2.2 The Speech Digitization Program

The speech digitization program contains the pitched repetition, the adaptive predictor, a maximum number of processed samples controller, a reciprocal function, the pitch extraction, the adaptive quantizer, the inverse adaptive quantizer, the bit buffer controller, the underflow controller, and a parameter update. The APU and the APS flow chart including the control flows are depicted in the Fig. 12.10 because the number of samples which are processed by the transmitter varies in each time slot, the flags APO, AF3 and G1 are used to communicate between the APU and the APS. The flag APO, which is controlled by the APU, is used to indicate the beginning of the digitization loop. The flag G1, which is controlled by the APU and the APS, is used to indicate an underflow of the sample biffer or reaching the maximum number of samples permitted in the time slot. The flag AF3, which is controlled by the APU, is used to indicate that at least 157 information bits have been generated.

The detail functions are as follows:

1. In response to the function list, the CSPU loads the APU and the APS modules of the speech digitization program into the APU and the APS respectively.
2. The CSPU then sets the flag RI which will initiate the APS module.
3. The APS module sets the flag RA which will start the APU module.
4. The APS produces the addresses of the system counters. The APU reads in the system counters. The system counters are the BIT counter, SAMPLE counter and the LEVEL counter. Because the fixed time slot is used as the main parameter for the

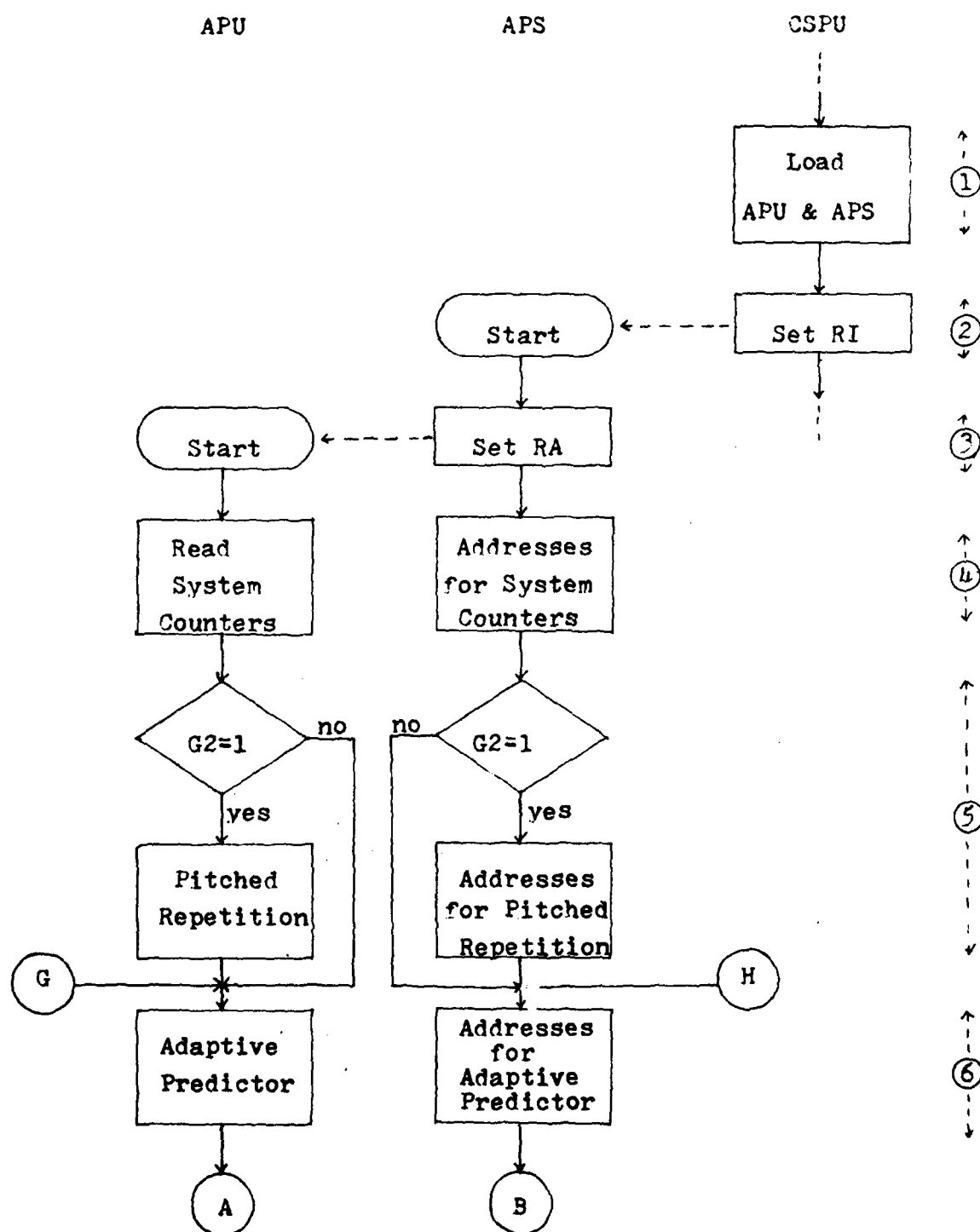


Fig. 12.10a The Flow Chart of the Speech Digitization Program

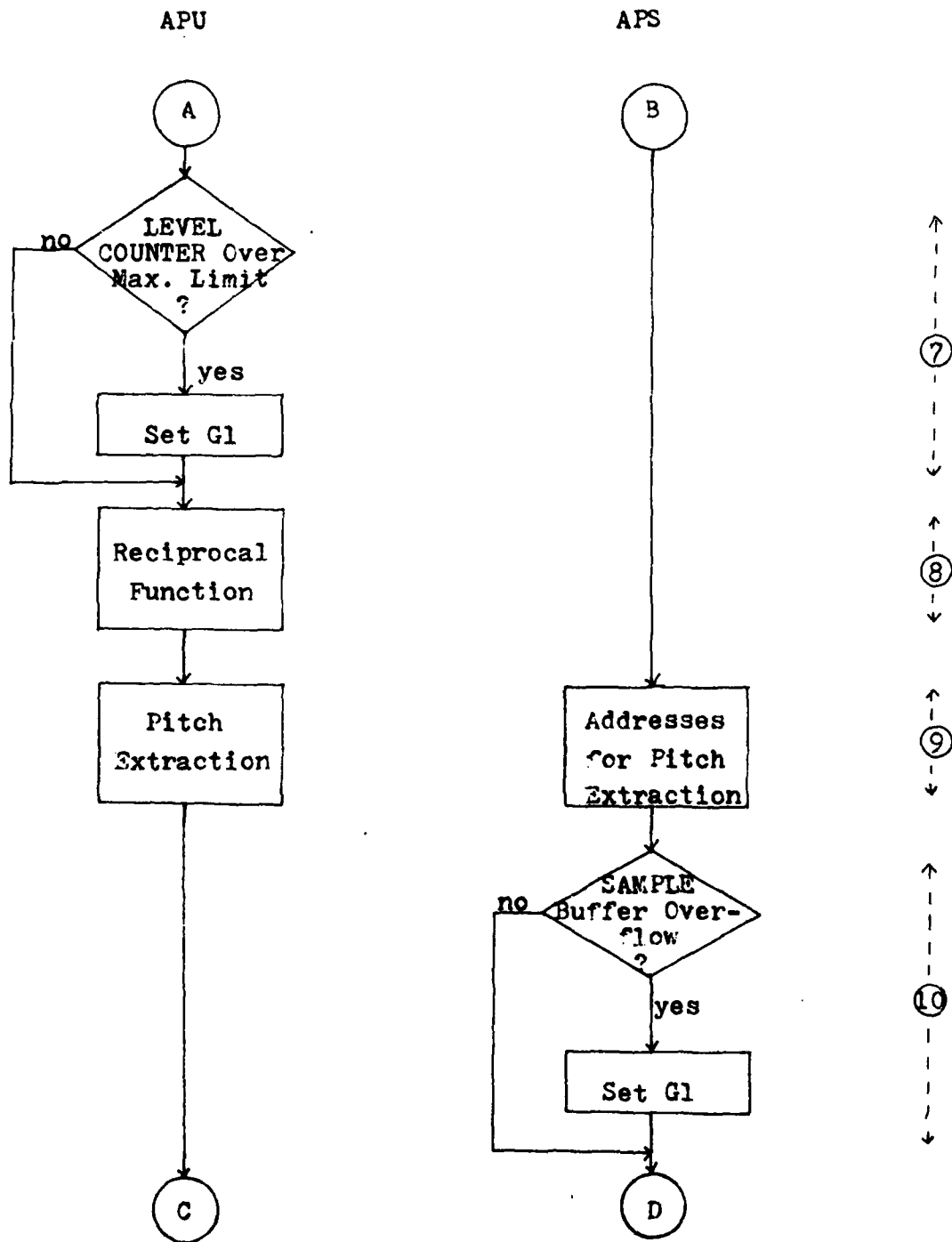


Fig. 12.10b The Flow Chart of the Speech Digitization Program

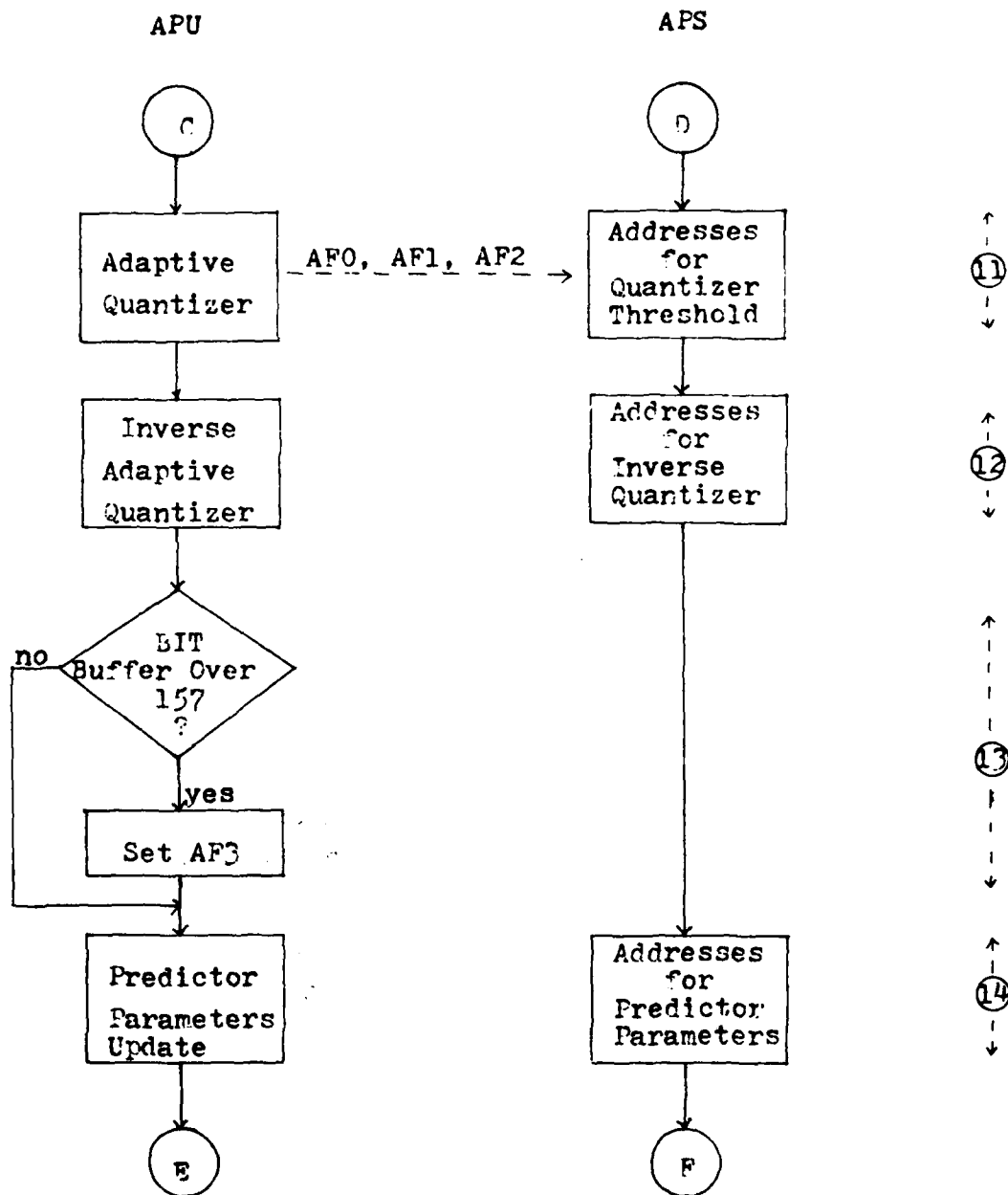


Fig. 12.10c The Flow Chart of the Speech Digitization Program

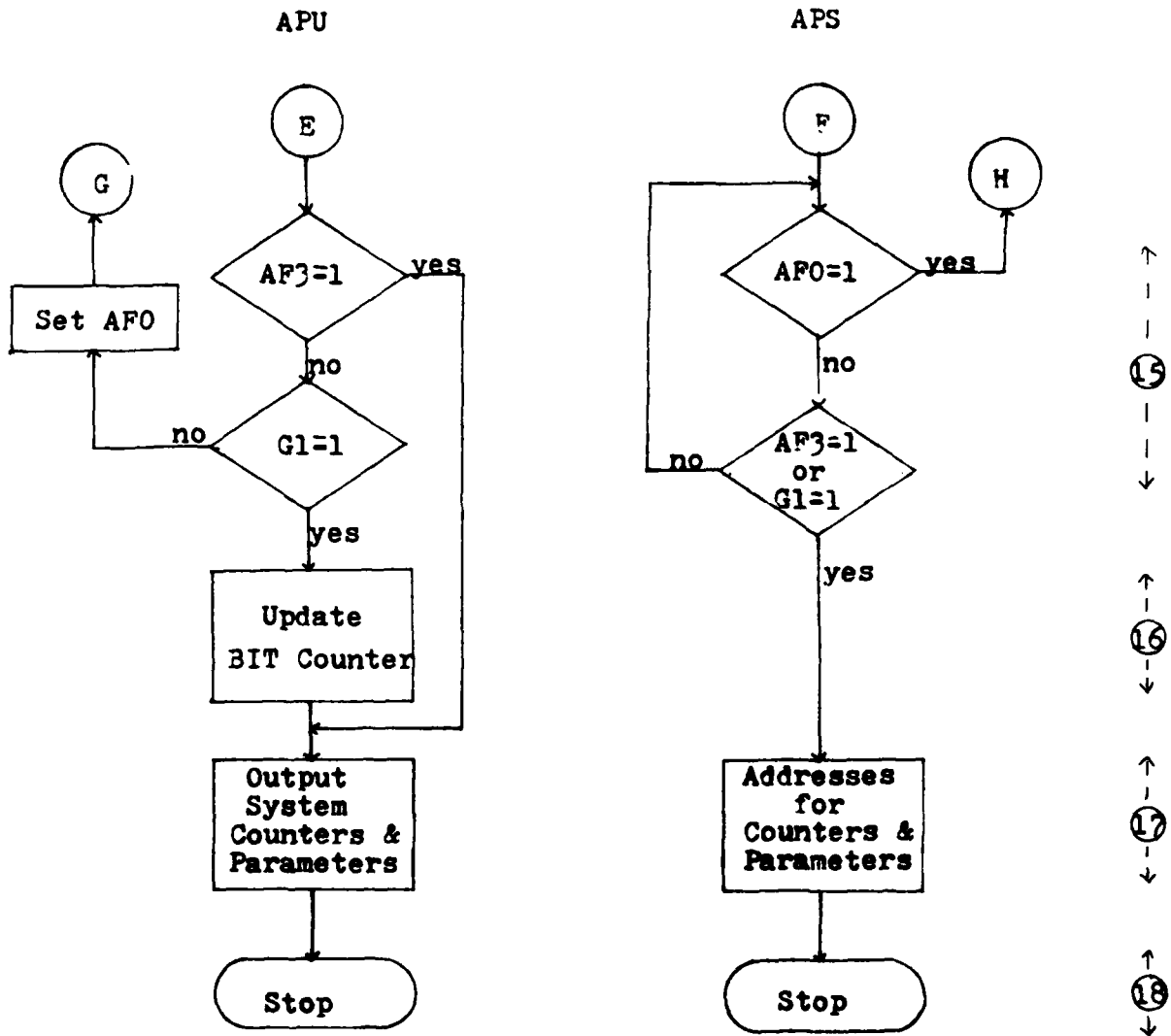


Fig. 12.10d The Flow Chart of the Speech Digitization Program

real-time system, the IOS-2 transfers 189 bits in a time slot. However, as described in Chapter 2, only 157 bits are available to encode the quantizer levels. the BIT counter is used to control the PARC-Transmitter so as to generate just more than 157 information bits. The SAMPLE counter is used to count the total number of samples processed by the PARC-Transmitter during a time slot. The address pointer update program will use this information to update the address pointers in the transmitter circular buffers. The LEVEL counter is used to limit the maximum number of samples which can be processed in a time slot. This assures real-time operation.

5. The APU and the APS check the flag G2 which is controlled by the pitch computation program to indicate the condition of the pitched repetition. If flag G2 is clear, there is no pitched repetition and the APU and the APS go to Step 6. Otherwise, the APS produces the addresses of SHAT buffer for the pitched repetition. The APU executes the pitched repetition. The size of the pitched repetition is a system parameter which can be input from the main Fortran program. The design of the real-time system assumes that the overflow of the sample buffer can be absolutely controlled by the pitched repetition. Thus, the size of the repetition has to be carefully considered.
6. The APS produces addresses for the predictor. The APU executes the predictor which employs a fourth order prediction.
7. The APU checks the LEVEL counter and sets the flag G1, if the counter is over the maximum number of samples allowed to be executed by the PARC-Transmitter.

8. The APU executes the reciprocal function to compute $1/(\text{RMS})^2$ and $1/\sigma$. To do the reciprocal of a 32-bit floating point number, the APU usually employs an 8 x 256 read only memory which has only 8-bit precision in the mantissa part. Thus, a special subalgorithm is used to increase the precision.
 9. The APS produces addresses for the pitch extraction. The APU executes the pitch extraction.
 10. The APS checks the SAMPLE buffer and sets the flat G1 if the buffer is empty.
 11. The APU and the APS quantize the input sample. Three flags are employed to communicate between the APU and the APS. The way to quantize an input sample is as follows:
 - A. The APU signals the APS to produce the threshold address.
 - B. The APU compares the magnitude of the input sample with the threshold.
 - C. If the threshold is larger than the magnitude, the APU signals the APS to stop the searching and jumps to the inverse quantizer portion. Otherwise, the APU checks whether the threshold is the largest one. If it is, the APU signals the APS and jumps to inverse quantizer portion. If it is not, control is returned to Step A.
- This design allows variable quantizer levels and reduces searching time, because most of the input samples are dropped in the first two levels.
12. The APU and the APS execute the inverse quantizer.
 13. The APU updates and checks the BIT counter and sets the flag AF3 if the 157-bit is reached.

14. The APU and the APS update the predictor parameters.
15. If the flag G1, which indicates an underflow of the SAMPLE buffer or reaching the maximum number of samples permitted in the time slots, is set, the APU goes to Step 16. If the flag AF3, which indicates the condition of BIT counter, is set, the APU goes to Step 17. Otherwise, the APU sets the flag AFO and goes to Step 6. The APS waits for the signals from the APU. If the flag AFO is set, it goes to Step 6. Otherwise, it goes to Step 17.
16. If this step is executed, there has been an underflow in the SAMPLE buffer, i.e., less than 157 bits are generated. The APU updates the BIT counter to account for a NULL code which indicates underflow. If the BIT counter is still less than 157 bits, the BIT counter is reset to synchronize the counters between the PARC-Transmitter and the encoder.
17. The APU and the APS output the contents of the system counters and parameters.
18. The AP interrupts the CSPU to indicate that it is finished.

12.3 The PARC Receiver

In this section, the PARC-Receiver, which resides in the AP, will be presented. The corresponding decoder and channel synchronizer will be explained in Section 12.5. The principal elements of the receiver, which is shown in Fig. 2.8, are also part of the transmitter. So, the implementation of receiver is much the same as that of the transmitter.

Before the description of the design of the PARC-Receiver, the buffer system has to be depicted. There are six buffers employed by the receiver as shown in Table 12.3.

Table 12.3 Buffer Configuration in the PARC-Receiver

Name	Bus#	Starting Location	Ending Location	Buffer Size	Buffer Type
AOM	2	3584	3835	252	Double Buffer, Short Floating Point
		3840	4091	252	
VHAT	3	2048	3071	1024	Circular Buffer, Short Floating Point
SHAT	2	2048	3071	1024	Circular Buffer, Short Floating Point
PARAMETER	2	4096	4255	80	Single Buffer, Long Floating Point
LEVEL	1	26000	26599	600	Double Buffer, Long Fixed Point
		28000	28599	600	
BIT	1	29000	30023	1024	Circular Buffer, Long Fixed Point

Among them, the AOM and the LEVEL buffers are double buffers which allow one processor to fill one-half of the buffer while another processor to process on the other half. This increases the parallel ability of the receiver. The VHAT, the SHAT and the BIT buffers are circular buffers which allow continuous access. However, the additional address pointers, which indicate the starting locations of these circular buffers at the beginning of each time slot, have to be considered. The structures of the LEVEL buffer and the BIT buffer will be explained in Section 12.5. The PARAMETER buffer is a single buffer which stores system parameters such as predictor coefficients, quantizer output scaling factors and quantizer expansion factors.

The flow chart of the receiver is shown in Fig. 12.11. The detailed functions are described as follows and the parallel processing is shown:

1. In response to the function list, the CSPU loads the APU and the APS modules of the PARC-Receiver program into the APU and the APS respectively.
2. The CSPU sets the flag RI which will initiate the APS module.
3. The APS then sets the flag RA which will start the APU module.
4. The APS produces 126 input addresses of the SHAT buffer and 252 output addresses of the AOM buffer. The APU processes the upsampling operation on 126 input reconstructed speech samples and produces 252 unsampled data points. The upsampling operation employs the linear interpolation technique which is explained in Section 2.8. The APU also limits the values of output data to be in the range of $[-2048/2048, 2047/2048]$ which is acceptable for the AOM.

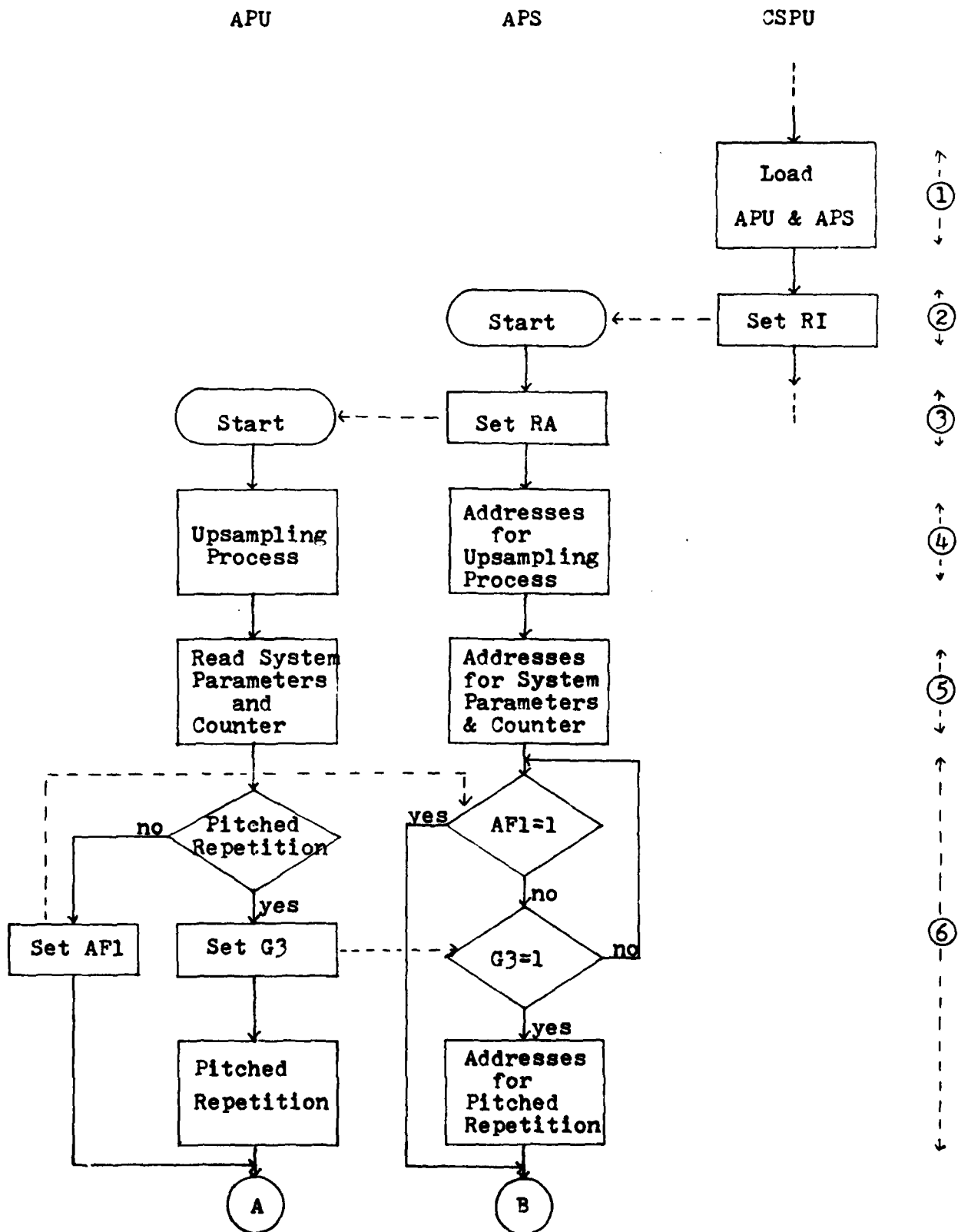


Fig. 12.11 The Flow Chart of the PARC-Receiver Program

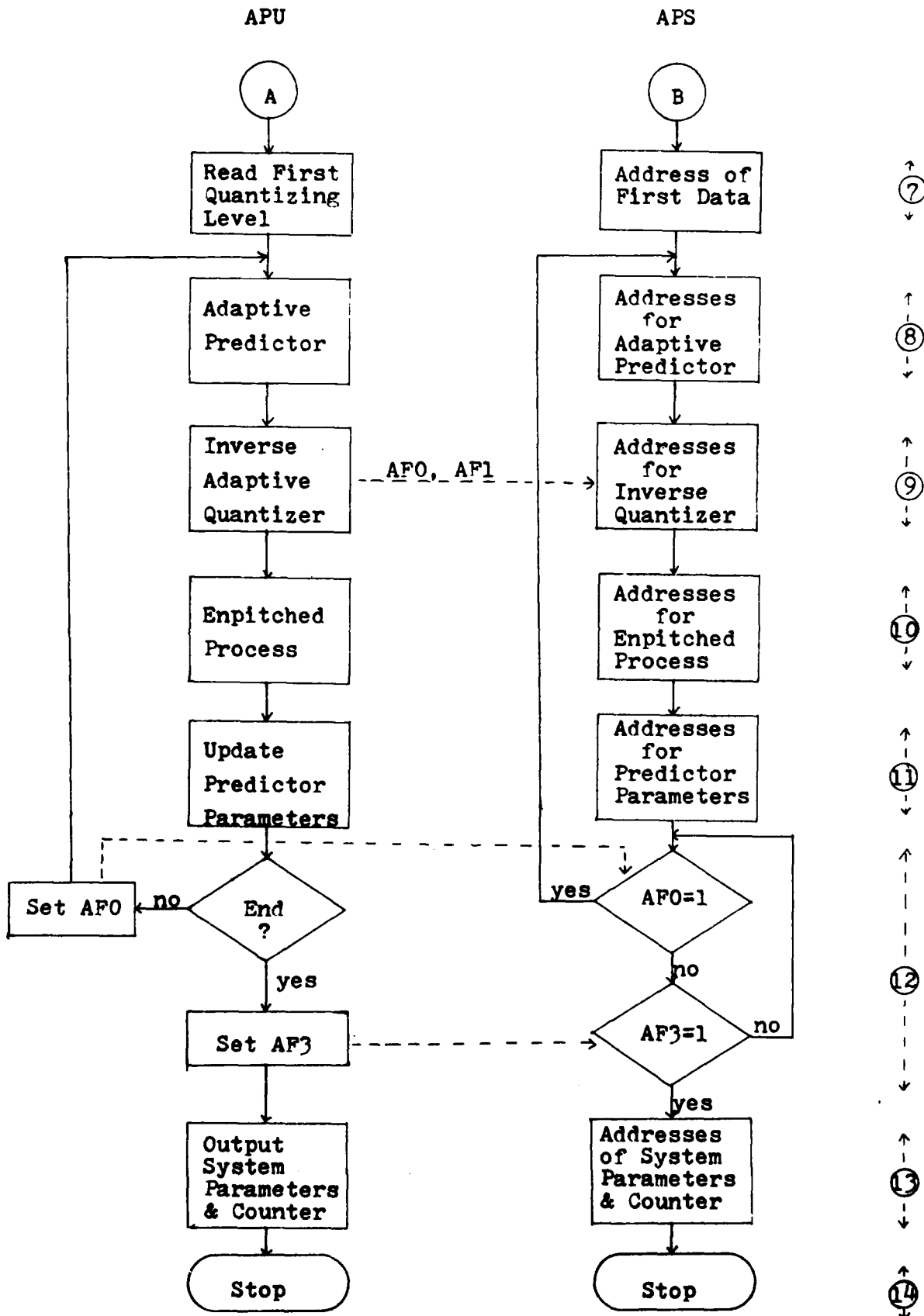


Fig. 12.11 The Flow Chart of the PARC-Receiver Program

5. The APS produces addresses of the system parameters and a counter. The APU reads the system parameters and the counter. There is only one system counter, the SAMPLE counter, which is used to count the total number of samples processed by the PARC-Receiver during a time slot. The receiver address pointer update program will use this information to update the address pointers of the receiver circular buffers.
6. The APU checks the first data in the LEVEL buffer and decides the condition of the pitched repetition. If no pitched repetition is employed, it sets the flag AF1 and goes to Step 7. Otherwise, it sets flag G3 and processes pitched repetition. The APS waits for signals from the APU. If the flag AF1 is set, it goes to Step 7. If the flag G3 is set, it produces addresses for the pitched repetition operation.
7. The APS produces the second data which is the first received quantizing level in the LEVEL buffer. The APU reads the first quantizing level.
8. The APS produces the addresses for the adaptive predictor. The APU processes the adaptive predictor which employs a fourth order prediction.
9. The AP operates the inverse adaptive quantizer. Two flags, AFO and AF1, are used to communicate between the APU and the APS.
10. The APS produces the address of $\hat{V}(K-T)$. The APU restores the pitch redundancy as shown in Eq. (2.1).
11. The APS produces addresses of the predictor parameters. The APU updates the predictor coefficients.

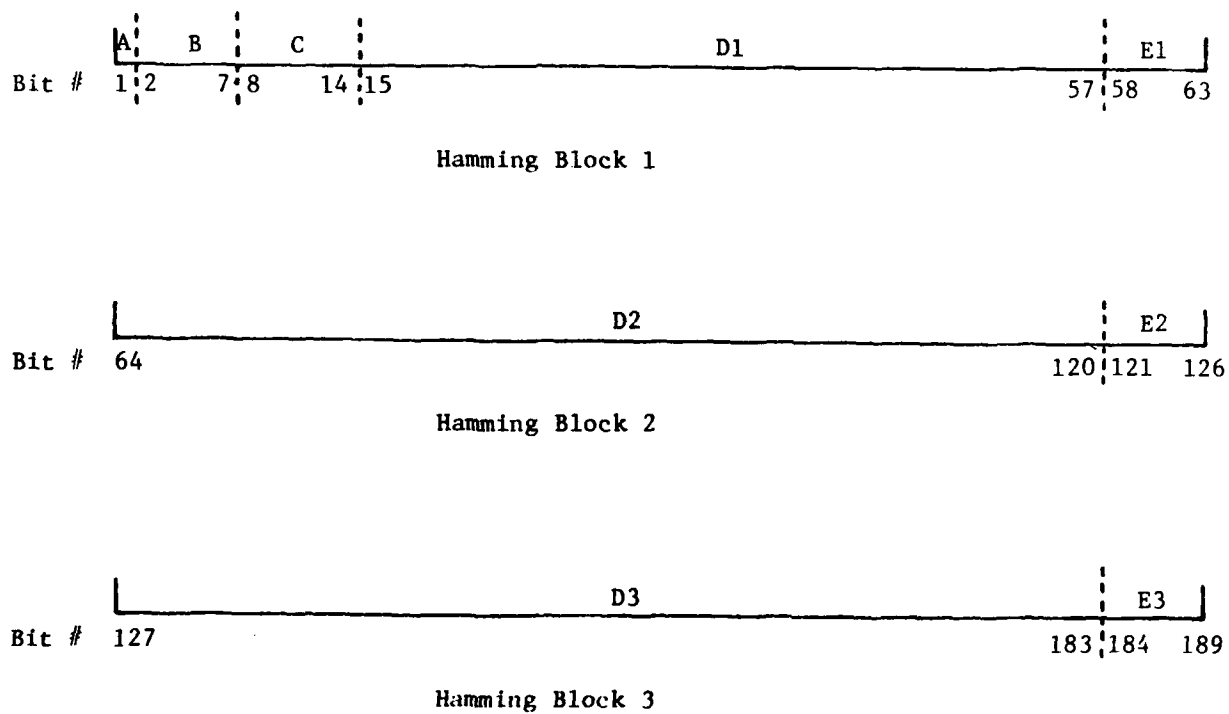
12. The APU reads the next data in the LEVEL buffer and checks whether it has reached the end of the buffer. If it has, the APU sets the flag AF3 and goes to Step 13. Otherwise, it sets the flag AF0 and goes to Step 8. The APS waits for signals from the APU. If the flag AF3 is set, it goes to Step 13. If the flag AF0 is set, it goes to Step 8 to continue the receiver loop.
13. The AP outputs the system parameters and the counter to the MAP memory.
14. The AP interrupts the CSPU to indicate that it is finished with the PARC-Receiver.

12.4 Noiseless Source Coder

This section describes the implementation of the noiseless source coder on the central processor of the MAP 300 called CSPU. As discussed in Section 2.7, the source coder performs several functions. Its primary purpose is encoding the quantizer levels and the side information for each block of N_B samples processed by the PARC transmitter. In addition, it supplies the bit pattern for frame synchronization and the parity code for channel error control.

At its input, the encoder shares a double buffer containing the pitched repetition indicator and N_B quantizer levels each with the PARC transmitter. The corresponding pitch reduction parameters β, T are in two sets of locations in the scalar table. At its output, it shares a double buffer containing 189 bits each with the IOS, the processor which interfaces with a digital modem.

The encoder strives to form a frame of 189 bits from the quantizer levels and the side information generated by the transmitter. The frame is subdivided into three Hamming blocks of 63 bits each. A Hamming block contains 57 information bits and 6 parity bits. Each of the 57 information bits in a block is protected against errors. Depending on whether or not pitched repetition is indicated, the output frame assumes one of two formats Figs. 12.12 and 12.13. The encoder proceeds sequentially to build the different fields in the output frame. It handles one Hamming block at a time. The parity codeword for the block is updated each time a bit is output. Immediately after the 57th information bit in a Hamming block is transferred, the parity codeword for that block is output. The codeword is then initialized for the next block, and the encoder continues to output further information bits. This



Field	Contents
A	Synchronization bit
B	Pitch period T
C	Pitch correlation coefficient β
D1, D2, D3	Quantizer levels
E1, E2, E3	Parity bits for error control

Fig. 12.12 Normal Format for Bit Frame Output of Encoder



Hamming Block 1

Field	Contents
A	Synchronization bit
B	Pitch period T
C'	False β , for pitched repetition
C	Pitch correlation coefficient β
D1	Quantizer levels
E1	Parity bits for error control

Fig. 12.13 Format of first Hamming Block during pitched repetition

process is repeated for three Hamming blocks, completing the output frame of 189 bits.

Fig. 12.14 shows the flow chart for the encoder. The program listing is contained in Appendix G. The encoder starts with field A, Fig. 12.12. It fetches the previous sync bit from a location where it was saved and inverts it. This new value is saved for use in the next frame. It is also output to field A. Next the encoded value of T is fetched and transferred to the output buffer, field B. The value of the pitched repetition indicator is then checked. If affirmative, the code 0000000 is transferred to field C', Fig. 12.13. Next the partially encoded value of β is fetched. It must be between 0 and 96 indicating one of 97 possible levels. Using this as an index, a 7-bit code is retrieved from the β encode table and output to field C, Figs. 12.12, 12.13. After this the quantizer levels are fetched one by one. The corresponding code-lengths and code-words are retrieved from the Q-level encode table and transferred to the output buffer until the fields D1, D2, and D3 are filled. This should coincide with the exhausting of the N_B Q-levels generated by the transmitter.

An exception to this occurs when the sample buffer runs out. The block of quantizer levels do not generate enough bits to fill up the fields D1, D2, and D3. In this case, a null code 11111110 is output to indicate the end of quantizer level information. If there is still more space available, it is padded in with 1's. The null code and the padding bits are error-protected the same as any other information bits.

The source code used here is a variable-length to variable-length mapping, and it is not necessary that fields D1, D2, and D3 are exactly filled by encoding an integer number of quantizer levels. The extra bits after a

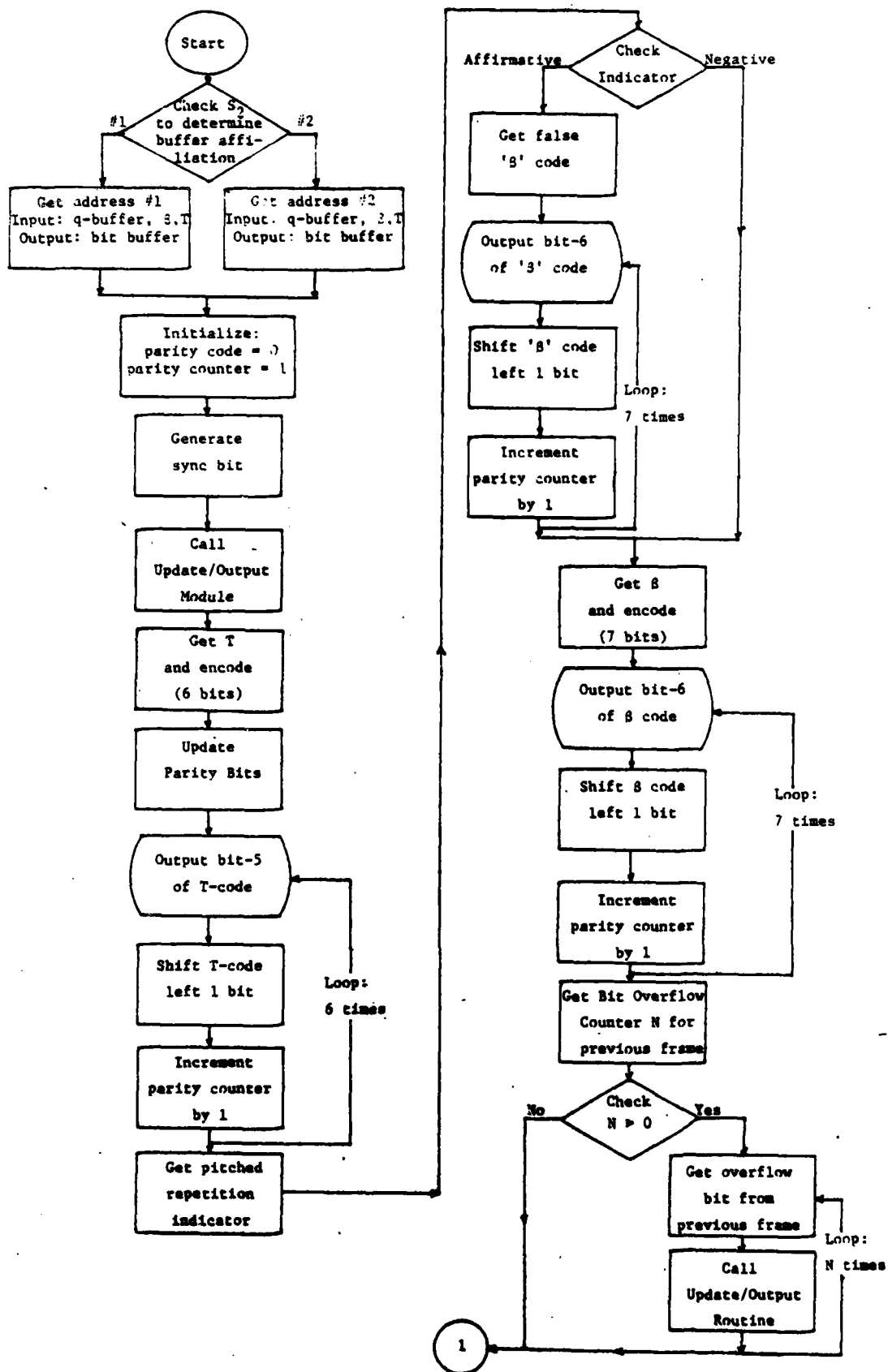


Fig. 12.14a Block Diagram for Encoder

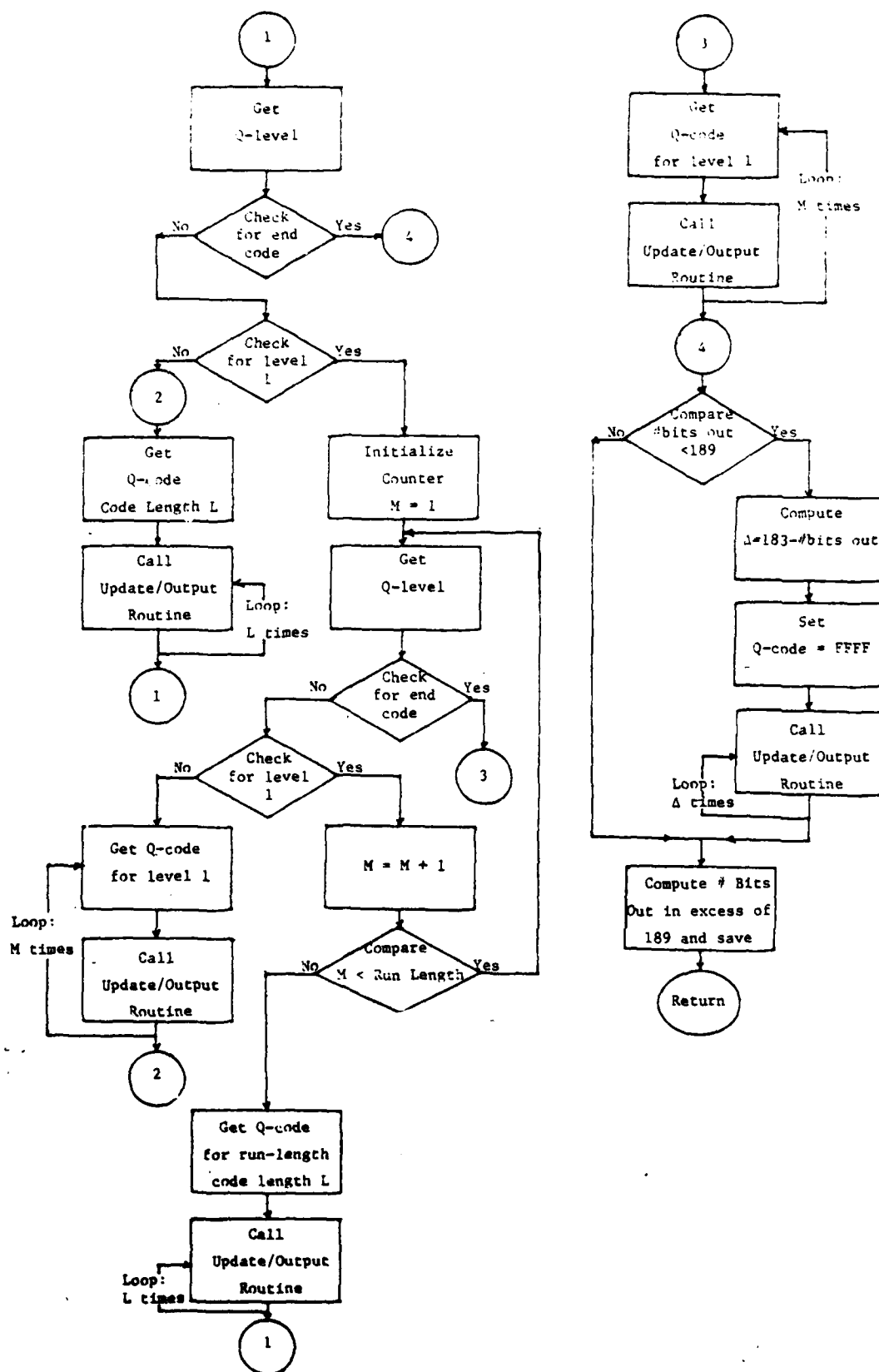
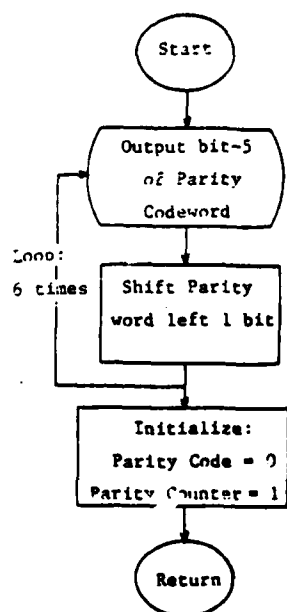


Fig. 12.14b Block Diagram for Encoder

A) Parity Output Routine



B) Update/Output Routine

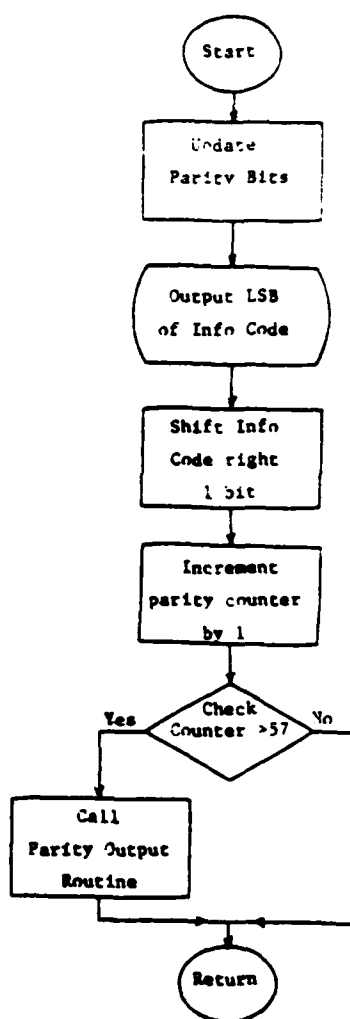


Fig. 12.14c Block Diagram for Encoder

Hamming block is full are passed to the appropriate field in the next Hamming block.

All the programs in each processor need to be able to process a block of information within a time frame of $19687.5 \mu \text{ sec}$ for the algorithm to keep in real time. This necessitated giving program efficiency priority over flexibility. The following is a list of the constraints on the encoder program as a result of this:

(i) All buffers used by the encoder must be located in Bus 1. They can be relocatable, as long as they do not occupy memory locations above 64K.

(ii) All the inputs to the encoder program must be in the fixed number format. The quantizer levels must be pre-multiplied by 2 to facilitate direct indexing. The values of β and T must be partially encoded. T must be limited to 64 values from 0 to 63, and β to 97 values from 0 to 96. Pitched repetition and the end of quantizer levels must be indicated by negative numbers.

(iii) The source code has the following constraints. The code for level 1 is 0. The locations for the run length code and the null code in the Q -level encoding table are 0 and 24, respectively. The other 11 codes for the quantizer levels Q are at locations $2Q$.

(iv) Because the LSB of a code-word is transmitted first, the code-word received at the decoder is inverted, e.g., 110 $\rightarrow$ 011. To compensate for this, the code-words stored in the Q -encoding table must be inverted and right justified.

(v) The Q -encoding table must contain $(L-1)$ for the various code lengths, L .

(vi) The code to indicate pitched repetition is 0000000. The 97

quantized values of β may not use any pattern with 2 or less 1's in it.

This is to give the pitched repetition indicator additional error protection.

(vii) The run length value can be selected at initialization.

(viii) Finally, the block of N_B quantizer levels should not generate more than 196 bits. Since the encoder does not monitor the maximum number of bits generated, the transmitter must ensure this limit.

12.5 Synchronizer, Decoder

This section describes the implementation of the decoder on the central processor of the MAP 300 called CSPU. The decoder essentially inverts at the receiver the operation performed by the encoder at the transmitter. From each received frame of 189 bits, it generates a block of N_B quantizer levels, the pitched repetition indicator, and the pitch extraction parameters for use by the PARC receiver. In addition, it performs several other operations. It monitors transmission errors, and corrects up to one error per Hamming block in the received frames. Because the received bit stream is arranged in frames, the decoder acquires the initial frame synchronization using the synchronization bit pattern transmitted by the encoder. Later, it monitors the received sync pattern to ascertain that synchronization is not lost. It monitors the state of the sample buffer in the PARC receiver to make sure the buffer does not overflow or underflow. Finally, it controls the mode of operation of the entire PARC algorithm.

The different operations outlined here are performed in different modules in the decoder. The decoder program is subdivided into three modules: the initialization module; the synchronization acquisition module; and the decoder module. The function to be performed and correspondingly the module to be used is determined by the three values of a function select switch, S1. The first time the program goes in for sync acquisition, several parameters and buffers need to be initialized. This mode is indicated by a value of 0 for S1. Subsequent sync acquisition operations are indicated by a value of -1 for S1. After frame sync has been established, the decoder can perform its task of inverting the bit stream to quantizer levels. This mode is indicated by a value of 1 for S1.

The decoder shares a double buffer with the PARC receiver at its output. A second function select switch, S2, indicates which half of the double buffer the decoder must use in the current time frame.

At its input, the decoder shares a circular buffer 1024 bits long with the IOS. The decoder expects to receive 189 bits in each time frame. However the clock that controls the inflow of the bit stream is at the other end of the communication system. It is different from the clock that measures the time frame at the decoder. On the average, the decoder does receive 189 bits per time frame. However, because of the different clocks, there could be temporary deviations. The large circular buffer which is several times the frame size allows for these deviations without causing a conflict between the decoder and the IOS over the bits being read out of and written into the buffer. The IOS, which receives the bit stream from the digital channel, has its base pointer which indicates the position of the base of the current frame to be entered into the buffer. When the PARC algorithm is initiated, the base pointer of the decoder which indicates the base of the frame of bits it should process in the current time frame is set several blocks behind the input pointer controlled by the IOS. Each processor updates its pointer independently at the end of the time frames. As long as the input and output rates match on the average, no conflict should occur and the two pointers should remain reasonably separated.

On being called, the decoder checks S1 and transfers control to the appropriate module. The module performs its function, and then it sets up the control and updates the relevant parameters and information for the operation in the next time frame. The program listing of the decoder is included in Appendix G. The following subsections describe in detail the operation of the three modules that make up the decoder.

12.5.1 Synchronization Acquisition Module

Before the decoder can start inverting the received bit stream, the boundaries of the frames, marked by the sync bit, must be located. This task is achieved by the synchronization module, over several time frames. The sync acquisition algorithm is described in Chapter 3. The implementation here varies slightly from the description in Chapter 3. This is to reduce the amount of computation and memory storage required and to make the program more time efficient. The block diagram of the sync acquisition module is shown in Fig. 12.15.

The encoder inserts an oscillating bit S_i in field A, Fig. 12.12, for synchronization. The decoder generates a similar pattern of expected sync bit values, $\hat{S}_i$. During sync acquisition, the correlation of $\hat{S}_i$ and each of the 189 bit positions starting from the base pointer is computed. A cumulative correlation tally from one time frame to the next for each of the bit positions is stored separately. Just in case the sync sequence being generated at the decoder is the inverse of the sync pattern being received from the transmitter, a cumulative tally of correlations with $\bar{\hat{S}}_i$ is also saved for each of the bit positions. The bit position corresponding to the maximum cumulative tally in each time frame is kept. When the correlation tally for a particular bit is maximum for ten consecutive frames, it is assumed the sync bit has been located. That position, marking the beginnings of frames generated by the encoder, is saved.

If at the end of a sync acquisition operation, it is determined that sync has not yet been acquired, the program sets up for one more sync operation in the next time frame. The base pointer in the circular buffer is updated by 189. The current value of $\hat{S}_i$ is inverted and saved. And the function select switch S2 is changed to indicate opposite buffer affiliations

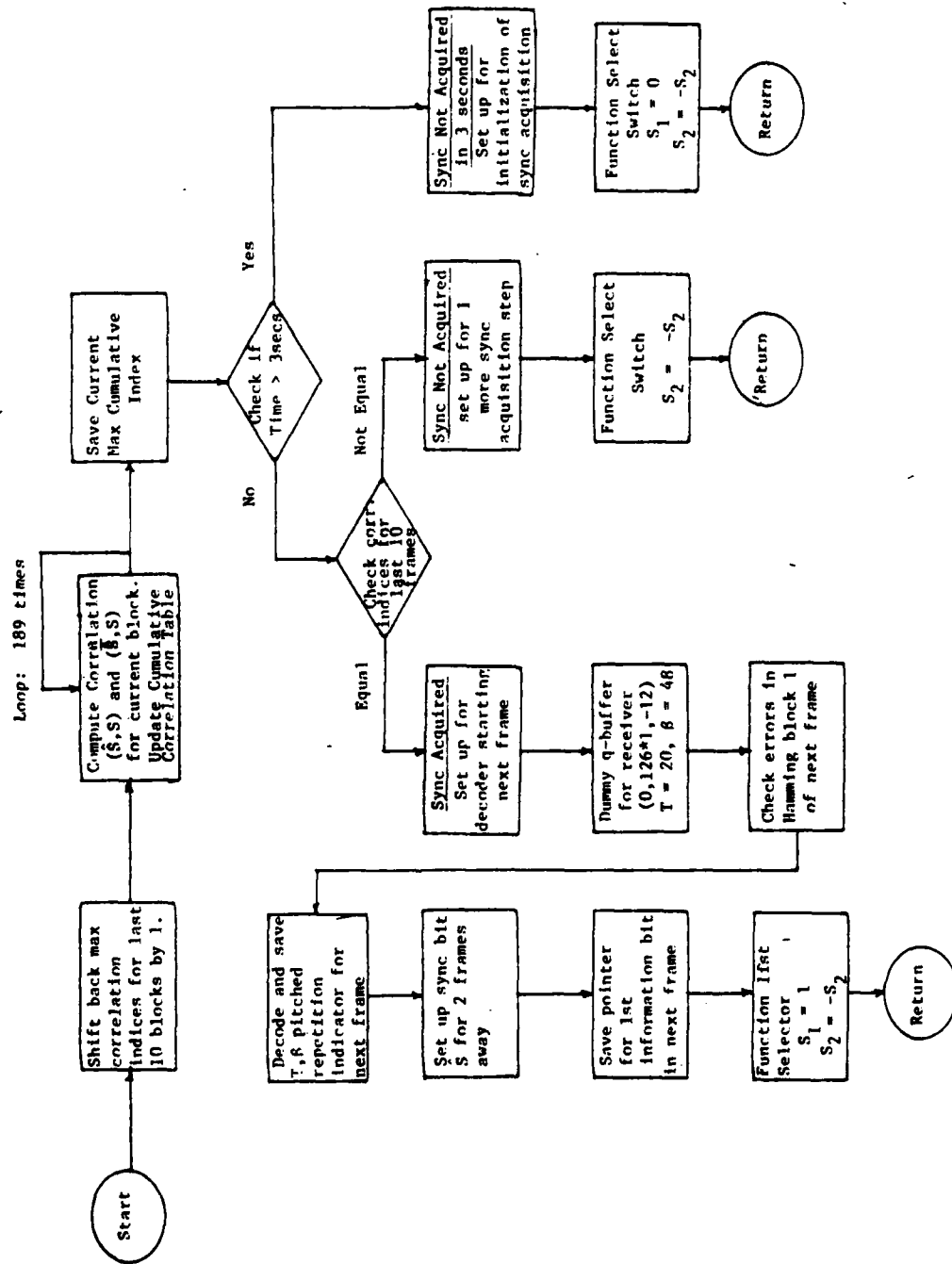


Fig. 12.15 Block Diagram for Sync Acquisition

for the next time frame.

If synchronization is achieved, preparations are made for subsequent decoder operations. In addition to the above controls and parameters, several others are updated. Function select switch S1 is changed to 1 to indicate a decoder operation in the next frame. Although no decoder operation has so far been performed, a dummy buffer consisting of 126 level 1's is prepared for use by the PARC receiver. Error check is conducted on the first Hamming block of the next bit frame; and up to one error is corrected. T, β and the pitched repetition indicator for the next frame are decoded and saved for output later. This leads to the location of the first bit in the field B1, Fig. 3.12, of the next frame. This is called the 1st information bit, information about quantizer levels. Its location is saved for use in the next frame. The cost function for sync monitor is initialized to -1. And the expected sync value $\hat{S}_{i+1}$ is set to match the next received sync value S_{i+1} . Finally, the state of the receiver sample buffer is initialized to a buffer-full condition. The control is then returned to the MAP executive.

12.5.2 Decoder Module

This module is the heart of the decoder. It corrects for transmission errors, if possible; monitors frame synchronization; inverts the received bit sequence to information usable by the PARC receiver; and performs buffer control on the receiver sample buffer. Its block diagram is shown in Fig.12.16.

After it decides its buffer affiliation by checking S2, it does error detection and correction on the 2nd and 3rd Hamming blocks of the current bit frame and the 1st Hamming block of the next bit frame. The reason for this offset between the bit frame and the error control frame is to make the extra bits generated during the encoding of the current frame available to the decoder when it performs the decoding. The extra bits reside at the

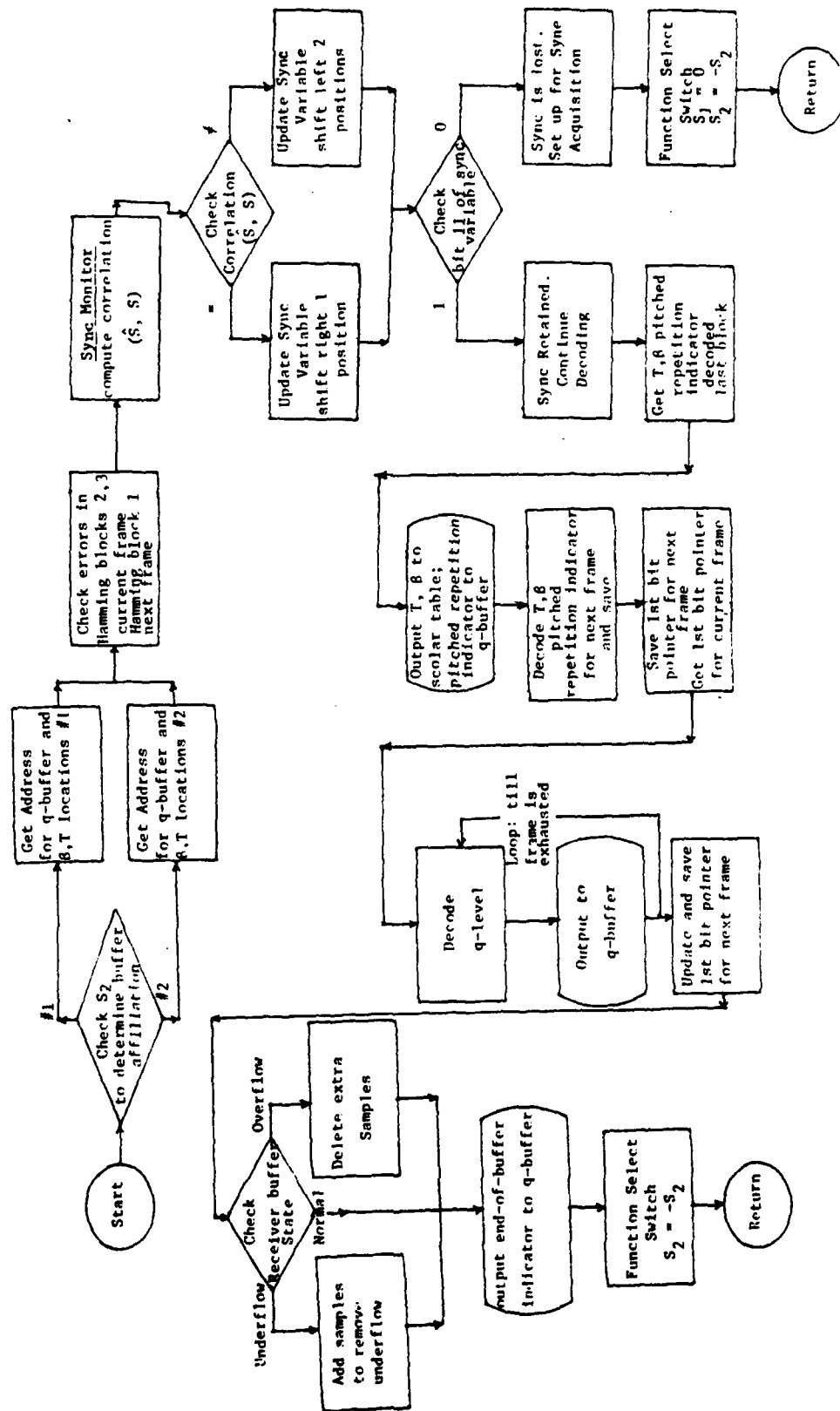


Fig. 12.16 Block Diagram for Decoder

beginning of field B1, Fig. 12.12 of the next frame.

The next step is sync monitor. The basis of the algorithm is explained in Chapter 3. The sync variable starts with an initial value of -1, FFFF in hexadecimal representation. The received sync bit S_i is compared with its expected value $\hat{S}_i$ using the logical operation EXOR. If equal, the bits in the sync variable v_i are shifted right one position. Bit 15, the sign bit is retained 1 in this operation. If not equal, the sync variable is shifted left two positions, causing 0's to be inserted in the least significant bits of v_i .

Bit 11 of v_i is monitored. If it is 0, sync is declared lost. The module sets the switch S1 to 0 to indicate a series of sync acquisition operations starting in the next time frame, and returns control to the MAP executive.

As long as bit 11 of v_i is 1, sync is declared preserved, and the module continues to its next phase of operation, source decoding. The pitch reduction parameters β and T which were decoded and saved in the previous time frame are output to their appropriate locations in the scaler table. The pitched repetition indicator, also decoded in the previous time frame, is transferred to the output buffer. If affirmative, the receiver buffer pointer is updated by the length of the repetition block.

After this, the pitch reduction parameters and the pitched repetition indicator for the next time frame are decoded and saved. This leads to the position of the 1st information bit in the next frame. If there were any extra bits in the current frame, the decoder will know where to look for them when it needs them.

The source decoding is a tree searching operation along the coding tree. Each successive bit received represents a node in the tree. Starting from the root of the tree, bits are read in, until the last bit received represents the termination of the tree. The corresponding quantizer level(s) are determined

from the decoding table and transferred to the output buffer. This process continues until either a null code is received or all the bits in the current frame are exhausted. At the end of the the decoding operation, the location of the 1st information bit for the next frame is saved.

The decoder can correct transmission errors to a point, after which errors filter through to the received bit stream. Errors have two effects on the decoding procedure. The quantizer levels generated are wrong. And the number of quantizer levels N_B in the current block can be different from what it should have been. This causes the state of the sample buffers in the transmitter and receiver to lose correspondence. To correct for this, the decoder provides a steady drift in the state of the receiver buffer. When N_B for a received block is 126, it indicates that the transmitter was processing silence when it generated this block and its sample buffer was empty. Correspondingly, the receiver buffer should be full. A few extra level 1's are added to the q-buffer generated for the receiver; so that if errors have caused the receiver to move away from its buffer-full condition, it should slowly drift back. This alleviates the mismatch between the buffer states at the receiver and the transmitter.

Finally, the state of the receiver sample buffer is checked. If it underflows, extra level 1's are added to remedy it's negative state. If it overflows, the extra samples are deleted.

After this, the module updates various controls and parameters for another decoder operation in the next time frame. It then returns control to the MAP executive.

12.5.3 Initialization Module

The initialization module precedes the first of a sequence of sync

acquisition operations. Sync acquisition requires two buffers initialized for its proper operation. This is done by the initialization module. The locations which contain the bit positions with max correlation tallies from the last ten frames are initialized to -1. The buffer where the correlation tallies are stored is initialized to 0. The switch S1 is set to -1 to indicate subsequent sync acquisition operations.

In addition, the length of the pitched repetition block is retrieved from the function control block and saved for later use in the decoder. It also gets the base address of the input buffer. It then transfers control to the sync acquisition module.

12.5.4 Decoder Constraints

As with the other modules in the PARC algorithm, the stress in developing the decoder program is execution efficiency rather than flexibility. This is to try and meet the rather tight limitations of real time operation. The following constraints are imposed on the decoder:

(i) All buffers used by the decoder must reside on Bus 1. They can be relocatable, as long as they do not occupy memory locations above 64K.

(ii) The input for this program is a circular buffer of length 1024. The information bit must be contained in the LSB of each word.

(iii) All the outputs of the decoder are in the fixed point number format. The β and T are partially encoded and are represented by 7 and 6 bits respectively. β is allowed one of 97 codewords, 0 to 96, and T one of 64, 0 to 63. Pitched repetition and end-of-quantizer-levels are indicated by negative numbers.

(iv) The run length value can be selected at initialization.

12.6 Program Timing and Speed

The programs in the real time implementation of PARC get executed in parallel on the different processors of the MAP 300. The set of programs on a given processor must be executed once in each time frame. The number of samples processed by these programs is not constant. It varies from about 50 during the voiced regions of speech to about 520 during the transition regions following voiced speech. For the algorithm to operate at its peak performance, the programs should be able to process the maximum number of 520 samples within a time frame. This is, however, limited by processor speeds and program efficiency.

The average number of samples that need to be processed by any program is 126; and it is imperative that the programs be able to process at least a few extra samples if necessary. Currently, the programs do satisfy this minimum requirement. The pitched reduction program, PARC transmitter, and PARC receiver which operate on the AP can process about 150 samples per time frame. The encoder and decoder programs which operate on the CSPU can process about 200 samples per time frame. So currently, the limit on the number of samples processed is set at 150. This exceeds the minimum requirement only marginally, and speeding up the programs would improve the performance of the algorithm considerably. Some ways to achieve this improvement are suggested below, and it is expected that different programs will execute 20% to 50% faster as a result of these modifications.

The pitched reduction program which currently requires about 5 msec can be speeded up 100% by doubling the parallelism in its computation. The transmitter and receiver programs need to be reorganized in terms of their register usage and operation sequencing. This should provide approximately 30% speed

improvement. These changes should increase the number of samples processed on the AP by about 50%.

The encoder can be speeded up 25% by rearranging the encoding table and changing all memory access from indirect to direct addressing. This would require the input quantizer levels to be $2(q-1)$ instead of $2q$ as they are now. The decoder can be improved slightly by rearranging its decoding table and operation sequencing.

Two of the three buses on the MAP 300 have slower MOS memories. Changing these to the fast bipolar memory would also help the speed on the various processors because of the large number of memory transfers performed in each time frame. With these modifications, it is expected the algorithm would be able to process 250-300 samples per time frame instead of the 150 it does now. Although this still allows the algorithm to operate well below its peak level of fidelity, this is about the level of performance that can be expected given the current speed of the MAP 300 array processor.

APPENDIX E

RESAMPLING PROGRAM

During the course of this project, because of the different sampling rates at which digitized speech was available from various sources and the different rates at which it was utilized, it was found necessary to develop an efficient resampling program to enable us to easily change sampling rates of speech. The following briefly describes the resampling algorithm and the operational details of the program. Fig. E.1 shows the flow chart, and program listings are included at the end of this appendix.

Sampled speech is the pulse amplitude representation of an analog speech signal at a sequence of time points separated by T_1 . At another sampling rate, it is the PAM representation of the same signal at a different set of points separated by T_2 . The new set can be obtained by interpolation from the first set of samples. So as also to limit the bandwidth of signals, interpolation was performed by discrete convolution of the original samples with the truncated impulse response of an ideal low pass filter. A first order Hamming window was used for truncation to reduce the effects of truncation. The impulse response and the Hamming window have the following form:

$$H_F(t-\tau) = 2F \operatorname{sinc}(2F \pi \tau) \quad -\infty < \tau < \infty$$

and

$$W_T(t-\tau) = \alpha + (1-\alpha) \cos(\pi \tau/2T) \quad -T < \tau < T$$

$$\alpha = 0.7$$

The convolution function is the product of these two. The discrete version of the convolution function is scaled down by T_1 , to compensate for the scaling introduced by discrete filtering.

$$C_1(nT_2 - (m-1)T_1) = \frac{2F}{T_1} \{ \alpha + (1-\alpha) \cos [((mT_1 - nT_2) + iT_1)/NT_r] \}$$

$$\{ \operatorname{sinc}[2F\pi((mT_1 - nT_2) + iT_1)] \}, \quad \frac{-NT_r}{T_1} < i < \frac{NT_r}{T_1}$$

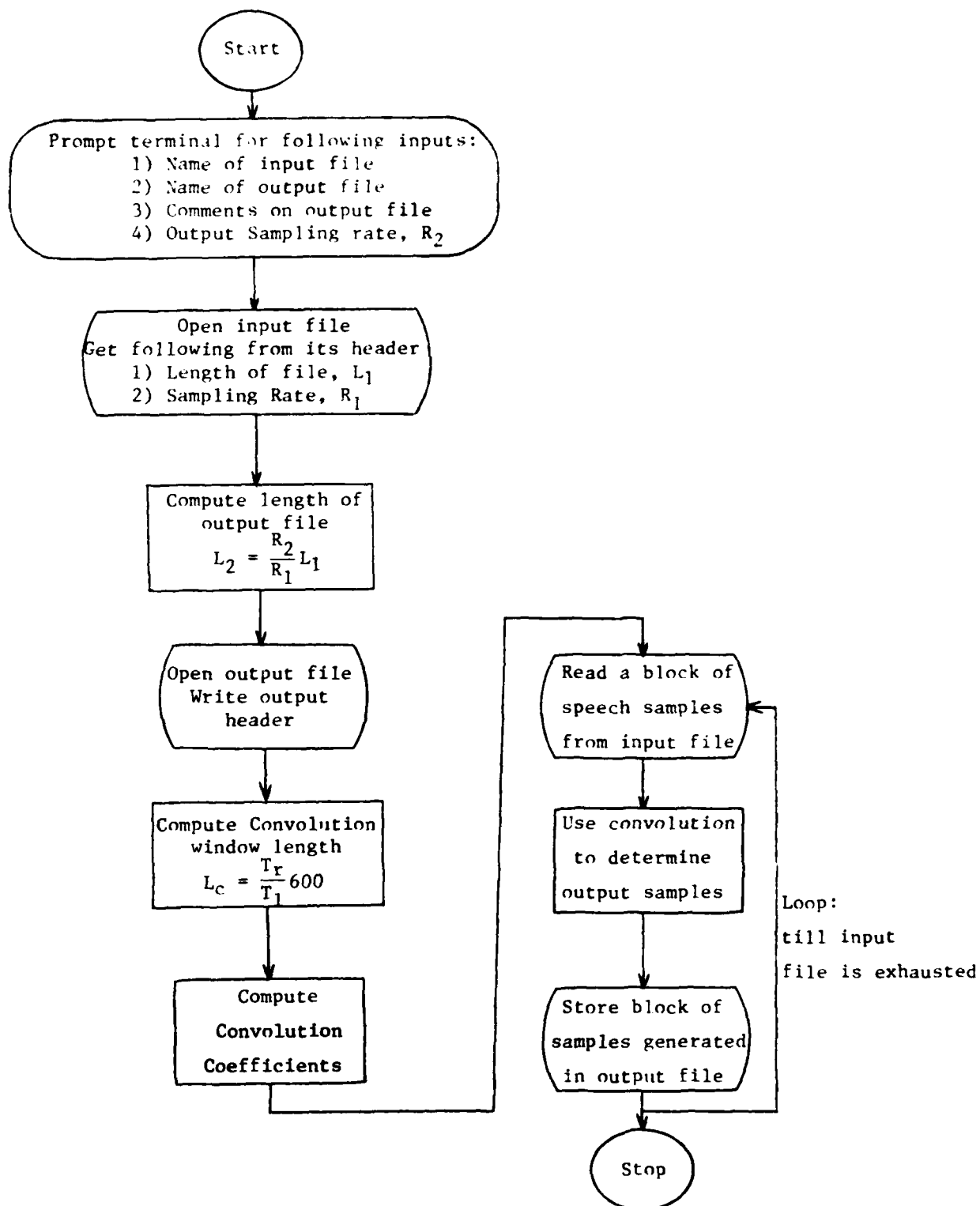


Fig. E.1 Flow Chart for the Resample Program

Here nT_2 identifies the n th output sample and mT_1 the closest preceding input sample.

Because the phase $(mT_1 - nT_2)$ changes arbitrarily, the correlation coefficients need to be computed for each output sample. However, if the two sampling rates are integer submultiples of a higher frequency $1/T_r$ called the reference frequency, the C_i need to be evaluated only once at the reference frequency. The phase $(mT_1 - nT_2)$ is always a multiple of T_r , and a subset of these coefficients can be utilized to perform the convolution. This reduces the computation required in the program by about an order of magnitude. For the different rates required in this project, the frequency 96000 is an integer multiple and was selected as the reference frequency.

Two versions of the resampling program were developed. In the first, all computation, initialization as well as the convolution (which comprises the bulk of the computation in resampling) is done on the PDP 11/60. In the second, the convolution is performed on the MAP 300 array processor using the SNAP-II software package of array functions supplied by CSPI. The PDP 11 does the initialization and handles the I/O interfacing between the disk and the MAP 300. The second version is almost an order of magnitude faster than the first.

E.1 Operating Details

The program is segmented into four modules. The main program acquires all relevant parameters required for the resampling operation, either from the header record of the input speech file or from the terminal. This program prompts for each input from the terminal, and if requested, gives a brief description, shows an acceptable example, and specifies the bound on the value of the input. The second module performs the convolution for resampling. The other two modules assist in the initialization by computing the convolution coefficients. Between the two versions of the program only the second module is different. The appropriate module is selected during the process of task-building.

To obtain an executable task image, the appropriate modules must be compiled and task-built. The following describes this process for the RSX11M operating system on a PDP 11 computer. The compilation step for each module has the following form.

```
>FOR MODULE = MODULE/I4
```

The task-building step is slightly different for the two versions. The command format for the first version is:

```
>TKB
TKB> QRESAM/CP/FP = MOD1, MOD2, MOD3, MOD4
TKB> /
ENTER OPTIONS:
TKB> UNITS = 7
TKB> ACTFIL = 7
TKB> //
```

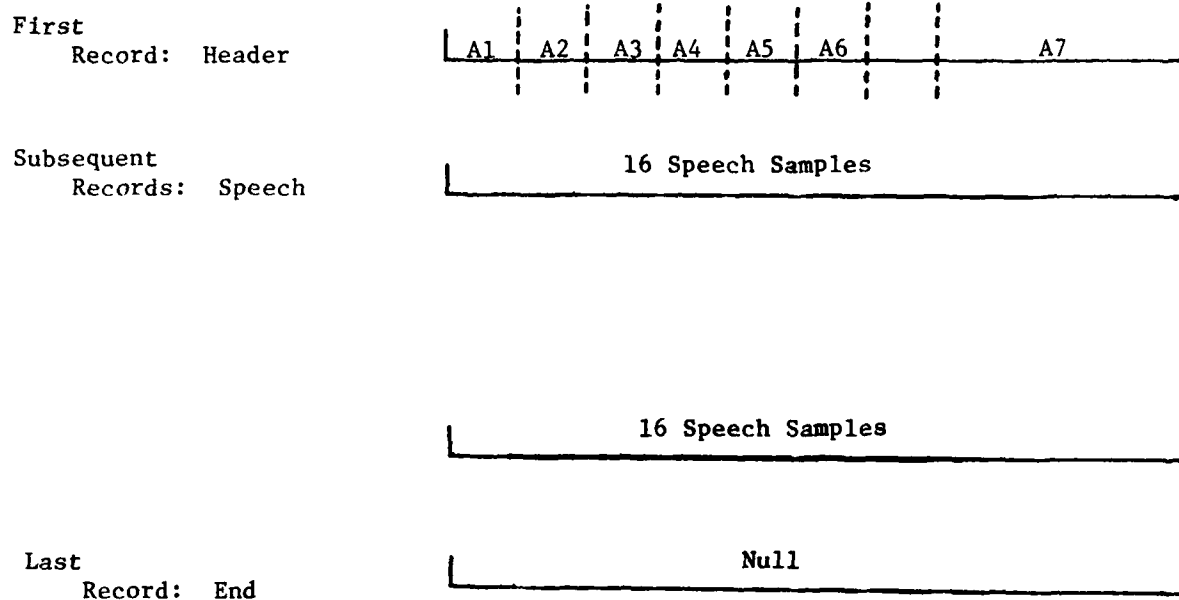
The underlined text is the prompt by the computer. The following text is the response by the user.

For the second version, the SNAP library must also be included in the task-building step.

```
>TKB
TKB> MRESAM/CP/MP = MOD1, MOD2, MOD3, MOD4, SNAPLIB/LB
TKB> /
ENTER OPTIONS:
TKB> UNITS = 10
TKB> ACTFIL = 10
TKB> //
```

The program requires the input speech file to conform to the Notre Dame speech file format. Fig. E.2 illustrates this format. The first record is the header. The seven fields in it are sentence identification number, sampling rate, number of samples in the files, lower and upper cutoff frequencies, truncation number for the convolution window, and comments identifying the speech file. The upper cutoff frequency is not used in this program. The truncation number which specifies the length of the convolution window is internally computed by the program based on the input sampling rate. Each of the subsequent records in the file contain 16 speech samples in the 15 format. The amplitude of speech samples is limited between -2048 and 2047. The speech file must end with a null record.

The output file generated by the resampling program also conforms to this format.



Format for each record of 80 columns

Header: (6I5, 10X, 2042)

Speech: (16I5)

Fig. E.2 Notre Dame Speech File Format

```

*****
*
*
*      MAINLINE INFO QRESAMPLE
*
*
*****

                ARVIND S ARORA

THIS PROGRAM SEEKS OUT THE INFORMATION NECESSARY TO
RUN THE QRESAMPLE SUBROUTINE. IT THEN OPENS THE INPUT AND OUTPUT
FILES,WRITES THE OUTPUT HEADER AND TRANSFERS CONTROL TO
QRESAMPLE. THE FOLLOWING INFORMATION IS SOUGHT:

1)NAM1      INPUT FILE NAME
2)NAM2      OUTPUT FILE NAME
3)IRATE1    INPUT SAMPLING RATE (FROM INPUT FILE HEADER)
4)IRATE2    OUTPUT SAMPLING RATE
5)NLEN1
5)NLEN1
5)NLEN2

LENGTH OF INPUT FILE (FROM INPUT FILE HEADER)
LENGTH OF OUTPUT FILE, ONLY IF PARTIAL
OUTPUT IS DESIRED
FILTER CORNER FREQ
(FILTER RESTRICTED TO INTEGER SUB-MULTIPLES OF 96000)
(IGNORE)
8)IF2      TRUNCATION NO. FOR FILTER
9)ITRUN    NO. OF TERMS IN FILTER=2*ITRUN+1
          (THIS IS DETERMINED BY THE PROGRAM DEPENDING ON IF1)
10)HEAD1   INPUT FILE HEADER COMMENTS (FROM INPUT FILE)
11)HEAD2   OUTPUT FILE HEADER COMMENTS

```

```

0001 INTEGER=2 NAM1(16),NAM2(16),HEAD1(20),HEAD2(20)
0002 LOGICAL=1 ANS,CHAR(3)
0003 COMMON /RESAM/NIEN1,NLEN2,IRATE2
0004 COMMON /COFF/IF1,IF2,ITRUM,IRATE1
0005 DATA CHAR/. . .N.V./
0006 TYPE '. . . SOLICITING INFORMATION FOR RESAMPLING.'
0007 TYPE '. . .
0008 TYPE '. . .
0009 TYPE 1009 IVANTO TO SEE EXAMPLES?
0010 ACCEPT 1003,ANS
0011 IFL=1
0012 IF(ANS.EQ.CHAR(3)) IFL=2
C ----- BEGIN QUERY
100 CONTINUE
C ----- NAME OF INPUT FILE
0014 IF(IFL.EQ.1) GO TO 110
0015 TYPE '. . . EXAMPLE OF FILE NAME *.
0016 TYPE '. . . DK:LSB.S01S11564.DAT.1'
0017 TYPE 1010 ENTER INPUT FILE NAME
0018 ACCEPT 1001,NAM1
0019 NAM1(16)=*

```

```

0020  C ----- OPEN INPUT FILE. IF ERROR IN OPENING, INDICATE ERROR
0021  C AND ASK AGAIN
0022  OPEN(UNIT=1,NAME=NAME,TYPE='OLD',READONLY,SHARED,ERR=128)
0023  READ(1,1000) ISEN,IRATE1,NLEN1,IF1,IF2,ITRUN,HEAD1
0024  ECHO INPUT FILE NAME AND INPUT HEADER
0025  TYPE 1004,NAME1
0026  TYPE 1005,ISEN,IRATE1,NLEN1,IF1,IF2,ITRUN,HEAD1
0027  GO TO 128
0028  CONTINUE
0029  TYPE '...' ERROR -- FAILED TO OPEN SPECIFIED FILE'
0030  TYPE '...' POSSIBLE REASONS: FILE DOES NOT EXIST'
0031  TYPE '...' DEVICE NOT AVAILABLE'
0032  GO TO 118
0033  CONTINUE
0034  OUTPUT FILE PARAMETERS
0035  TYPE '...' RESAMPLING PARAMETERS'
0036  OUTPUT SAMPLING RATE
0037  IF(1FL.EQ.1) GO TO 148
0038  TYPE '...' EXAMPLE OF OUTPUT SAMPLING RATE ...'
0039  TYPE '...' THE USUAL RATES FOR SPEECH ARE 6000<RATE<16000'
0040  TYPE '...' FOR EXAMPLE 6400'
0041  CONTINUE
0042  TYPE 1011 OUTPUT SAMPLING RATE
0043  ACCEPT 'IRATE2'
0044  TYPE 'IRATE2'
0045  CHECK IF RATE IS LEGAL
0046  IF(IRATE2.GT.8.AND.96000/IRATE2=IRATE2.EQ.96000) GO TO 158
0047  TYPE '...' ERROR -- CAN'T HAVE RATE<8'
0048  TYPE '...' OR SUBMULTIPLE OF 96000'
0049  GO TO 148
0050  CONTINUE
0051  CORNER FREQUENCIES
0052  IF(1FL.EQ.1) GO TO 168
0053  TYPE '...' EXAMPLE OF FILTER CORNER FREQUENCIES F1 ...'
0054  TYPE '...' THE USUAL RANGE IS RATE/2>F1>8'
0055  TYPE '...' FOR EXAMPLE 3000'
0056  CONTINUE
0057  TYPE 1012 ICORNER FREQ
0058  ACCEPT 'IF1'
0059  TYPE 'IF1'
0060  CHECK IF FREQS. ARE LEGAL
0061  IF(IF1.GT.8) GO TO 178 ILLEGAL
0062  TYPE '...' ERROR -- ILLEGAL FREQUENCY'
0063  TYPE '...' F1>8'
0064  GO TO 168
0065  CONTINUE
0066  TRUNCATION NUMBER
0067  CONTINUE
0068  ITRUN=6000*IRATE1/96000
0069  TYPE 'ITRUN'
0070  DETERMINE LENGTH OF OUTPUT FILE
0071  NLEN2=FLOAT(NLEN1)*FLOAT(IRATE2)/FLOAT(IRATE1)
0072  NBLK=NLEN2/100*1
0073  TYPE '...' THE ESTIMATED LENGTH OF THE OUTPUT FILE IS'
0074  TYPE '...' NLEN2, ' SAMPLES, OR 'NBLK, ' BLOCKS ON DISK'
0075  TYPE 1013 IASK IF PARTIAL OUTPUT WANTED

```

```

FORTRAN IV-PLUS V02-51      16:14:00      15-APR-68      PAGE 3
GRESAMINF.FTM /14/TR:BLOCKS/WR

2006 ACCEPT 1003,ANS
2007 IF(ANS.NE.CHAR(3)) GO TO 2008 ICOMPLETE OUTPUT
2008 TYPE 1014 I DESIRED LENGTH OF FILE
2009 ACCEPT ' ',NLEN2
2010 TYPE ' ',NLEN2
2011 CONTINUE
2012 C -----
2013 OUTPUT FILE NAME
2014 TYPE 1019 IPROMPT FOR OUTPUT FILE NAME
2015 ACCEPT 1001,NAM2
2016 TYPE 1004,NAM2
2017 NAM2(16)=B
2018 C -----
2019 HEADER COMMENTS
2020 TYPE ' ',** ENTER HEADER COMMENTS, 40 CHARACTERS OR LESS'
2021 IF(IFL.EQ.1) GO TO 210
2022 TYPE 1015, HEAD1
2023 CONTINUE
2024 C -----
2025 IHEADER COMMENTS PROMPT
2026 TYPE 1016 IHEADER COMMENTS PROMPT
2027 ACCEPT 1001,HEAD2
2028 TYPE ' ',** THE HEADER CARD FOR THE OUTPUT FILE IS:'
2029 TYPE 1005,ISEN,IRATE2,NLEN2,IF1,IF2,ITRUN,HEAD2
2030 ONE OPPORTUNITY TO CHANGE ENTRIES
2031 C -----
2032 TYPE 1017 IVANNA CHANGE SOMETHING
2033 ACCEPT 1003,ANS
2034 IF(ANS.EQ.CHAR(3)) GO TO 1008 I DO WANT TO CHANGE
2035 OPEN(UNIT=2,NAME=NAM2,TYPE='NEW',CARRIAGECONTROL='LIST',ERR=220)
2036 WRITE(2,1000) ISEN,IRATE2,NLEN2,IF1,IF2,ITRUN,HEAD2
2037 TYPE ' ',** BEGINNING RESAMPLING. WILL INFORM YOU ' '.
2038 I'ON COMPLETION'
2039 CALL RESAMP(1,2)
2040 STOP
2041 220 CONTINUE
2042 TYPE ' ',** ERROR -- FAILED TO OPEN OUTPUT FILE'
2043 TYPE ' ', LOCATE PROBLEM AND TRY AGAIN'
2044 STOP
2045 1000 FORMAT(615,10X,20A2)
2046 1001 FORMAT(20A2)
2047 1002 FORMAT(615)
2048 1003 FORMAT(A1)
2049 1004 FORMAT(1X,20A2)
2050 1005 FORMAT(1X,615,10X,20A2)
2051 1006 DO YOU WISH TO SEE EXAMPLES OF RESPONSES? (Y,N):'
2052 1007 FORMAT('S' ** ENTER INPUT FILE NAME. NAME='')
2053 1008 FORMAT('S' ** ENTER RESAMPLING RATE. RATE='')
2054 1009 FORMAT('S' ** ENTER CORNER FREQ F1='')
2055 1010 FORMAT('S' ** WOULD YOU LIKE A PARTIAL OUTPUT? (Y,N):'
2056 1011 FORMAT('S' ** ENTER LENGTH DESIRED. LENGTH='')
2057 1012 FORMAT('S' ** EXAMPLE:' ',20A2)
2058 1013 FORMAT('S' ** COMMENTS:')
2059 1014 DO YOU WISH TO CHANGE ENTRIES? (Y,N):'
2060 1015 FORMAT('S' ** ENTER TRUNCATION NUMBER. ITRUN='')
2061 1016 FORMAT('S' ** ENTER OUTPUT FILE NAME. NAME='')
2062 1017
2063 1018
2064 1019
2065 1020
2066 1021
2067 1022
2068 1023
2069 1024
2070 1025
2071 1026
2072 1027
2073 1028
2074 1029
2075 1030
2076 1031
2077 1032
2078 1033
2079 1034
2080 1035
2081 1036
2082 1037
2083 1038
2084 1039
2085 1040
2086 1041
2087 1042
2088 1043
2089 1044
2090 1045
2091 1046
2092 1047
2093 1048
2094 1049
2095 1050
2096 1051
2097 1052
2098 1053
2099 1054
2100 1055
2101 1056
2102 1057
2103 1058
2104 1059
2105 1060
2106 1061
2107 1062
2108 1063
2109 1064
2110 1065
2111 1066
2112 1067
2113 1068
2114 1069
2115 1070
2116 1071
2117 1072
2118 1073
2119 1074
2120 1075
2121 1076
2122 1077
2123 1078
2124 1079
2125 1080
2126 1081
2127 1082
2128 1083
2129 1084
2130 1085
2131 1086
2132 1087
2133 1088
2134 1089
2135 1090
2136 1091
2137 1092
2138 1093
2139 1094
2140 1095
2141 1096
2142 1097
2143 1098
2144 1099
2145 1100
2146 1101
2147 1102
2148 1103
2149 1104
2150 1105
2151 1106
2152 1107
2153 1108
2154 1109
2155 1110
2156 1111
2157 1112
2158 1113
2159 1114
2160 1115
2161 1116
2162 1117
2163 1118
2164 1119
2165 1120
2166 1121
2167 1122
2168 1123
2169 1124
2170 1125
2171 1126
2172 1127
2173 1128
2174 1129
2175 1130
2176 1131
2177 1132
2178 1133
2179 1134
2180 1135
2181 1136
2182 1137
2183 1138
2184 1139
2185 1140
2186 1141
2187 1142
2188 1143
2189 1144
2190 1145
2191 1146
2192 1147
2193 1148
2194 1149
2195 1150
2196 1151
2197 1152
2198 1153
2199 1154
2200 1155
2201 1156
2202 1157
2203 1158
2204 1159
2205 1160
2206 1161
2207 1162
2208 1163
2209 1164
2210 1165
2211 1166
2212 1167
2213 1168
2214 1169
2215 1170
2216 1171
2217 1172
2218 1173
2219 1174
2220 1175
2221 1176
2222 1177
2223 1178
2224 1179
2225 1180
2226 1181
2227 1182
2228 1183
2229 1184
2230 1185
2231 1186
2232 1187
2233 1188
2234 1189
2235 1190
2236 1191
2237 1192
2238 1193
2239 1194
2240 1195
2241 1196
2242 1197
2243 1198
2244 1199
2245 1200
2246 1201
2247 1202
2248 1203
2249 1204
2250 1205
2251 1206
2252 1207
2253 1208
2254 1209
2255 1210
2256 1211
2257 1212
2258 1213
2259 1214
2260 1215
2261 1216
2262 1217
2263 1218
2264 1219
2265 1220
2266 1221
2267 1222
2268 1223
2269 1224
2270 1225
2271 1226
2272 1227
2273 1228
2274 1229
2275 1230
2276 1231
2277 1232
2278 1233
2279 1234
2280 1235
2281 1236
2282 1237
2283 1238
2284 1239
2285 1240
2286 1241
2287 1242
2288 1243
2289 1244
2290 1245
2291 1246
2292 1247
2293 1248
2294 1249
2295 1250
2296 1251
2297 1252
2298 1253
2299 1254
2300 1255
2301 1256
2302 1257
2303 1258
2304 1259
2305 1260
2306 1261
2307 1262
2308 1263
2309 1264
2310 1265
2311 1266
2312 1267
2313 1268
2314 1269
2315 1270
2316 1271
2317 1272
2318 1273
2319 1274
2320 1275
2321 1276
2322 1277
2323 1278
2324 1279
2325 1280
2326 1281
2327 1282
2328 1283
2329 1284
2330 1285
2331 1286
2332 1287
2333 1288
2334 1289
2335 1290
2336 1291
2337 1292
2338 1293
2339 1294
2340 1295
2341 1296
2342 1297
2343 1298
2344 1299
2345 1300
2346 1301
2347 1302
2348 1303
2349 1304
2350 1305
2351 1306
2352 1307
2353 1308
2354 1309
2355 1310
2356 1311
2357 1312
2358 1313
2359 1314
2360 1315
2361 1316
2362 1317
2363 1318
2364 1319
2365 1320
2366 1321
2367 1322
2368 1323
2369 1324
2370 1325
2371 1326
2372 1327
2373 1328
2374 1329
2375 1330
2376 1331
2377 1332
2378 1333
2379 1334
2380 1335
2381 1336
2382 1337
2383 1338
2384 1339
2385 1340
2386 1341
2387 1342
2388 1343
2389 1344
2390 1345
2391 1346
2392 1347
2393 1348
2394 1349
2395 1350
2396 1351
2397 1352
2398 1353
2399 1354
2400 1355
2401 1356
2402 1357
2403 1358
2404 1359
2405 1360
2406 1361
2407 1362
2408 1363
2409 1364
2410 1365
2411 1366
2412 1367
2413 1368
2414 1369
2415 1370
2416 1371
2417 1372
2418 1373
2419 1374
2420 1375
2421 1376
2422 1377
2423 1378
2424 1379
2425 1380
2426 1381
2427 1382
2428 1383
2429 1384
2430 1385
2431 1386
2432 1387
2433 1388
2434 1389
2435 1390
2436 1391
2437 1392
2438 1393
2439 1394
2440 1395
2441 1396
2442 1397
2443 1398
2444 1399
2445 1400
2446 1401
2447 1402
2448 1403
2449 1404
2450 1405
2451 1406
2452 1407
2453 1408
2454 1409
2455 1410
2456 1411
2457 1412
2458 1413
2459 1414
2460 1415
2461 1416
2462 1417
2463 1418
2464 1419
2465 1420
2466 1421
2467 1422
2468 1423
2469 1424
2470 1425
2471 1426
2472 1427
2473 1428
2474 1429
2475 1430
2476 1431
2477 1432
2478 1433
2479 1434
2480 1435
2481 1436
2482 1437
2483 1438
2484 1439
2485 1440
2486 1441
2487 1442
2488 1443
2489 1444
2490 1445
2491 1446
2492 1447
2493 1448
2494 1449
2495 1450
2496 1451
2497 1452
2498 1453
2499 1454
2500 1455
2501 1456
2502 1457
2503 1458
2504 1459
2505 1460
2506 1461
2507 1462
2508 1463
2509 1464
2510 1465
2511 1466
2512 1467
2513 1468
2514 1469
2515 1470
2516 1471
2517 1472
2518 1473
2519 1474
2520 1475
2521 1476
2522 1477
2523 1478
2524 1479
2525 1480
2526 1481
2527 1482
2528 1483
2529 1484
2530 1485
2531 1486
2532 1487
2533 1488
2534 1489
2535 1490
2536 1491
2537 1492
2538 1493
2539 1494
2540 1495
2541 1496
2542 1497
2543 1498
2544 1499
2545 1500
2546 1501
2547 1502
2548 1503
2549 1504
2550 1505
2551 1506
2552 1507
2553 1508
2554 1509
2555 1510
2556 1511
2557 1512
2558 1513
2559 1514
2560 1515
2561 1516
2562 1517
2563 1518
2564 1519
2565 1520
2566 1521
2567 1522
2568 1523
2569 1524
2570 1525
2571 1526
2572 1527
2573 1528
2574 1529
2575 1530
2576 1531
2577 1532
2578 1533
2579 1534
2580 1535
2581 1536
2582 1537
2583 1538
2584 1539
2585 1540
2586 1541
2587 1542
2588 1543
2589 1544
2590 1545
2591 1546
2592 1547
2593 1548
2594 1549
2595 1550
2596 1551
2597 1552
2598 1553
2599 1554
2600 1555
2601 1556
2602 1557
2603 1558
2604 1559
2605 1560
2606 1561
2607 1562
2608 1563
2609 1564
2610 1565
2611 1566
2612 1567
2613 1568
2614 1569
2615 1570
2616 1571
2617 1572
2618 1573
2619 1574
2620 1575
2621 1576
2622 1577
2623 1578
2624 1579
2625 1580
2626 1581
2627 1582
2628 1583
2629 1584
2630 1585
2631 1586
2632 1587
2633 1588
2634 1589
2635 1590
2636 1591
2637 1592
2638 1593
2639 1594
2640 1595
2641 1596
2642 1597
2643 1598
2644 1599
2645 1600
2646 1601
2647 1602
2648 1603
2649 1604
2650 1605
2651 1606
2652 1607
2653 1608
2654 1609
2655 1610
2656 1611
2657 1612
2658 1613
2659 1614
2660 1615
2661 1616
2662 1617
2663 1618
2664 1619
2665 1620
2666 1621
2667 1622
2668 1623
2669 1624
2670 1625
2671 1626
2672 1627
2673 1628
2674 1629
2675 1630
2676 1631
2677 1632
2678 1633
2679 1634
2680 1635
2681 1636
2682 1637
2683 1638
2684 1639
2685 1640
2686 1641
2687 1642
2688 1643
2689 1644
2690 1645
2691 1646
2692 1647
2693 1648
2694 1649
2695 1650
2696 1651
2697 1652
2698 1653
2699 1654
2700 1655
2701 1656
2702 1657
2703 1658
2704 1659
2705 1660
2706 1661
2707 1662
2708 1663
2709 1664
2710 1665
2711 1666
2712 1667
2713 1668
2714 1669
2715 1670
2716 1671
2717 1672
2718 1673
2719 1674
2720 1675
2721 1676
2722 1677
2723 1678
2724 1679
2725 1680
2726 1681
2727 1682
2728 1683
2729 1684
2730 1685
2731 1686
2732 1687
2733 1688
2734 1689
2735 1690
2736 1691
2737 1692
2738 1693
2739 1694
2740 1695
2741 1696
2742 1697
2743 1698
2744 1699
2745 1700
2746 1701
2747 1702
2748 1703
2749 1704
2750 1705
2751 1706
2752 1707
2753 1708
2754 1709
2755 1710
2756 1711
2757 1712
2758 1713
2759 1714
2760 1715
2761 1716
2762 1717
2763 1718
2764 1719
2765 1720
2766 1721
2767 1722
2768 1723
2769 1724
2770 1725
2771 1726
2772 1727
2773 1728
2774 1729
2775 1730
2776 1731
2777 1732
2778 1733
2779 1734
2780 1735
2781 1736
2782 1737
2783 1738
2784 1739
2785 1740
2786 1741
2787 1742
2788 1743
2789 1744
2790 1745
2791 1746
2792 1747
2793 1748
2794 1749
2795 1750
2796 1751
2797 1752
2798 1753
2799 1754
2800 1755
2801 1756
2802 1757
2803 1758
2804 1759
2805 1760
2806 1761
2807 1762
2808 1763
2809 1764
2810 1765
2811 1766
2812 1767
2813 1768
2814 1769
2815 1770
2816 1771
2817 1772
2818 1773
2819 1774
2820 1775
2821 1776
2822 1777
2823 1778
2824 1779
2825 1780
2826 1781
2827 1782
2828 1783
2829 1784
2830 1785
2831 1786
2832 1787
2833 1788
2834 1789
2835 1790
2836 1791
2837 1792
2838 1793
2839 1794
2840 1795
2841 1796
2842 1797
2843 1798
2844 1799
2845 1800
2846 1801
2847 1802
2848 1803
2849 1804
2850 1805
2851 1806
2852 1807
2853 1808
2854 1809
2855 1810
2856 1811
2857 1812
2858 1813
2859 1814
2860 1815
2861 1816
2862 1817
2863 1818
2864 1819
2865 1820
2866 1821
2867 1822
2868 1823
2869 1824
2870 1825
2871 1826
2872 1827
2873 1828
2874 1829
2875 1830
2876 1831
2877 1832
2878 1833
2879 1834
2880 1835
2881 1836
2882 1837
2883 1838
2884 1839
2885 1840
2886 1841
2887 1842
2888 1843
2889 1844
2890 1845
2891 1846
2892 1847
2893 1848
2894 1849
2895 1850
2896 1851
2897 1852
2898 1853
2899 1854
2900 1855
2901 1856
2902 1857
2903 1858
2904 1859
2905 1860
2906 1861
2907 1862
2908 1863
2909 1864
2910 1865
2911 1866
2912 1867
2913 1868
2914 1869
2915 1870
2916 1871
2917 1872
2918 1873
2919 1874
2920 1875
2921 1876
2922 1877
2923 1878
2924 1879
2925 1880
2926 1881
2927 1882
2928 1883
2929 1884
2930 1885
2931 1886
2932 1887
2933 1888
2934 1889
2935 1890
2936 1891
2937 1892
2938 1893
2939 1894
2940 1895
2941 1896
2942 1897
2943 1898
2944 1899
2945 1900
2946 1901
2947 1902
2948 1903
2949 1904
2950 1905
2951 1906
2952 1907
2953 1908
2954 1909
2955 1910
2956 1911
2957 1912
2958 1913
2959 1914
2960 1915
2961 1916
2962 1917
2963 1918
2964 1919
2965 1920
2966 1921
2967 1922
2968 1923
2969 1924
2970 1925
2971 1926
2972 1927
2973 1928
2974 1929
2975 1930
2976 1931
2977 1932
2978 1933
2979 1934
2980 1935
2981 1936
2982 1937
2983 1938
2984 1939
2985 1940
2986 1941
2987 1942
2988 1943
2989 1944
2990 1945
2991 1946
2992 1947
2993 1948
2994 1949
2995 1950
2996 1951
2997 1952
2998 1953
2999 1954
3000 1955
3001 1956
3002 1957
3003 1958
3004 1959
3005 1960
3006 1961
3007 1962
3008 1963
3009 1964
3010 1965
3011 1966
3012 1967
3013 1968
3014 1969
3015 1970
3016 1971
3017 1972
3018 1973
3019 1974
3020 1975
3021 1976
3022 1977
3023 1978
3024 1979
3025 1980
3026 1981
3027 1982
3028 1983
3029 1984
3030 1985
3031 1986
3032 1987
3033 1988
3034 1989
3035 1990
3036 1991
3037 1992
3038 1993
3039 1994
3040 1995
3041 1996
3042 1997
3043 1998
3044 1999
3045 2000
3046 2001
3047 2002
3048 2003
3049 2004
3050 2005
3051 2006
3052 2007
3053 2008
3054 2009
3055 2010
3056 2011
3057 2012
3058 2013
3059 2014
3060 2015
3061 2016
3062 2017
3063 2018
3064 2019
3065 2020
3066 2021
3067 2022
3068 2023
3069 2024
3070 2025
3071 2026
3072 2027
3073 2028
3074 2029
3075 2030
3076 2031
3077 2032
3078 2033
3079 2034
3080 2035
3081 2036
3082 2037
3083 2038
3084 2039
3085 2040
3086 2041
3087 2042
3088 2043
3089 2044
3090 2045
3091 2046
3092 2047
3093 2048
3094 2049
3095 2050
3096 2051
3097 2052
3098 2053
3099 2054
3100 2055
3101 2056
3102 2057
3103 2058
3104 2059
3105 2060
3106 2061
3107 2062
3108 2063
3109 2064
3110 2065
3111 2066
3112 2067
3113 2068
3114 2069
3115 2070
3116 2071
3117 2072
3118 2073
3119 2074
3120 2075
3121 2076
3122 2077
3123 2078
3124 2079
3125 2080
3126 2081
3127 2082
3128 2083
3129 2084
3130 2085
3131 2086
3132 2087
3133 2088
3134 2089
3135 2090
3136 2091
3137 2092
3138 2093
3139 2094
3140 2095
3141 2096
3142 2097
3143 2098
3144 2099
3145 2100
3146 2101
3147 2102
3148 2103
3149 2104
3150 2105
3151 2106
3152 2107
3153 2108
3154 2109
3155 2110
3156 2111
3157 2112
3158 2113
3159 2114
3160 2115
3161 2116
3162 2117
3163 2118
3164 2119
3165 2120
3166 2121
3167 2122
3168 2123
3169 2124
3170 2125
3171 2126
3172 2127
3173 2128
3174 2129
3175 2130
3176 2131
3177 2132
3178 2133
3179 2134
3180 2135
3181 2136
3182 2137
3183 2138
3184 2139
3185 2140
3186 2141
3187 2142
3188 2143
3189 2144
3190 2145
3191 2146
3192 21
```

```

*****
*
* SUBROUTINE RESAMP(NIN,NOUT)
*
*****

ARVIND S ARORA

PROGRAM DESCRIPTION:

THIS PROGRAM RESAMPLES A GIVEN FILE AT ONE SAMPLING
RATE TO THAT AT ANOTHER SAMPLING RATE. BECAUSE THE COEFFS
REQUIRED FOR THE FILTERING OPERATION DURING RESAMPLING ARE
COMPUTED ONLY ONCE AT 96000 SAMPLES/SEC, THE INPUT AND
OUTPUT FREQUENCIES ARE RESTRICTED TO INTEGER SUB-MULTIPLES
OF 96000, E.G. 16000, 12000, 9600, 6400. HOWEVER THIS MAKES
THIS PROGRAM OPERATE AN ORDER OF MAGNITUDE FASTER THAN THE
GENERAL VERSION OF RESAMP. ALL COMPUTATIONS IN THIS PROGRAM
ARE DONE IN THE PDP-11 CPU.

THE PROGRAM REQUIRES THE FOLLOWING INFORMATION BEFORE IT
CAN RESAMPLE A GIVEN FILE:

1) NIN LUN FOR INPUT FILE
2) NOUT LUN FOR OUTPUT FILE
3) NLEN1 LENGTH OF INPUT FILE
4) NLEN2 LENGTH OF OUTPUT FILE
5) IRATE1 SAMPLING RATE OF INPUT FILE
6) IRATE2 SAMPLING RATE OF OUTPUT FILE
7) IF1 CORNER FREQ OF FILTER
8) IF2 (IGNORED)
9) ITRUN (IGNORED)

IN ADDITION THE INPUT AND OUTPUT FILES MUST BE OPEN
FOR ACCESS.

DATA FORMATS:

BOTH THE INPUT & OUTPUT SAMPLES MUST HAVE VALUES V IN THE
RANGE -2048 TO 2047 (12 BIT RESOLUTION). THEY MUST BE STORED
USING A (1615) FORMAT.

SIGNIFICANT VARIABLES:

1) A1 INPUT FILE BUFFER (INTEGER*2)
2) A2 OUTPUT FILE BUFFER (INTEGER*2)
3) H FILTER COEFFS (REAL)
SYMMETRIC, CENTRE COEFF IN H(1),
COEFFS ONLY FOR +VE TIME

```

FORTRAN IV-PLUS V92-S1 16:14:44 15-APR-88 PAGE 2
 QRESAMP.FTN /I4/TR:BLOCKS/VR

C CALLS TO SUBROUTINES:

1)SUBROUTINE ERRSET(24,...,FALSE...)
 2)FUNCTION-SUBROUTINE SECNDS(TIME)
 3)SUBROUTINE COEFF(H??)

```

9991 SUBROUTINE RESAMP(NIN,NOUT)
9992   INTEGER*2 A1(32),A2(16)
9993   DIMENSION H(64)
9994   COMMON /RESAMP/NLEN1,NLEN2,IRATE2
9995   COMMON /COFF/IF1,IF2,ITRUN,IRATE1
9996   CONTINUE
9997   NLEN2=NLEN2-1
9998   IF(96888/IRATE1=IRATE1.NE.96888) GO TO 118
9999   IF(96888/IRATE2=IRATE2.NE.96888) GO TO 118
1000   GO TO 128
1001   C ----- INDICATE ILLEGAL FREQUENCIES AND STOP
1002   118 CONTINUE
1003   CLOSE(UNIT=NIN,DISPOSE='SAVE')
1004   CLOSE(UNIT=NOUT,DISPOSE='DELETE')
1005   TYPE '*** ERROR QUICK RESAMPLE.'
1006   TYPE '...'
1007   TYPE '...THIS SUBROUTINE HANDLES ONLY A LIMITED SET OF FREQS.'
1008   TYPE '...INPUT AND OUTPUT FREQS MUST BE SUBMULTIPLES OF 96888'
1009   TYPE '...AND YOUR FREQS ARE, IN FREQ=,IRATE1, OUT FREQ=,IRATE2'
1010   STOP
1011   128 CONTINUE
1012   CALL ERRSET(24,...,FALSE...)
1013   C ----- INITIALIZE CLOCK TO COMPUTE TIME REQUIRED FOR RESAMPLING
1014   CLCK=SECNDS(8.8)
1015   288 CONTINUE
1016   C ----- INITIALIZE ALL VARIABLES AND THE INPUT BUFFER
1017   INTV1=96888/IRATE1
1018   INTV2=96888/IRATE2
1019   NBUF1=328
1020   NBUF2=168
1021   IHOLD=ITRUN
1022   ITRUN=688/INTV1
1023   NFLG1=ITRUN+1
1024   NFLG2=NBUF1-ITRUN
1025   IBUF1=NFLG1+MOD((NBUF1-NFLG1+1),16)
1026   DO 218 IAI=1,IBUF1-1
1027     A1(IAI)=8
1028   218 CONTINUE
1029   READ(NIN,1888)(A1(IAI),IAI=IBUF1,NBUF1)
1030   C ----- INITIALIZE INDICES
1031   IA1=8
1032   IA2=8
1033   IBUF2=1
1034   C ----- CAL SUBROUTINE TO COMPUTE COEFFICIENTS
1035   CALL COEFF(H)
1036   INTVN=INTV1+1

```

```

TRA--PL--B2-16--44--5-A--AGE
CRSAMP.FIN /14/TR:BLOCKS/WR

C ----- BEGIN CONVOLUTION LOOP FOR FILTERING
22# CONTINUE
C ----- COMPUTE PHASE AND INITIALIZE INDICES FOR CONVOLUTION
IPHASE=IA2-INTV2-IA1*INTV1
IIN=IBUF1-1
IIP=IBUF1+1
IHN=INTVN+IPHASE
IHP=INTVN-IPHASE
C ----- PERFORM CONVOLUTION
SUM=H(1+IPHASE)*A1(IBUF1)
DO 25# IJ=IHN,691,INTV1
SUM=SUM+H(IJ)*A1(IIN)+H(IHP)*A1(IIP)
IIN=IIN-1
IIP=IIP+1
IHP=IHP+INTV1
25# CONTINUE
ITEMP=SUM
IF(ITEMP.GE.2948) ITEM=2947
IF(ITEMP.LT.-2948) ITEM=-2948
A2(IBUF2)=ITEMP
C ----- CONVOLUTION COMPLETED
C
C ----- INCREMENT VARIOUS INDICES
IA2=IA2+1
IBUF2=IBUF2+1
C ----- FIND CORRESPONDING INDICES IN THE INPUT FILE
INEXT=INTV2-IA2/INTV1
IBUF1=IBUF1+INEXT-IA1
IA1=INEXT
C ----- CHECK IF OUTPUT FILE IS COMPLETE
IF(IA2.LE.NLEN2) GO TO 39# IIF NOT COMPLETE, PROCEED
IF FILE COMPLETE, WRITE OUTPUT BUFFER, CLOSE FILES & QUIT
WRITE(NOUT,1999) (A2(J),J=1,IBUF2-1)
NLEN2=NLEN2+1
ITRUN=I HOLD
C ----- FIGURE OUT TIME TAKEN, SAY OPERATION COMPLETE & QUIT
CLICK=SECONDS(CLCK)
TYPE '...', TIME FOR RESAMPLING = '.CLCK.' SECONDS'
TYPE '...',
TYPE '...',
TYPE '...', RESAMPLING OPERATION COMPLETE.'
RETURN
C ----- CHECK IF OUTPUT BUFFER IS FULL
39# CONTINUE
IF(IBUF2.LE.NBUF2) GO TO 31# IIF NOT FULL, PROCEED
IF FULL, WRITE OUT OUTPUT BUFFER & INITIALIZE FLAG IBUF2
WRITE(NOUT,1999) A2
IBUF2=1
C ----- CHECK IF RUN OUT OF INPUT BUFFER
31# CONTINUE
IF(IBUF1.LE.NFLG2) GO TO 35# IIF NOT RUN OUT, PROCEED
IF RUN OUT, REPLENISH BUFFER BY MOVING OLD DATA OVER
IFLNXT=NFLG1+MOD((IBUF1-NFLG1),16)
ISHFT=IBUF1-IFLNXT
IBUF1=IFLNXT
DO 32# J=ISHFT+1,NBUF1

```

```

FORTRAN IV-PLUS V02-51      16:14:44      15-APR-88      PAGE 4
QRESAMP.FTN /14/TR:BLOCKS/VR

2004      A1(J-ISHFT)=A1(J)
2005      CONTINUE
2006      C -----
2007      FILL IN THE REST OF THE BUFFER WITH NEW DATA
2008      READ(NIN,1000,ERR=330) (A1(J),J=NBUFF1-ISHFT+1,NBUFF1)
2009      GO TO 350 IREAD IN BUFFER, SO PROCEED
2010      C -----
2011      RAN OUT OF INPUT SAMPLES SO FILL IN BUFFER WITH ZEROS
2012      CONTINUE
2013      DO 340 J1=J,NBUFF1
2014      A1(J1)=0
2015      CONTINUE
2016      C -----
2017      ALL SYSTEMS GO 1 BACK FOR NEXT SAMPLE
2018      GO TO 220
2019      C -----
2020      ALL THOSE LOUSY FORMAT STATEMENTS
2021      C
2022      1000    FORMAT(16I5)
2023      END
2024

```


FORTRAN IV-PLUS V82-51 16:15:21 15-APR-88 PAGE 2
 MAPRESAM1.FTN /14/TR:BLOCKS/VR

```

C SIGNIFICANT VARIABLES:
C
C 1) A1 INPUT FILE BUFFER (INTEGER*2)
C 2) A2 OUTPUT FILE BUFFER (INTEGER*2)
C 3) M FILTER COEFFS (REAL)
C SYMMETRIC. CENTRE COEFF IN H(1),
C COEFFS ONLY FOR +VE TIME
C
C CALLS TO SUBROUTINES:
C
C 1) SUBROUTINE ERRSET(24,.....FALSE..)
C 2) FUNCTION-SUBROUTINE SECNDS(TIME)
C 3) SUBROUTINE COEFF(H?)
C 4) SUBROUTINES IN SNAP LIBRARY
C
C .....
C
C SUBROUTINE RESAMP(NIN,NOUT)
C INTEGER*2 A1(648),A2(648)
C INTEGER SP2847,SN2848,SFLAG
C DIMENSION H(681)
C COMMON /RESAMP/NLEN1,NLEN2,IRATE2
C COMMON /COFF/IF1,IF2,ITRUN,IRATE1
C CONTINUE
C
C ----- NAMES OF VARIOUS BUFFERS IN MAP AND HOST
C NMRI=1 1 INPUT BUFFER IN HOST: A1
C NMRI=2 1 OUTPUT BUFFER IN HOST: A2
C
C NMRI=28
C MB1=21
C MB2=21
C MB12=22
C MB3=23
C MB4=24
C MB5=25
C MB6=26
C MB13=27
C MZERO=29
C MORI=11
C MOR1=12
C MOR2=13
C MOR12=14
C MCB=38
C MC1=31
C MC2=32
C
C ----- NAMES OF VARIOUS SCALARS IN MAP
C SP2847=58
C SN2848=51
C SFLAG=52
C IF(96888/IRATE1*IRATE1.NE.96888) GO TO 118
C IF(96888/IRATE2*IRATE2.NE.96888) GO TO 118
C IF(1181*GT.32888.OR.IRATE1.LT.6488) GO TO 118
C IF(1182*GT.32888.OR.IRATE2.LT.6488) GO TO 118
C GO TO 128

```

```

C ----- INDICATE ILLEGAL FREQUENCIES AND STOP
110 CONTINUE
CLOSE(UNIT=NIN,DISPOSE='SAVE')
CLOSE(UNIT=NOUT,DISPOSE='DELETE')
TYPE '*** ERROR QUICK RESAMPLE.'
TYPE '***'
TYPE '*** THIS SUBROUTINE HANDLES ONLY A LIMITED SET OF FREQS. '
TYPE '*** INPUT AND OUTPUT FREQS MUST BE SUBMULTIPLES OF 96000.'
TYPE '*** AND MUST LIE BETWEEN 6400 AND 32000.'
TYPE '*** AND YOUR FREQS ARE, IN FREQ=,IRATE1, OUT FREQ=,IRATE2
STOP

120 CONTINUE
CALL ERASET(24,.....FALSE,.)
INITIALIZE CLOCK TO COMPUTE TIME REQUIRED FOR RESAMPLING
CLK=SECONDS(0.0)
C ----- CALL SUBROUTINE TO COMPUTE COEFFICIENTS
C ----- CALL SUBROUTINE TO COMPUTE COEFFICIENTS
200 C ----- CONVOLUTION LOOP STARTING #200
CONTINUE
INTV1=96000/IRATE1
INTV2=96000/IRATE2
NPACK1=1920/INTV1
NPACK2=1920/INTV2
C ----- OPEN MAP
C ----- CALL REPORT(10,MPOPN(0))
C ----- CONFIGURE INPUT AND CONVOLUTION BUFFERS
CALL REPORT(20,MPCLB(MB1,0,3120,0,1,0))
CALL REPORT(30,MPCLB(MB2,5040,0,0,0,0,0,INTV1,0))
CALL REPORT(40,MPCLB(MB12,5040,0,0,0,0,0,INTV1,0))
CALL REPORT(50,MPCLB(MB13,1200,0,0,0,0,0,INTV1,0))
CALL REPORT(55,MPCLB(MB13,1200,0,0,0,0,0,INTV1,0))
SIZE=000/INTV1-1
CALL REPORT(60,MPCLB(MB4,1200,0,0,0,0,0,INTV1,0))
CALL REPORT(70,MPEOB(MB5,MBR2))
CALL REPORT(80,MPCLB(MB6,5040,0,0,0,0,0,INTV1,0))
C ----- CONFIGURE OUTPUT BUFFERS
CALL REPORT(90,MPCLB(29,9000,0,0,0,0,0,1,0))
CALL REPORT(100,MPEOB(MB1,29))
CALL REPORT(110,MPCLB(29,11000,0,0,0,0,0,1,0))
CALL REPORT(120,MPEOB(MB2,29))
CALL REPORT(130,MPCLB(29,9000,0,0,0,0,0,2,0))
CALL REPORT(140,MPEOB(MB1,29))
CALL REPORT(150,MPCLB(29,11000,0,0,0,0,0,2,0))
CALL REPORT(160,MPEOB(MB2,29))
C ----- CONFIGURE COEFF BUFFERS
CALL REPORT(170,MPCLB(MC0,0,1200,0,0,1,0))
CALL REPORT(180,MPCLB(MC1,1200,0,0,1,0))
CALL REPORT(190,MPCLB(MC2,1200,0,0,1,0))
C ----- CONFIGURE MOST RESIDENT BUFFERS
CALL REPORT(200,MPODN(MB1,1,1),A1(NPACK1))
CALL REPORT(202,MPODN(MB2,A2(1),A2(NPACK2))
INITIALIZE INPUT BUFFER AREA TO ZERO
CALL REPORT(205,MPCLB(29,0,4500,0,1,0))
CALL REPORT(206,MPCV(MZERO,0,4500,0))
CALL REPORT(207,VSM1(28,0,MZERO,0))
C ----- WRITE IN THE SCALARS

```

```

0070 CALL REPORT(211,MPVST(SP2B47,2B47,.1,1))
0080 CALL REPORT(212,MPVST(SN2B48,-2B48,.1,1))
0090 CALL REPORT(213,MPVST(SFLAG,1B,.1,1))
0100 C ----- COPY COEFFS INTO COEFF BUFFER
0110 CALL REPORT(220,MPVDB(MC1,H(1),4,1,H(51)))
0120 CALL REPORT(230,MPVDB(MC2,H(1),4,1,H(51)))

```

```

0004
0006
0008
0010
0012
0014
0016
0018
0020
0022
0024
0026
0028
0030
0032
0034
0036
0038
0040
0042
0044
0046
0048
0050
0052
0054
0056
0058
0060
0062
0064
0066
0068
0070
0072
0074
0076
0078
0080
0082
0084
0086
0088
0090
0092
0094
0096
0098
0100
0102
0104
0106
0108
0110
0112
0114
0116
0118
0120
0122
0124
0126
0128
0130
0132
0134
0136
0138
0140
0142
0144
0146
0148
0150
0152
0154
0156
0158
0160
0162
0164
0166
0168
0170
0172
0174
0176
0178
0180
0182
0184
0186
0188
0190
0192
0194
0196
0198
0200
0202
0204
0206
0208
0210
0212
0214
0216
0218
0220
0222
0224
0226
0228
0230
0232
0234
0236
0238
0240
0242
0244
0246
0248
0250
0252
0254
0256
0258
0260
0262
0264
0266
0268
0270
0272
0274
0276
0278
0280
0282
0284
0286
0288
0290
0292
0294
0296
0298
0300
0302
0304
0306
0308
0310
0312
0314
0316
0318
0320
0322
0324
0326
0328
0330
0332
0334
0336
0338
0340
0342
0344
0346
0348
0350
0352
0354
0356
0358
0360
0362
0364
0366
0368
0370
0372
0374
0376
0378
0380
0382
0384
0386
0388
0390
0392
0394
0396
0398
0400
0402
0404
0406
0408
0410
0412
0414
0416
0418
0420
0422
0424
0426
0428
0430
0432
0434
0436
0438
0440
0442
0444
0446
0448
0450
0452
0454
0456
0458
0460
0462
0464
0466
0468
0470
0472
0474
0476
0478
0480
0482
0484
0486
0488
0490
0492
0494
0496
0498
0500
0502
0504
0506
0508
0510
0512
0514
0516
0518
0520
0522
0524
0526
0528
0530
0532
0534
0536
0538
0540
0542
0544
0546
0548
0550
0552
0554
0556
0558
0560
0562
0564
0566
0568
0570
0572
0574
0576
0578
0580
0582
0584
0586
0588
0590
0592
0594
0596
0598
0600
0602
0604
0606
0608
0610
0612
0614
0616
0618
0620
0622
0624
0626
0628
0630
0632
0634
0636
0638
0640
0642
0644
0646
0648
0650
0652
0654
0656
0658
0660
0662
0664
0666
0668
0670
0672
0674
0676
0678
0680
0682
0684
0686
0688
0690
0692
0694
0696
0698
0700
0702
0704
0706
0708
0710
0712
0714
0716
0718
0720
0722
0724
0726
0728
0730
0732
0734
0736
0738
0740
0742
0744
0746
0748
0750
0752
0754
0756
0758
0760
0762
0764
0766
0768
0770
0772
0774
0776
0778
0780
0782
0784
0786
0788
0790
0792
0794
0796
0798
0800
0802
0804
0806
0808
0810
0812
0814
0816
0818
0820
0822
0824
0826
0828
0830
0832
0834
0836
0838
0840
0842
0844
0846
0848
0850
0852
0854
0856
0858
0860
0862
0864
0866
0868
0870
0872
0874
0876
0878
0880
0882
0884
0886
0888
0890
0892
0894
0896
0898
0900
0902
0904
0906
0908
0910
0912
0914
0916
0918
0920
0922
0924
0926
0928
0930
0932
0934
0936
0938
0940
0942
0944
0946
0948
0950
0952
0954
0956
0958
0960
0962
0964
0966
0968
0970
0972
0974
0976
0978
0980
0982
0984
0986
0988
0990
0992
0994
0996
0998
1000
1002
1004
1006
1008
1010
1012
1014
1016
1018
1020
1022
1024
1026
1028
1030
1032
1034
1036
1038
1040
1042
1044
1046
1048
1050
1052
1054
1056
1058
1060
1062
1064
1066
1068
1070
1072
1074
1076
1078
1080
1082
1084
1086
1088
1090
1092
1094
1096
1098
1100
1102
1104
1106
1108
1110
1112
1114
1116
1118
1120
1122
1124
1126
1128
1130
1132
1134
1136
1138
1140
1142
1144
1146
1148
1150
1152
1154
1156
1158
1160
1162
1164
1166
1168
1170
1172
1174
1176
1178
1180
1182
1184
1186
1188
1190
1192
1194
1196
1198
1200
1202
1204
1206
1208
1210
1212
1214
1216
1218
1220
1222
1224
1226
1228
1230
1232
1234
1236
1238
1240
1242
1244
1246
1248
1250
1252
1254
1256
1258
1260
1262
1264
1266
1268
1270
1272
1274
1276
1278
1280
1282
1284
1286
1288
1290
1292
1294
1296
1298
1300
1302
1304
1306
1308
1310
1312
1314
1316
1318
1320
1322
1324
1326
1328
1330
1332
1334
1336
1338
1340
1342
1344
1346
1348
1350
1352
1354
1356
1358
1360
1362
1364
1366
1368
1370
1372
1374
1376
1378
1380
1382
1384
1386
1388
1390
1392
1394
1396
1398
1400
1402
1404
1406
1408
1410
1412
1414
1416
1418
1420
1422
1424
1426
1428
1430
1432
1434
1436
1438
1440
1442
1444
1446
1448
1450
1452
1454
1456
1458
1460
1462
1464
1466
1468
1470
1472
1474
1476
1478
1480
1482
1484
1486
1488
1490
1492
1494
1496
1498
1500
1502
1504
1506
1508
1510
1512
1514
1516
1518
1520
1522
1524
1526
1528
1530
1532
1534
1536
1538
1540
1542
1544
1546
1548
1550
1552
1554
1556
1558
1560
1562
1564
1566
1568
1570
1572
1574
1576
1578
1580
1582
1584
1586
1588
1590
1592
1594
1596
1598
1600
1602
1604
1606
1608
1610
1612
1614
1616
1618
1620
1622
1624
1626
1628
1630
1632
1634
1636
1638
164
```

```

00000000  FUNCTION LIST 2  (USES OUTPUT BUFFER 2)
00000004  CALL REPORT(348,MPBFLL(2))
00000008  CALL REPORT(358,VCLT(MBR2,39,MBI2,8))
0000000C  CALL REPORT(368,DVCM(MOR2,INTV2,MBR1,MCS))
00000010  CALL REPORT(378,VSMAL(MBR4,1,MBR6,8))
00000014  CALL REPORT(388,VSMAL(MBR3,1,MBR5,8))
00000018  CALL REPORT(398,VCLIP(MOR2,SP2847,MOR2,SN2848))
0000001C  CALL REPORT(408,VCLT(MOR2,38,MOR2,11))
00000020  CALL REPORT(418,MP8FO(MBI2,NHR1,2,1))
00000024  CALL REPORT(428,MP8FI(MOI2,NHR2,2,1))
00000028  CALL REPORT(438,MPEFL(2))
0000002C  FUNCTION LIST 3  (MASTER FUNCTION LIST)

```

```

0108      FUNCTION LIST 3 (MASTER FUNCTION LIST)
C -----
0109      CALL REPORT(448,MPBFL(3))
0100      CALL REPORT(458,MPXFL(1))
0101      CALL REPORT(458,MPXFL(2))
0102      CALL REPORT(458,MPXFL(3))
0103      CALL REPORT(478,MPPEFL(3))
0104      END OF FUNCTION LISTS IN MAP
C -----

```

C ---- PREPARING FOR LOOP IN HOST

```

C ----- INITIALIZE VARIOUS POSITION FLAGS. INITIALIZE INPUT BUFFER
C BUFFER IN MAP. TURN OVER OUTPUT BUFFER TO MAP
0100 READ(NIN,1555)IA(I,IAI),IAI=1,NPACK1)
0110 CALL REPORT(555,MPVDB(MB13,AI(1),2,1,AI(NPACK1)))
0110 CALL REPORT(515,VFLT(MB13,39,MB13,5))

```

```

0111 READ(NIN,10000)(A1(JA1),JA1=1,NPACK1)
0112 CALL REPORT(520,MPVDB(MB12,A1(1),2,1,A1(NPACK1)))
0113 CALL REPORT(530,MPFBA(NHR2))

```

```
0114      IOUT=8  
0115      CALL REPORT(549,MPVHL(SFLAG,2,3))  
          C-----  
          MOST LOOP  
0116      488 CONTINUE  
          C-----  
          INPUT TO MAP
```

```

0117 C CALL REPORT(555,MPVBA(NHR1))
0118 READ(MIN,1000,ERR=455)(A1(A1),IA1=1,NPACK1)
0119 GO TO 555
C -----
0120 455 RUN OUT OF INPUT, FILL IN WITH ZEROS
CONTINUE
0121 DO 465 JAI=IA1,NPACK1
0122 A1(JAI)=0
0123 CONTINUE
C -----
0124 465 IF NOT RUN OUT, CARRY ON WITH SAMPLES READ IN
CONTINUE
0125 CALL REPORT(565,MPFBA(NHR1))
OUTPUT FROM MAP
C -----
0126 CALL REPORT(575,MPVBA(NHR2))
0127 IOUT=IOUT+NPACK2
0128 IF(IOUT.GE.NLEN2) GO TO 555
0129 WRITE(ROUT,1000)(A2(IA2),IA2=1,NPACK2)
0130 CALL REPORT(595,MPFBA(NHR2))
0131 GO TO 455
C ----- END OF HOST LOOP
C -----
C ----- IF YOU GOT ALL OUTPUT SAMPLES, STOP MAP LOOP AND RETURN
C -----
0132 555 CONTINUE
C -----
0133 ILST2=NPACK2+NLEN2-IOUT
0134 WRITE(ROUT,1000)(A2(IA2),IA2=1,ILST2)
C -----
0135 FIGURE OUT TIME TAKEN, SAY OPERATION COMPLETE & QUIT
CLCK=SECONDS(CLCK)
TYPE '...'
TYPE '...' TIME FOR RESAMPLING =',CLCK,' SECONDS'
TYPE '...'
TYPE '...' RESAMPLING OPERATION COMPLETE.'
C -----
0136 CLOSE MAP
CALL REPORT(615,MPCLS(B))
RETURN
C -----
C ----- AND THEN THERE'S THE FORMAT STATEMENTS
0142 1000 FORMAT(16I5)
0143 END

```

PORTMAN IV-PLUS V82-S1 16:17:53 15-APR-88
 MAPRESAM1.FTH /14/TR:BLOCKS/VR

PAGE 8

```

C ----- SUBROUTINE TO REPORT MAP ERRORS
C
0001 SUBROUTINE REPORT(I,J)
0002 INTEGER*2 I,J
0003 IF(J.EQ.0) RETURN
0004 TYPE *, 'MAP ERROR', J, ' AT', I
0005 STOP
0006 END

```

```

*****
*
* SUBROUTINE COEFF(H)
*
*****

                ARVIND S ARORA

SUBROUTINE DESCRIPTION:

THIS SUBROUTINE COMPUTES COEFFICIENTS FOR A LOW PASS
FILTERING OPERATION. IT COMPUTES THE COEFFICIENTS SAMPLED AT
96 KSAM/SEC ONCE, AND THESE CAN BE USED FOR THE ENTIRE
RESAMPLING OPERATION. THIS SPEEDS UP THE OPERATION BY
ALMOST ONE ORDER OF MAGNITUDE. HOWEVER, IT RESTRICTS THE
INPUT AND OUTPUT FREQUENCIES TO INTEGER SUB-MULTIPLES OF
96 K. E.G. 16000, 12000, 8000, 6400.
THE FOLLOWING INPUTS ARE REQUIRED:
1) H FILTER COEFF ARRAY CENTRE COEFF AT H(1)
2) IF1 SYMMETRIC, COEFFS FOR ONLY +VE TIME
3) IF2 CORNER FREQ FOR THE FILTER
4) IRATE1 (IGNORED)
5) ITRUN SAMPLING RATE OF THE INPUT FILE
    FILTER TRUNCATION NO. (IGNORED)

THIS FILTER USES THE IMPULSE RESPONSE (NON CAUSAL) OF A LOW-
PASS FILTER TRUNCATED USING A 1ST ORDER HAMMING WINDOW WITH THE
ONE COEFFICIENT ALFA=.7.

CALLS TO SUBROUTINES:

1) FUNCTION-SUBROUTINE SINCE(ANGLE)

*****
SUBROUTINE COEFF(H)
DIMENSION H(601)
COMMON /COFF/IF1,IF2,ITRUN,IRATE1
DATA ALFA/.7/
C----- COMMENCE COMPUTATION
T=1./96000.
PI=2.*ASIN(1.)
HCEN=2*IF1*PI
TSPAN=12000.*HCEN
H(1)=2.*FLOAT(IF1)/FLOAT(ITRATE1)
DO 150 I=1,600
ARG=HCEN*PI*FLOAT(I)
H(I+1)=H(1)*SINCE(ARG)=(ALFA+(1.-ALFA)*COS(ARG/TSPAN))
CONTINUE

```

858

PAGE 2

15-APR-88

16:10:34

FORTRAN IV-PLUS V02-51
GCOEFF.PTN /14/TR:BLOCKS/WR0014 RETURN
0015 END

[illegible]

APPENDIX F CVSD ALGORITHM

In this appendix, the CVSD algorithm will be discussed. Before the design of this real-time CVSD system can be presented, it is essential to understand the structure of CVSD systems.

F.1 The CVSD System

In this section, the structure of a CVSD system is presented. The system described here is identical to that of [1]. A schematic of an encoder and decoder pair for CVSD is shown in Figure F.1.

A CVSD encoder is simply a delta modulator with an adaptive stepsize. In order to minimize the difference between adjacent samples, the input signal $s(k)$ is normally oversampled. Typically, the sampling rate is 16 k samples per second. Since one bit per sample is transmitted, the transmission rate is 16 K bits per second.

The encoder quantizes the difference $e(k)$ between an input speech sample and its estimate which is predicted by a first order predictor.

$$e(k) = s(k) - \alpha * \hat{s}(k-1)$$

where $\hat{s}(k-1)$ is the reconstructed speech of the input speech sample at time $k-1$. Then the encoder outputs a single bit, $b(k)$, for each corresponding input sample where

$$b(k) = \begin{cases} 1 & \text{if } e(k) \geq 0 \\ 0 & \text{if } e(k) < 0 \end{cases}$$

The adaptive stepsize logic derives its output $\Delta(k)$ from the bit stream $b(k)$ and an appropriate initial state,

$$\Delta(k) = \beta * \Delta(k-1) + g(k)$$

where

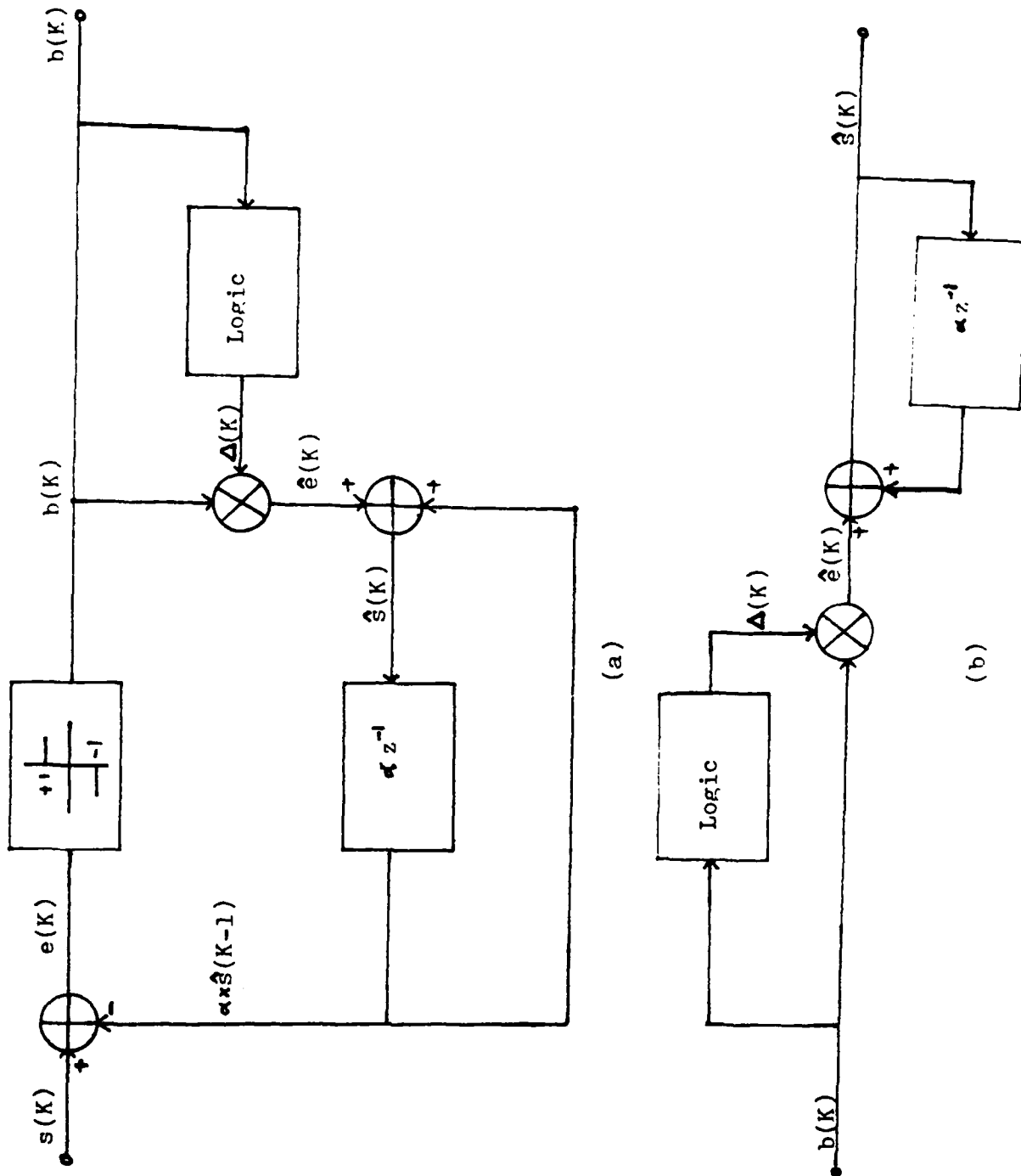


Fig. F.1 The CVSD System (a) Encoder (b) Decoder

$$g(k) = \begin{cases} \Delta_{\max} * (1 - \beta) & \text{If } b(k) = b(k-1) = b(k-2) \\ \Delta_{\min} * (1 - \beta) & \text{otherwise} \end{cases}$$

For a long run in which no three consecutive bits are identical, the stepsize approaches $\Delta_{\min}$. On the other hand, if the many consecutive bits are the same, the stepsize will approach $\Delta_{\max}$.

The estimate of the difference signal is given by

$$\hat{e}(k) = b(k) * \Delta(k)$$

Then the reconstructed speech is computed from

$$\hat{s}(k) = \alpha * \hat{s}(k-1) + \hat{e}(k)$$

Because the reconstructed speech in the transmitter is determined solely by the bit stream $b(k)$ and an agreed initial state, the receiver can produce the same reconstructed speech samples if no channel error occur.

Normally, there is no best criteria to judge the performance of a speech algorithm. However, the basic signal to noise ratio criteria is used here.

$$SNR = 10 * \log_{10} \frac{\sum_{k=1}^M s(k)^2}{\sum_{k=1}^M (s(k) - \hat{s}(k))^2}$$

F.2 The Real-Time CVSD System

In this section, the design of the real-time CVSD system is presented. The major design decisions and real-time algorithm are discussed. The program listings are followed.

F.2.1 Design Overview

In order to design a real-time speech system, several points have to be considered. First, the flexibility of changing system parameters such as

speech algorithm parameters, buffer sizes and buffer starting locations must be provided. This allows system performance to be studied without any further modification of the real-time program. However, this approach will increase overhead time. Therefore, a "pre-bound" concept will be introduced. Second, the arithmetic execution time has to be minimized. This can be done by efficiently utilizing the arithmetic processors. Therefore, the "hiding" technique which hides additions inside the period of multiplication will be employed. Third, in order to decrease overhead time, all processors have to be utilized as asynchronously and as simultaneously as possible. However, the necessary synchronization between the processors has to be considered.

The real-time CVSD implementation employs five MAP processors to do speech processing asynchronously and simultaneously. The whole algorithm is divided into two steps: An initialization step and a processing step. Each processor has its own role in each step.

1. The Initialization Step

In this step, the host computer will initialize the MAP-300 system, set up the appropriate command sequence, and initiate the real-time CVSD algorithm. Each processor in the MAP-300 has to do the following functions:

- A. The CSPU acts as the resource controller. Responding to a host command, it loads a data acquisition program into the ADAM, loads digital-to analog conversion program into the AOM, configures the buffers, loads an arithmetic function into the AP which will reset the contents of the buffers, builds a recurrent sequence of commands called a function list and prebinds the APS module of the CVSD program.

- B. The AP executes the arithmetic function which is loaded by the CSPU. The purpose of this execution is to reset the buffers.

2. The Processing Step

After initialization, the MAP-300 can execute its functions without any further commands from host computer. The role of each processor are described as follows:

- A. The CSPU acts as the resource controller. Responding to each interrupt, it updates the contents of the corresponding processor control block. Responding to the function list, it checks the availability of the appropriate buffer and initiates AP operation by loading the CVSD program from MAP memory. The synchronization is also done by CSPU which is described in the next section.
- B. The APU executes the real-time CVSD algorithm.
- C. The APS produces data addresses and acts as the controller of the real-time CVSD system.
- D. The ADAM samples the input speech signal from telephone module and stores them into MAP memory.
- E. AOM converts input digital samples into an analog signal and output it to telephone module.

In the next two subsections, the detailed description of these two steps will be presented.

F.2.2 The Initialization Step

The initialization step, which is controlled by host computer, includes reading system parameters, giving commands to the CSPU and building a function list. The algorithm is shown in Fig. F.2.

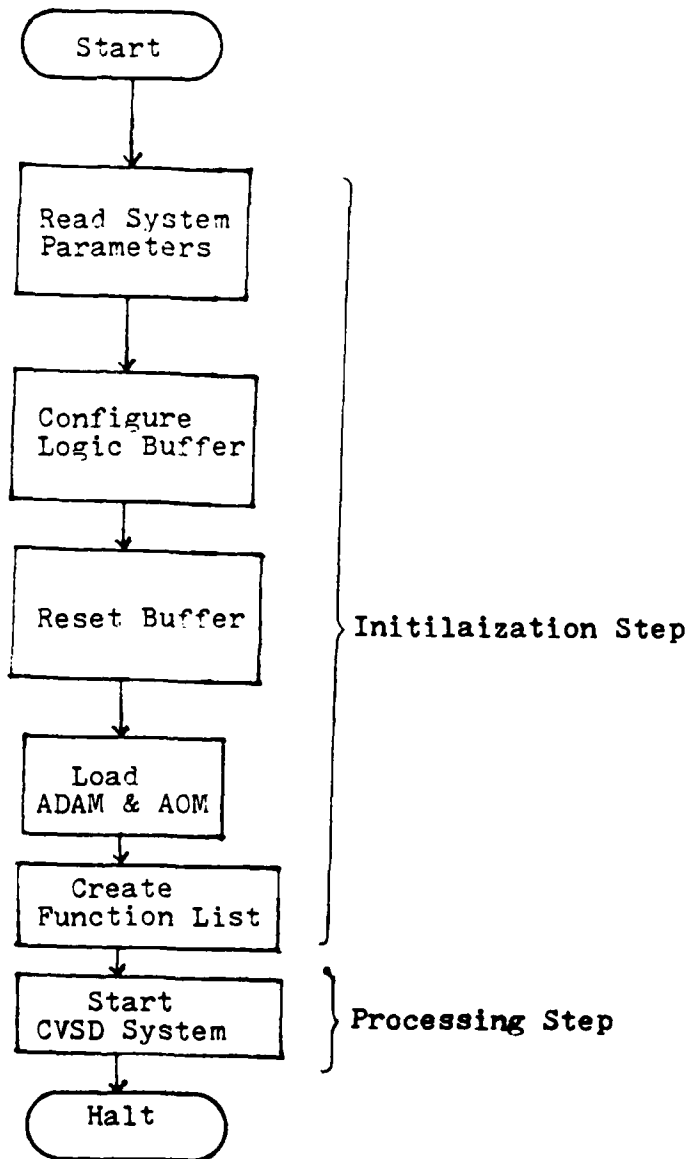


Fig. F.2 Algorithm for Initialization Step

AD-A086 134

NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(11)
APR 80 J L NELSA, D L COHN, A ARORA

F/6 17/2

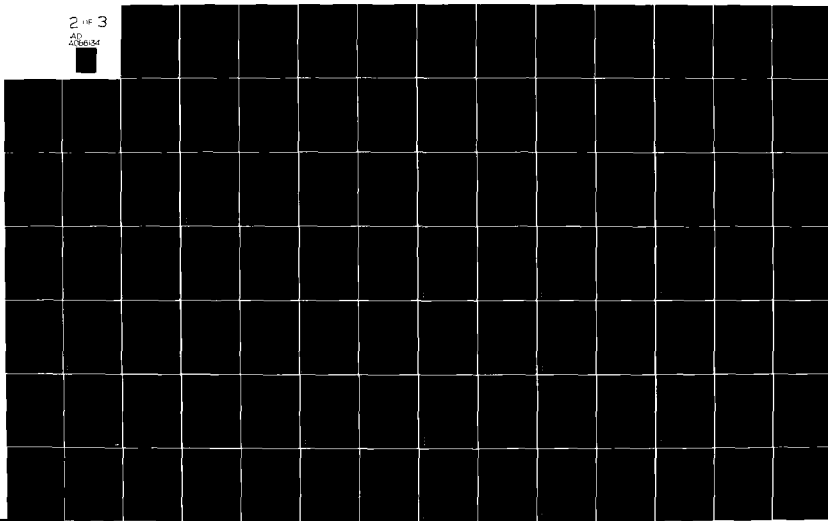
DCA100-79-C-0005

NL

UNCLASSIFIED

2 OF 3

AD
A086134



There are two points worth to note as follows:

1. Because of parallel processing in the MAP-300, the synchronization between the AP and the AOM, and between the AP and the ADAM has to be considered. Memory is the common place which every processor has to access. Therefore, the synchronization is done by checking buffer availability as follows:
 - A. If the first ADAM buffer is busy, CSPU goes to wait. Otherwise, goes to Step B.
 - B. CSPU initiates AP. AP starts to read input samples from the first ADAM buffer, execute CVSD function and output results into the first AOM buffer.
 - C. If the second ADAM buffer is busy, CSPU goes to wait, otherwise, goes to Step D.
 - D. CSPU initiates AP. AP does the same function as Step B except it reads samples from the second ADAM buffer and outputs results to the second AOM buffer.
 - E. Go to Step A.This structure is included in a function list.
2. In order to increase the flexibility of changing system parameters, there are some substitutive attributes inside the APS module. In a normal operation, CSPU gets values for those attributes from a host command and then binds them with the APS module before loading it into the APS memory. In order to decrease this overhead time, a "pre-bound" has to be done prior to real-time execution. The prebinding process will create a new APS module from old APS module by permanently binding the attributes. At the execution time, all normal operations of binding and buffer protection will be bypassed. Thus, the overhead time is decreased.

F.2.3 The Processing Step

In the processing step, each of three processors has a program. Among them, the data acquisition and data output programs, which are included in the Simple Notation for Array Processing library (SNAP library), are fairly simple. The actual CVSD algorithm is executed by the AP processor.

The real-time CVSD program can be divided into two parts as follows:

1. APU module of CVSD algorithm which is shown in Fig. F.3 will do the whole arithmetic operation described in the Section F.1. The input and output data are directed by APS module. The "hiding" technique is employed to reduced execution time.
2. The APS module of the CVSD algorithm which is shown in Fig. F.4 will produce address sequences of input and output data for APU module. Also, APS module acts as the controller of the real-time CVSD system. This AP control protocol is shown in Fig. F.5 and described as follows:
 - * CPU starts APS-input by setting RI
 - * APS sets the program counter for write routine and starts APS-output by setting RO.
 - * APS starts APU by setting RA.
 - * After producing addresses for input data, APS-input turns itself off by clearing RI.
 - * When FI is cleared, APU leaves the loop.
 - * After producing addresses for output data, APS-output turns itself off by clearing RO.
 - * APU turns itself off by clearing RA.
 - * CSPU is interrupted by both RAI (RA off) and DNI (AP done).

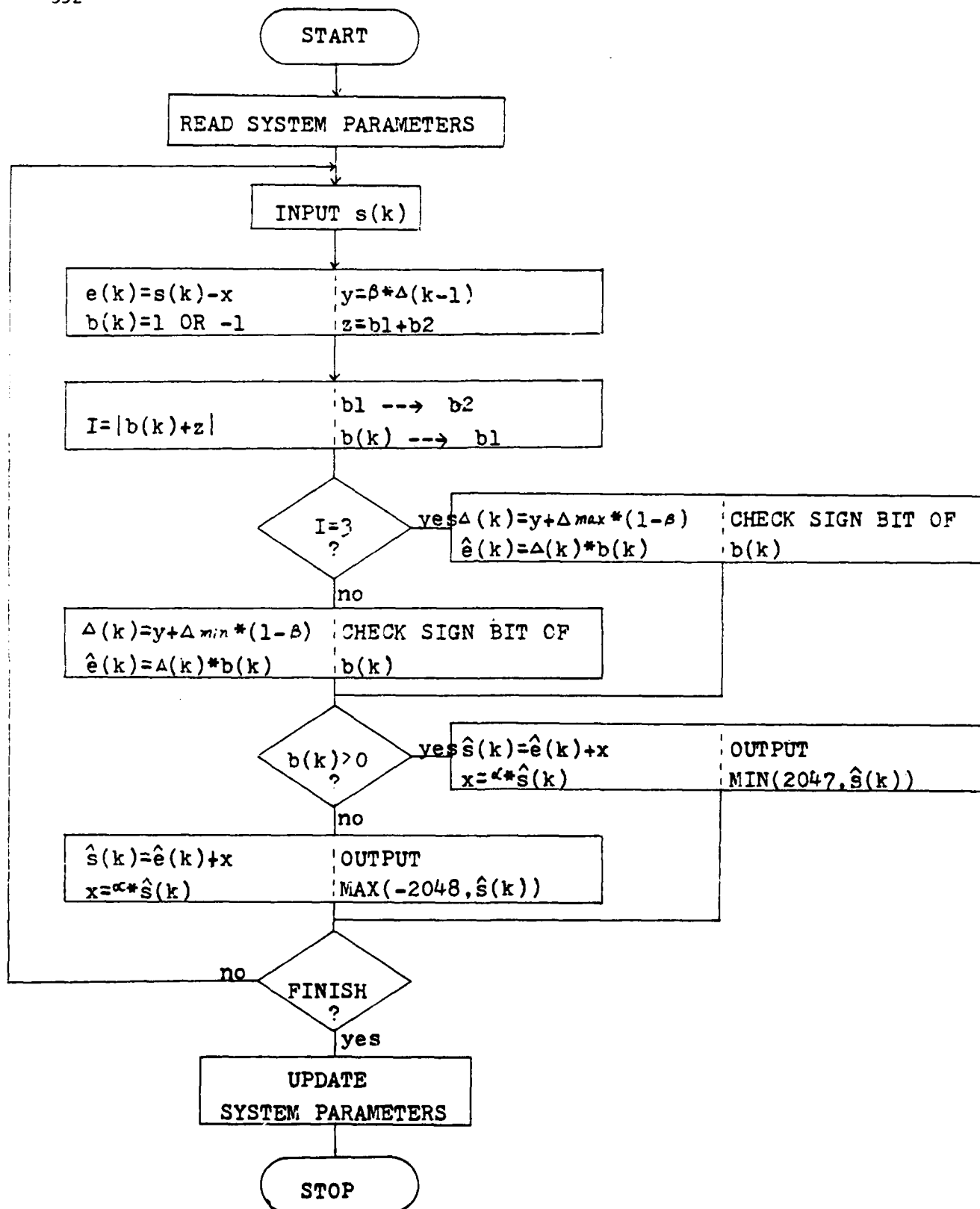
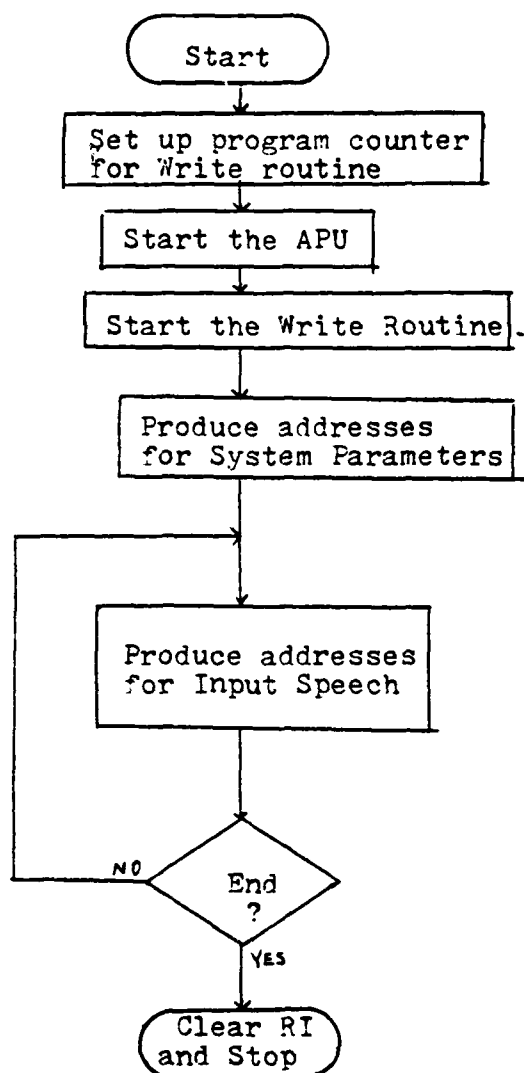
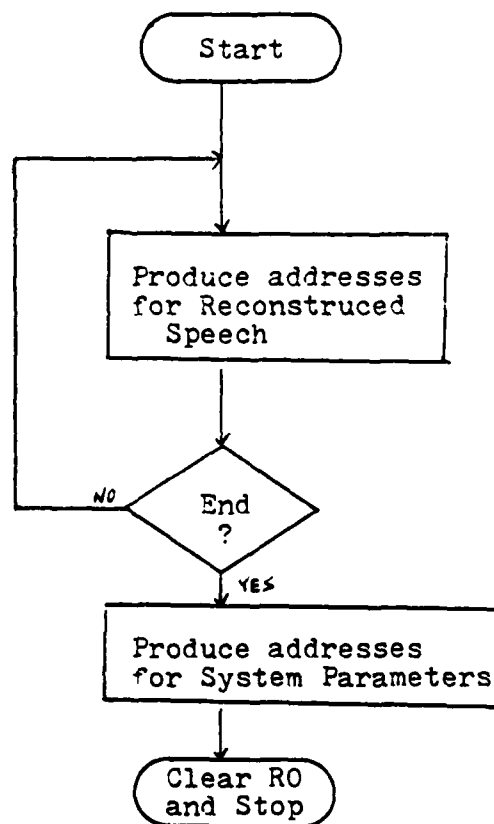


Fig. F.3 APU Flow Chart



(a)



(b)

Fig. F.4 APS Flow Chart

(a) APS-Input

(b) APS-Output

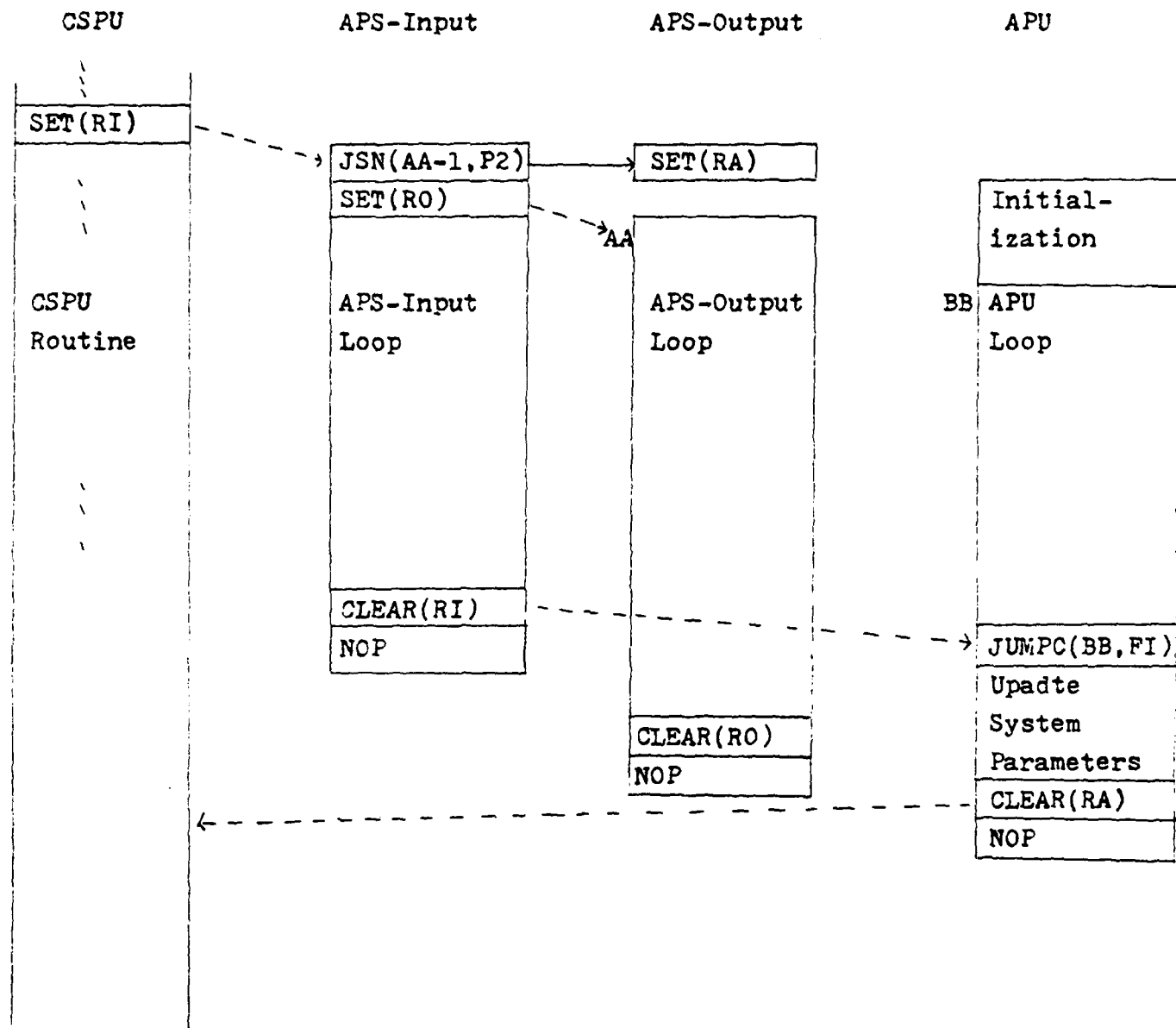


Fig. F.5 The AP Control Protocol

F.3 Conclusion

The execution time for the real-time CVSD algorithm is approximately 12 micro-seconds per sample. So, this implementation can be used with a sampling rate up to 83 KHz in the real-time mode. A set of parameters which gives a very good quality of speech is as follows:

$$\Delta_{\max} = 750$$

$$\Delta_{\min} = 25$$

$$\alpha = 0.94$$

$$\beta = 0.99$$

A two seconds sample speech is executed by this algorithm, and a 11.87 db signal-to-noise ratio performance is obtained.

F.4 Reference

- [1] R. W. Schafer et. al., "Tandem Interconnection of LPC and CVSD Digital Speech Coders" Final Report to Defense Communications Agency, DCA Contract No. DCA 100-76-C-0073, November, 1977.

```

PAGE 1: MAP MODULE OF C V S D PROGRAM --- VERSION 1 --- FEB. 19, 1988

(00001) *
(00002) *
(00003) *
(00004) *
(00005) *
(00006) *
(00007) *
(00008) *
(00009) *
(00010) *
(00011) *
(00012) *
(00013) *
(00014) *
(00015) *
(00016) *
(00017) *
(00018) *
(00019) *
(00020) *
(00021) *
(00022) *
(00023) *
(00024) *
(00025) *
(00026) *
(00027) *
(00028) *
(00029) *
(00030) *
(00031) *
(00032) *
(00033) *
(00034) *
(00035) *
(00036) *
(00037) *
(00038) *
(00039) *
(00040) *
(00041) *
(00042) *
(00043) *
(00044) *

MAP MODULE OF C V S D PROGRAM --- VERSION 1 --- FEB. 19, 1988
*****
* REAL-TIME CVSD TRANSMITTER *
*****

DATA: 2/19/88
MAVLIN YEH

PROGRAM NAME : CVSD.ND1

PROGRAM DESCRIPTION:

THIS MAP-FUNCTION SIMULATES AN ADAPTIVE DELTA-MODULATOR WHICH
USES A FIRST ORDER PREDICTOR WITH THE COEFFICIENT ALPHA AND
A FIRST ORDER ADAPTOR WITH THE COEFFICIENT BETA. THE VALUES OF
ALPHA AND BETA ARE TRANSFERRED FROM HOST PROGRAM TO MAP SCALAR
TABLE.

PROCEDURE OF FUNCTIONS:

1.READ IN SPEECH --- S(K)
2.CALCULATE THE PREDICTING ERROR --- E(K)=S(K)-P(K)
3.PASS E(K) THROUGH DELTA MODULATOR AND PRODUCE L(K)
4.GET THE LOGIC OUTPUT G(K)
5.CALCULATE ADAPTIVE STEP SIZE --- D(K)=BETA*D(K-1)+G(K)
6.CALCULATE RECONSTRUCTED PREDICTING ERROR --- EHAT(K)=L(K)*D(K)
7.CALCULATE RECONSTRUCTED SPEECH --- SHAT(K)=SHAT(K-1)+EHAT(K)
8.UPDATE PREDICTOR'S PARAMETER = ALPHA=SHAT(K)

THIS PROGRAM PROVIDES UPPER LIMIT AND LOWER LIMIT OF OUTPUT
DATA. THESE TWO LIMITS ARE TRANSFERRED FROM HOST PROGRAM.
THIS PROGRAM CONTAINS TWO SUB-PROGRAMS WHICH ARE CVSD FOR APU
AND V2260D FOR APS. PASSING THROUGH MAP-SIMULATOR, CVSD TAKES
2.669 MICROSECONDS. (LOOP TIME)

CALL TO THIS PROGRAM:

CALL CVSD(Y,A,U)
OR
MAP=CVSD(Y,A,U)

WHERE V=OUTPUT BUFFER FROM MAP

```

U-INPUT BUFFER TO MAP
A-STARTING LOCATION OF THE FOLLOWING PARAMETERS
1ST PARAMETER = UPPER LIMIT OF OUTPUT DATA
2ND PARAMETER = LOWER LIMIT OF OUTPUT DATA
3RD PARAMETER = LOWER LOGIC OUTPUT
4TH PARAMETER = HIGHER LOGIC OUTPUT
5TH PARAMETER = BETA
6TH PARAMETER = THRESHOLD FOR LOGIC
7TH PARAMETER = DELTA
8TH PARAMETER = ALPHA

CONTAINS FOR ASSEMBLY

```

MSS=0
DMYS=5794
BUS1=$001
TOESPT=$2B8
CSPUSNOS=$21FC
AFDTS=$8E8
DUMMY OUTPUT SPACE
BUS 1
TOP OF EXEC
LOCATION OF NO-CSPU SUPPORT
STARTING ADDRESS OF AFDT

```

DISPATCH TABLE ENTRY

FCB=245
 $\phi L = AFDTS + 3 * 2 * (FCB - 128)$
 FCB=245
 STARTING LOCATION OF AFDT FOR 245

ADDR	DATA
ADDR	CVSDS(R7,1)
ADDR	V226EDS(R7,1)
ADDR	CSPUSNOS(.1,B)

APU & APS MODULES

STARTING LOCATION OF CVSD AND V2260D

APU MODULE

FUNCTIONS:

```

***** ADM1 *****
READ SCALARS
CALCULATE E(K)
CALCULATE DI(K)
CALCULATE ENAT(K) & SHAT(K)
UPDATE PREDICTOR'S PARAMETERS
BOUND THE OUTPUT DATA

***** ADM2 *****
READ SCALARS
LOGIC OPERATION
UPDATE LOGIC PARAMETERS
NO OPERATION
BOUND THE OUTPUT DATA

```



```

0019. 1000
MAP MODULE OF C.V.S. D PROGRAM
VERSION 1
A19 04834 43882268 (00133)
A1A 04836 88888888 (00134)
A1B 04838 84288288 (00135)
A1C 0483A 88988888 (00136)
A1D 0483C 88888888 (00137)
A1E 0483E 8881F828 (00138)
A1F 04840 88858888 (00139)
A20 04842 41A88888 (00140)
A21 04844 88988888 (00141)
A22 04846 888E8851 (00142)
A23 04848 85C86728 (00143)
A24 0484A 8888889C (00144)
A25 0484C 88818888 (00145)
A26 0484E 8810888C (00146)
A27 04850 18888838 (00147)
A28 04852 88858888 (00148)
A29 04854 41A88888 (00149)
A2A 04856 88988888 (00150)
A2B 04858 888E8851 (00151)
A2C 0485A 85C86728 (00152)
A2D 0485C 8888889C (00153)
A2E 0485E 88818888 (00154)
A2F 04860 8810888C (00155)
A30 04862 82288468 (00156)
A31 04864 889C8288 (00157)
A32 04866 8888827C (00158)
A33 04868 8888889C (00159)
A34 0486A 8888888C (00160)
A35 0486C 28322832 (00161)
A36 0486E 88888888 (00162)
A37 04870 18888888 (00163)
A38 04872 46888268 (00164)
A39 04874 88888888 (00165)
A3A 04876 18888818 (00166)
A3B 04878 88888888 (00167)
A3C 0487A 88888888 (00168)
A3D 0487C 88888888 (00169)
A3E 0487E 88888888 (00170)
A3F 04880 88888888 (00171)
A40 04882 88888888 (00172)
A41 04884 88888888 (00173)
A42 04886 88888888 (00174)
A43 04888 88888888 (00175)
A44 0488A 88888888 (00176)
A45 0488C 88888888 (00177)
A46 0488E 88888888 (00178)
A47 04890 88888888 (00179)
A48 04892 88888888 (00180)
A49 04894 88888888 (00181)
A4A 04896 88888888 (00182)
A4B 04898 88888888 (00183)
A4C 0489A 88888888 (00184)
A4D 0489C 88888888 (00185)
A4E 0489E 88888888 (00186)
A4F 048A0 88888888 (00187)
A4G 048A2 88888888 (00188)
A4H 048A4 88888888 (00189)
A4I 048A6 88888888 (00190)
A4J 048A8 88888888 (00191)
A4K 048AA 88888888 (00192)
A4L 048AC 88888888 (00193)
A4M 048AE 88888888 (00194)
A4N 048B0 88888888 (00195)
A4O 048B2 88888888 (00196)
A4P 048B4 88888888 (00197)
A4Q 048B6 88888888 (00198)
A4R 048B8 88888888 (00199)
A4S 048BA 88888888 (00200)
A4T 048BC 88888888 (00201)
A4U 048BE 88888888 (00202)
A4V 048B0 88888888 (00203)
A4W 048B2 88888888 (00204)
A4X 048B4 88888888 (00205)
A4Y 048B6 88888888 (00206)
A4Z 048B8 88888888 (00207)
A50 048BA 88888888 (00208)
A51 048BC 88888888 (00209)
A52 048BE 88888888 (00210)
A53 048B0 88888888 (00211)
A54 048B2 88888888 (00212)
A55 048B4 88888888 (00213)
A56 048B6 88888888 (00214)
A57 048B8 88888888 (00215)
A58 048BA 88888888 (00216)
A59 048BC 88888888 (00217)
A5A 048BE 88888888 (00218)
A5B 048B0 88888888 (00219)
A5C 048B2 88888888 (00220)
A5D 048B4 88888888 (00221)
A5E 048B6 88888888 (00222)
A5F 048B8 88888888 (00223)
A60 048BA 88888888 (00224)
A61 048BC 88888888 (00225)
A62 048BE 88888888 (00226)
A63 048B0 88888888 (00227)
A64 048B2 88888888 (00228)
A65 048B4 88888888 (00229)
A66 048B6 88888888 (00230)
A67 048B8 88888888 (00231)
A68 048BA 88888888 (00232)
A69 048BC 88888888 (00233)
A6A 048BE 88888888 (00234)
A6B 048B0 88888888 (00235)
A6C 048B2 88888888 (00236)
A6D 048B4 88888888 (00237)
A6E 048B6 88888888 (00238)
A6F 048B8 88888888 (00239)
A6G 048BA 88888888 (00240)
A6H 048BC 88888888 (00241)
A6I 048BE 88888888 (00242)
A6J 048B0 88888888 (00243)
A6K 048B2 88888888 (00244)
A6L 048B4 88888888 (00245)
A6M 048B6 88888888 (00246)
A6N 048B8 88888888 (00247)
A6O 048BA 88888888 (00248)
A6P 048BC 88888888 (00249)
A6Q 048BE 88888888 (00250)
A6R 048B0 88888888 (00251)
A6S 048B2 88888888 (00252)
A6T 048B4 88888888 (00253)
A6U 048B6 88888888 (00254)
A6V 048B8 88888888 (00255)
A6W 048BA 88888888 (00256)
A6X 048BC 88888888 (00257)
A6Y 048BE 88888888 (00258)
A6Z 048B0 88888888 (00259)
A70 048B2 88888888 (00260)
A71 048B4 88888888 (00261)
A72 048B6 88888888 (00262)
A73 048B8 88888888 (00263)
A74 048BA 88888888 (00264)
A75 048BC 88888888 (00265)
A76 048BE 88888888 (00266)
A77 048B0 88888888 (00267)
A78 048B2 88888888 (00268)
A79 048B4 88888888 (00269)
A7A 048B6 88888888 (00270)
A7B 048B8 88888888 (00271)
A7C 048BA 88888888 (00272)
A7D 048BC 88888888 (00273)
A7E 048BE 88888888 (00274)
A7F 048B0 88888888 (00275)
A7G 048B2 88888888 (00276)
A7H 048B4 88888888 (00277)
A7I 048B6 88888888 (00278)
A7J 048B8 88888888 (00279)
A7K 048BA 88888888 (00280)
A7L 048BC 88888888 (00281)
A7M 048BE 88888888 (00282)
A7N 048B0 88888888 (00283)
A7O 048B2 88888888 (00284)
A7P 048B4 88888888 (00285)
A7Q 048B6 88888888 (00286)
A7R 048B8 88888888 (00287)
A7S 048BA 88888888 (00288)
A7T 048BC 88888888 (00289)
A7U 048BE 88888888 (00290)
```

PAGE 5: MAP MODULE OF C V S D PROGRAM --- VERSION 1 --- FEB. 19, 1988

84878 88888838 (88177)
(88178)
(88179)

CVSDSSZ=8A-CVSDSSA
END
EJECT

; COMPUTE THE SIZE OF THE MODULE

PAGE 51

MAP MODULE OF C.V.S.D. PROGRAM - VERSION 19.0000

APS3-V2260D

```

(00180) *
(00181) *
(00182) *
(00183) *
(00184) *
(00185) *
(00186) *
(00187) *
(00188) *
(00189) *
(00190) *
(00191) *
(00192) *
(00193) *
(00194) *
(00195) *
(00196) *
(00197) *
(00198) *
(00199) *
(00200) *
(00201) *
(00202) *
(00203) *
(00204) *
(00205) *
(00206) *
(00207) *
(00208) *
(00209) *
(00210) *
(00211) *
(00212) *
(00213) *
(00214) *
(00215) *
(00216) *
(00217) *
(00218) *
(00219) *
(00220) *
(00221) *
(00222) *
(00223) *

```

```

04878 000048C2
0487A 000048B4
0487C 0001
0487D 004A
0487E 000048B2

```

```

V2260DS BEGIN APS(V2260D) ; BEGIN OF THIS MODULE

```

```

JSN(V2260DS-1,P2) ; SET APS-OUTPUT PC P2
SET(RO) ; RUN APS-OUTPUT

```

SCALAR ADDRESSES

```

LOAD(BR0,MSS(1),L,TF) ; UPPER LIMIT
ADD(BR0,2,TF) ; LOWER LIMIT
ADD(BR0,2,TF) ; LOWER LOGIC OUTPUT
ADD(BR0,2,TF) ; HIGHER LOGIC OUTPUT
ADD(BR0,2,TF) ; BETA
ADD(BR0,2,TF) ; THRESHOLD
ADD(BR0,2,TF) ; ALPHA
ADD(BR0,2,TF) ; DELTA
ADD(BR0,2,TF) ; L(K-1)
ADD(BR0,2,TF) ; L(K-2)
ADD(BR0,2,TF) ; ALPHA=SHAT(K-1)

```

1ST INPUT PROGRAM

```

LOAD(BR3,[1]) ; SET STARTING ADDR.
LOAD(BR1,MSS) ; COUNTER--- SAMPLE SIZE - 1
SUB(BR3,MSS) ; BASE ADDRESS - SPACING
ADD(BR3,[9],TF) ; GIVE ADDR.
SUBL(BR1,1),JUMPP(V2260DS3) ; CHECK END ?

```

THIS IS APS PROGRAM FOR CVSD.
IT EMPLOYES TWO INPUT-BUFFERS AND TWO OUTPUT-BUFFERS.
ALSO ONE CAN REPEAT THIS PROGRAM WITHOUT LOADING IT AGAIN.

```

EVEN ; START ON WORD BOUNDARY
ADDR ; PTR TO CONSTR INSTR BLOCK
V2260DS+2*V2260DS ; STARTING ADDR. OF SCALAR
DATA 1 ; ONE SCALAR
V2260DSZ ; SIZE
ADDR ; CHAIN ANCHOR
V2260DSA
EVEN

```

```

A00 04800 00201540
A01 04802 02300030

```

```

A02 04804 04C20000
A03 04806 068A0002
A04 04808 088A0002
A05 0480A 0A8A0002
A06 0480C 0C8A0002
A07 0480E 0E8A0002
A08 04810 108A0002
A09 04812 128A0002
A0A 04814 148A0002
A0B 04816 168A0002
A0C 04818 188A0002

```

```

A0D 0481A 1A701000
A0E 0481C 1C500000
A0F 0481E 1E320000
A10 04820 20BA9000
A11 04822 22191001

```

[illegible]

PAGE 8: MAP MODULE OF C.V.S. D. PROGRAM VERSION 1.19.1999

```

AFDTS: 000E8 (00064) (00069)
BUSIS: 00091 (00061) (00259)
CSPUSMOS: 021FC (00063) (00073)
CVSDS: 04002 (00071) (00094)
CVSDSSA: 00000 (00091) (00102) (00177)
CVSDSSZ: 00030 (00092) (00177)
DMYS: 00794 (00060)
FCB: 000F5 (00060) (00069)
LOOP: 0000C (00117) (00150) (00162)
LOOP1: 00030 (00131) (00173)
LOOP2: 00010 (00136) (00175)
LOOP3: 00030 (00151) (00164)
LOOP4: 00020 (00139) (00155)
MSS: 00000 (00059) (00205) (00220) (00221) (00232) (00233)
TOESPRT: 00200 (00257) (00259)
V22600S: 04000 (00062) (00250)
V22600S1: 00000 (00072) (00190) (00197) (00253)
V22600S3: 00010 (00222) (00223)
V22600S5: 00016 (00200) (00231)
V22600S6: 00019 (00234) (00235)
V22600SA: 04002 (00193) (00244)
V22600S1: 04002 (00189) (00251)
V22600S5: 00002 (00190) (00205)
V22600S2: 0004A (00192) (00253)

```

LINES WITH ERRORS: 0 (MAP VERSION 000101.10) E- 0

/TR:BLOCKS/VR

[illegible]

```

*****
*                                     *
*          CVSD HOST SUPPORT PROGRAM          *
*                                     *
*****

```

DATE: 11/14/79
MAVLIN YEH

CVSD HOST SUPPORT PROGRAM

CALL CVSD(Y,A,U)

OR

MAP=CVSD(Y,A,U)

WHERE: Y=OUTPUT BUFFER

A-STARTING LOCATION OF SCALARS

U-INPUT BUFFER

INTEGER FUNCTION CVSD(Y,A,U)

INTEGER Y, A, U, FCBN

CVSD=FCBGN(245,Y,A,U,B,B,B,B,5)

RETURN

END

1001 1002 1003 1004 1005

FORTRAN IV-PLUS V02-11
CVSDHS TR-CKS/

PAGE 2

28:01:27 28-MAR-88

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	25	RV,I,CON,LCL
2	SPDATA	6	RV,D,CON,LCL
3	SIDATA	12	RV,D,CON,LCL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
------	------	---------	------	------	---------	------	------	---------

CVSD	I*2	1-888888						
------	-----	----------	--	--	--	--	--	--

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
A	I*2	F-888884	U	I*2	F-888886	Y	I*2	F-888882

FUNCTIONS AND SUBROUTINES REFERENCED

FCBGN

TOTAL SPACE ALLOCATED = 888126 43

NO FPP INSTRUCTIONS GENERATED

CVSDHS,CVSDHS/-SP-CVSDHS

+CVSDBN.FTN

/TR:BLOCKS/VR

```

*****
C      C V S D
C      IN REAL TIME MODE
C
*****

```

PROGRAM: REALCVSD.FTN

DATE: 2-19-88
MAVLIN VEN

THIS PROGRAM DOES NOT HAVE THE SOURCE CODING ROUTINES AND CVSD-RECEIVER.

ITS FUNCTION IS DESCRIBED AS FOLLOWS:

A CVSD TRANSMITTER IS SIMPLY A DELTA MODULATOR WITH AN ADAPTIVE STEP SIZE. IN ORDER TO MINIMIZE THE DIFFERENCE BETWEEN ADJACENT SAMPLES, THE INPUT SIGNAL $S(K)$ IS NORMALLY OVERSAMPLED. TYPICALLY, THE SAMPLING RATE IS 16K BITS PER SECOND.

THE TRANSMITTER QUANTIZES THE DIFFERENCE $E(K)$ BETWEEN AN INPUT SPEECH SAMPLE AND ITS ESTIMATE WHICH IS PREDICTED BY A FIRST ORDER PREDICTOR.

$$E(K) = S(K) - \text{ALPHA} * \text{SHAT}(K-1)$$

WHERE $\text{SHAT}(K-1)$ IS THE RECONSTRUCTED SPEECH OF THE INPUT SPEECH SAMPLE AT TIME $K-1$.

THEN THE TRANSMITTER OUTPUTS A SINGLE BIT, $B(K)$, FOR EACH CORRESPONDING INPUT SAMPLE WHERE

$$B(K) = \begin{cases} 1 & \text{IF } E(K) > 0 \\ 0 & \text{IF } E(K) \leq 0 \end{cases}$$

$$B(K) = \begin{cases} 1 & \text{IF } E(K) > 0 \\ 0 & \text{IF } E(K) \leq 0 \end{cases}$$

THE ADAPTIVE STEP SIZE LOGIC DRIVES ITS OUTPUT $\Delta(K)$ FROM THE BIT STREAM $B(K)$ AND AN APPROPRIATE INITIAL STATE.

$$\Delta(K) = \text{BETA} * \Delta(K-1) + G(K)$$

WHERE

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

$$G(K) = \begin{cases} 1 & \text{DELTA-MAX} * (1 - \text{BETA}) \quad \text{IF } B(K) = B(K-1) \\ 0 & \text{OTHERWISE} \end{cases}$$

THE SET OF PARAMETERS WHICH CAN GET THE BEST RESULT FOR CHANNEL NOISE ARE AS FOLLOWS:

$$\begin{aligned} \text{ALPHA} &= 0.94 \\ \text{BETA} &= 0.99 \\ \text{DELTA-MAX} &= 75 \\ \text{DELTA-MIN} &= 25 \end{aligned}$$

THE WAY TO SET UP CVSD IN MAP-388 IS AS FOLLOWS:

1. TO PRODUCE A NEW SNAP-LIBRARY:

F4P CVSDHS-CVSDHS

LBR SNPLIB/IN-CVSDHS

2. TO PRODUCE A CVSD TASK FILE:

F4P REALCVSD-REALCVSD


```

FORTRAN IV-PLUS V02-B1      19:59:12      28-MAR-88      PAGE 3
CVSDBN.FTN      /TR:BLOCKS/VR

0031      SCAL(8)=SCAL(8)/16.**3
0032      TYPE = '...' TYPE IN SAMPLING FREQ.'
0033      READ(4,*)FREQ
0034      WRITE(6,*)' SAMPLING FREQUENCY =',FREQ
0035      CLOSE(UNIT=4)
0036      DEFINE BUS IDENTIFIERS
0037      IBUS1=64
0038      IBUS2=128
0039      IBUS3=192
0040
0041      C---
0042      C
0043      MAP CONFIGURATION BEGINS HERE
0044      M=MPOPM(1)
0045      DO 100 I=1,2
0046      SSIZE=(I-1)*SIZE
0047      M=MPCLB(IBUS3+BSIN(I),B,B+SSIZE,SIZE,REAL,CONTIG,SHORT)
0048      M=MPCLB(IBUS2+BSIN(I),B,B+SSIZE,SIZE,REAL,CONTIG,SHORT)
0049      INITIALIZE THESE BUFFERS
0050      M=VSHA1(BSIN(1),B,BSIN(1),B)
0051      M=VSHA1(BSHTR(1),B,BSHTR(1),B)
0052      CONTINUE
0053      M=MPCLB(IBUS1+IDAOM,38888,B,512,B,FIXED,CONTIG,LONG)
0054      M=MPCLB(IBUS1+IDADM,31888,B,512,B,FIXED,CONTIG,LONG)
0055      M=MPCLB(IBUS1+11,10246,100,2,1,B)
0056      CREATE PREBOUND FUNCTION
0057      M=MPCBF(11,25B)
0058      M=CVSD(BSHTR(1),5B,BSIN(1))
0059      M=CVSD(BSHTR(2),5B,BSIN(2))
0060      M=MPTEF(B)
0061      LOAD THE ADAM AND AOM
0062      M=ADMD(IDAOM,FREQ,B,BSHTR(2),BSHTR(1),2)
0063      M=ADMSD(IDADM,FREQ,1,B,B,CHAN1,BSIN(2),BSIN(1))
0064      M=PLDS(AOM,1052,IDAOM)
0065      M=PLDS(ADM,1052,IDAOM)
0066      M=MPVST(5B,SCAL(1),15,1)
0067
0068      C---
0069      C
0070      FUNCTION LIST
0071      M=MPBFL(FL1)
0072      M=MPVT(2,BSIN(1))
0073      M=MPXBF(25B)
0074      M=MPVT(2,BSIN(2))
0075      M=MPXBF(251)
0076      M=MPBFL(FL1)
0077      M=MPRAA(ADM,B,B,AOM,B,B)
0078      M=MPVHL(B,B,FL1)
0079      GO INTO WAIT STATE
0080      TYPE = '...' REAL TIME CVSD IS RUNNING ...
0081      TYPE = '...' TYPE IN B-STOP OR 1-CHANGE PARAMETERS'
0082      READ(5,*)LDAT
0083      IF(LDAT.NE.B.AND.LDAT.NE.1)GO TO 500
0084      M=MPSAA(AOM)
0085      M=MPSAA(ADM)
0086      M=MPCLS(B)
0087      IF(LDAT.EQ.1)GO TO 999
0088      STOP
0089      END

```

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	882522	681 RV,I,CON,LCL
2	SPDATA	888762	249 RV,D,CON,LCL
3	S1DATA	888328	184 RV,D,CON,LCL
4	SVARS	888248	88 RV,D,CON,LCL
5	STEPS	888882	1 RV,D,CON,LCL

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ADM	I*2	4-888116	ALT	I*2	4-888174	ADM	I*2	4-888112
DELTH	R*4	4-888284	FIXED	I*2	4-888164	FL1	I*2	4-888184
IBUS1	I*2	4-888228	IBUS2	I*2	4-888228	IBUS3	I*2	4-888224
IOS2	I*2	4-888186	ISIZE	I*2	4-888282	LDAT	I*2	4-888236
REAL	I*2	4-888168	SHORT	I*2	4-888178	SIZE	R*4	4-888176
						CMPLX	I*2	4-888162
						FREQ	R*4	4-888214
						IDADM	I*2	4-888114
						LONG	I*2	4-888166
						SSIZE	R*4	4-888232
						THRE	R*4	4-888218
						CONTIG	I*2	4-888172
						I	I*2	4-888238
						IDAOM	I*2	4-888118
						M	I*2	4-888226

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
BSNTR	I*2	4-888188	888884	2 (2)
BSIN	I*2	4-888874	888884	2 (2)
CHAN1	I*2	4-888128	888848	16 (16)
SCAL	R*4	4-888888	888874	38 (15)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
188	88	988	1-882348	999	1-888876

FUNCTIONS AND SUBROUTINES REFERENCED

ADMSD	ADMID	CLOS	CVSD	MPBFL	MPCBF	MPCLB	MPCLS	MPEFL	MPLDS	MPOPN	MPRAA	MPSAA	MPTBF	MPVHL	MPVST
MPUT	MPXBF	VSNAL													

TOTAL SPACE ALLOCATED = 884266 1115

CVSDBN,CVSDBN/-SP-CVSDBN

APPENDIX G

PROGRAM LISTINGS FOR REAL TIME IMPLEMENTATION OF PARC ON MAP-300

This appendix contains the program listings of all the modules in the real time implementation of the PARC algorithm on the MAP-300 array processor. There are twenty modules in all, some in the PDP-11, and others in the various processors of the MAP 300. Ten modules execute in the MAP-300. Nine of these have host support programs in the PDP-11. And finally, there is the main program in the PDP-11 which sets up and initializes the real time PARC operation on the MAP-300.

The following is a list of these programs and the processors they execute on.

1.	PARC Mainline	PDP-11
2-10.	Host Support Subroutines	PDP-11
11.	Receiver Update Module	CSPU/MAP-300
12.	Transmitter Update Module	CSPU/MAP-300
13.	Initialization Module	CSPU/MAP-300
14.	Pitch Computation Module	AP/MAP-300
15.	PARC Transmitter Module	AP/MAP-300
16.	PARC Receiver Module	AP/MAP-300
17.	Encoder Module	CSPU/MAP-300
18.	IOS Simulator	CSPU/MAP-300
19.	Decoder Module	CSPU/MAP-300
20.	Digital I/O Module	IOS/MAP-300

Operator Sequence to Run PARC at DCA

The two MAP's at DCA have different logical device names, MP and MA. There is a different task images corresponding to each MAP, called PARP.TSK and PARA.TSK. The task names for these programs are NDP and NDA.

To run the real time PARC algorithm, the following steps must be followed for each MAP. For the steps shown below, the devices are called MX. The underlined text is the computer response at the terminal. The rest of the text is the operator input.

```

> LOA MX:      (Load the map device)
> INS MXLD/TASK = ... XLO      (Install map loader program)
> XLO          (Load the map executive including PARC routines.
                The executive is called NDQL)
> INS PARX     (Install the initialization routines for PARC)
> NDX          (Run the initialization program)

```

MODE ?(1 - Internal Loopback, 2-Full Duplex)

```

2              (Mode Selection)

```

These steps for setting up and executing the PARC algorithm are combined into an indirect command file called PARC.CMD. If the indirect command file is used, the operator only needs to run the program (NDX) and select the mode.

PARC96.FTN

/TR:BLOCKS/VR

```

*****
"      PITCH EXTRACTION ADAPTIVE RESIDUAL CODER
"      P A R C
*****

```

MAR 25, 1985

```

REAL TIME IMPLEMENTATION INCLUDING ENCODER, DECODER,
IOS LOOP-BACK.

```

```

5551 REAL SCAL3(26), SCAL1(15), SCAL2(29), A1(85), A2(85)
5552 REAL OUTSCA(25), EXPN(25), T(25), B(25), HOBETA
5553 INTEGER ADM1, ADM2, SAMP, TXVHAT, T1SHAT, T2SHAT, TXPARA
5554 INTEGER INIS(14), KBLK, AOM1, AOM2, AOM3, AOM4
5555 INTEGER RCVHAT, R1SHAT, R2SHAT, RCPARA, BOUT1, BOUT2, TBLK, FBLK
5556 INTEGER RCVNT2
5557 INTEGER IOSPCH(255), SIZE
5558 INTEGER ENCT(35), ENCB(97), DECT(256), DECB(128)
5559 INTEGER PHSTA(8), DECTAB(2, 14), ISEL(3), OFFSET
5560 INTEGER ENTRB, ENTRT, DCRCCT, DCRCB
5561 INTEGER QTR1, QTR2, BTR1, BTR2, ORC1, ORC2, BRC
5562 INTEGER ITBTR1, ITBTR2, ITBRC1, ITBRC2
5563 INTEGER ADM, AOM, IOS, IOS2
5564 INTEGER IDADM, IDAOM, IDIOS
5565 INTEGER IFSEL, CHAN1(16)
5566 INTEGER REAL, CMPLX, FIXED, LONG, SHORT, CONTIG, ALT
5567 DATA IBUS1, IBUS2, IBUS3/64, 128, 192/
5568 DATA ADM1, ADM2, BOUT1, BOUT2/1, 2, 3, 4/
5569 DATA SAMP, TXVHAT, T1SHAT, T2SHAT, TXPARA/5, 6, 7, 8, 9/
5570 DATA RCVHAT, R1SHAT, RCVHAT2, RCPARA/15, 11, 12, 13/
5571 DATA ENTRB, ENTRT, DCRCCT, DCRCB/14, 15, 16, 17/
5572 DATA QTR1, QTR2, ORC1, ORC2, BTR1, BTR2, BRC/18, 19, 20, 21, 22, 23, 24/
5573 DATA IDADM, IDAOM, IDIOS/27, 28, 29/
5574 DATA CHAN1/1, -1, 14-B/
5575 DATA LONG, SHORT, REAL, CMPLX, FIXED, CONTIG, ALT/5, 1, 2, 1, 2, 1/
5576 DATA AOM1, AOM2, AOM3, AOM4/31, 32, 33, 34/
5577 DATA ADM, AOM, IOS, IOS2/23, 22, 16, 2/
5578 DATA ITBTR1, ITBTR2, ITBRC1, ITBRC2, IFSEL/88, 91, 94, 96, 85/
5579 DATA IFPQL, IFPRC, IFPIOS, IFMODE, IFSYNC, IFRCVR/1, 2, 3, 4, 5, 6/
5580 DATA IFSYN1, IFSYN2, IFREC1, IFREC2/7, 8, 9, 15/

```

```

5581 DATA ENCT/3, 7, 5, 2, 5, 3, 11, 5, 47,
1 6, 95, 7, 191, 2, 1, 3, 3, 5, 15,
2 6, 31, 7, 63, 7, 127, 7, 255, 7, 255/
5582 DATA DECTAB/14, -25, 5, 1, 5, 2, 13, 3, 61, 4,
1 125, 5, 253, 6, 4, 7, 12, 8, 65, 9,
2 124, 15, 252, 11, 254, -12, 255, -12/
5583 DATA ENCB/7, 11, 13, 14, 15, 19, 21, 22, 23, 25, 26, 27, 28, 29, 35, 31,
1 35, 37, 38, 39, 41, 42, 43, 44, 45, 46, 47, 49, 55, 51, 52, 53,
2 54, 55, 56, 57, 58, 59, 65, 61, 62, 63, 67, 69, 75, 71, 73, 74,
3 75, 76, 77, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 89, 95, 91,
4 92, 93, 94, 95, 97, 98, 99, 155, 151, 152, 153, 154, 155, 156,

```

```

FORTAN IV-PLUS V02-51      12:09:54      02-MAY-80      PAGE 2
PARC96.FTN

      C      5      107.108.
      C      6      109.110.111.112.113.114.115.116.117.118.119.120.
      C      7      121.122.123.124.125/

DATA PHBTA/B.1.2.4.8.16.32.64/

      C      INFORMATION AND PARAMETERS

      C      999
      D      OPEN(UNIT=4,NAME='PARA.DAT',SHARED,TYPE='OLD',READONLY)
      D      TYPE = 'TYPE IN FILTER PARAMETERS : A, B?'
      D      READ(4,*)SCAL1(2),SCAL1(3)
      D      TYPE = 'TRANSFER BLOCK SIZE (MULTIPLES OF 2) ?'
      D      READ(4,*)TBLK
      D      TYPE = 'FILTERING BLOCK SIZE?'
      D      READ(4,*)FBLK
      D      TYPE = 'CORRELATION BLOCK SIZE (MULTIPLES OF 4) ?'
      D      READ(4,*)KBLK
      D      INIS(3)=FBLK-1
      D      INIS(5)=KBLK-1
      D      INIS(7)=TBLK
      D      INIS(14)=TBLK
      D      FBLK=FBLK
      D      KBLK=KBLK
      D      TBULK=TBULK
      D      SCAL1(1)=TBULK/2.-0.5
      D      SCAL1(4)=FBLK-0.5
      D      SCAL1(5)=KBLK/4.-0.5
      D      SCAL3(5)=TBULK-0.5
      D      TYPE = 'RUN LENGTH, BIT 0 FOR LEVEL 1 AND BIT 0 FOR RUN LENGTH CODE ?'
      D      READ(4,*)SCAL2(5),BIT1,BITLNT
      D      SCAL2(16)=SCAL2(5)*BIT1-BITLNT
      D      NRUNE=SCAL2(5)-1
      D      SCAL2(15)=SCAL2(5)
      D      TYPE = 'PITCH PERIOD SEARCH RANGE: FROM ? TO ?'
      D      READ(4,*)ITSTRT,ITEND
      D      TSTRT=ITSTRT
      D      TEND=ITEND
      D      INIS(4)=ITSTRT-1
      D      INIS(10)=ITSTRT
      D      SCAL1(6)=TEND-TSTRT+0.5
      D      SCAL1(8)=TEND-0.5
      D      SCAL1(10)=TSTRT
      D      TYPE = 'BIT BLOCK SIZE, 0 BITS NULL CODE, GAP SIZE, 0 BITS PHONEY BETA ?'
      D      READ(4,*)SCAL2(2),SCAL2(21),IGAP,SCAL2(6)
      D      INIS(6)=IGAP
      D      INIS(9)=IGAP
      D      INIS(13)=IGAP
      D      GAP=IGAP
      D      SCAL2(7)=GAP-0.5
      D      SCAL3(19)=SCAL2(7)
      D      TYPE = 'GAP THRESHOLD & FILTER THRESHOLD ?'
      D      READ(4,*)INIS(1),INIS(2)

```

FORTRAN IV-PLUS V82-51 12:59:54 52-MAY-88 PAGE 3
PARC96.FTN /TR:BLOCKS/VR

```

C
D
TYPE ' ' VALUES FOR RMSHIN,ALAD,ALP,SMIN,G,AINV,INIT STATE?
READ(4,*)RMSHIN,ALAD,ALP,SMIN,G,AINV,STATE
N=4
SCAL2(18)=AINV*(1/ALAD-1)
SCAL2(19)=ALAD
SCAL2(20)=N-2.5
SCAL2(3)=RMSHIN/16.**3
SCAL2(4)=STATE
SCAL2(8)=RMSHIN/16.**3
SCAL2(9)=N-2.5
SCAL2(10)=1-ALP
SCAL2(11)=ALP
SCAL2(12)=SMIN
SCAL2(13)=G
SCAL3(6)=RMSHIN/16.**3
SCAL3(7)=STATE
SCAL3(9)=RMSHIN/16.**3
SCAL3(11)=ALAD
SCAL3(18)=N-2.5
SCAL3(15)=1-ALP
SCAL3(16)=ALP
SCAL3(12)=SMIN
SCAL3(14)=G
SCAL3(17)=AINV*(1/ALAD-1)

C
D
TYPE ' ' OF QUANTIZER LEVELS?
READ(4,*)IK
IL=IK/2
SCAL2(14)=IL
ILL=IL+1
TYPE ' ' OUTPUT SCALING FACTORS FROM 1 TO',ILL
TYPE ' ' EXPANSION/CONTRACTION FACTORS FROM 1 TO',ILL
READ(4,*)(EXP(J),J=1,ILL)
TYPE ' ' # BITS CORRESPONDING TO EACH LEVEL FROM 1 TO',IK
READ(4,*)(B(I),I=1,IK)
K1=9
K11=9
A1(K1)=(OUTSCA(1)+OUTSCA(2))/2
T(1)=A1(K1)
A1(K1+1)=OUTSCA(1)
A1(K1+2)=EXPN(1)
A1(K1+3)=B(1)
A1(K1+4)=B(1)
A2(K11)=EXPN(1)
A2(K11+1)=OUTSCA(1)
DO 48 I=2,ILL
T(I)=(OUTSCA(1)+OUTSCA(I+1))/2
K11=K11+2
A2(K11)=EXPN(1)
A2(K11+1)=OUTSCA(1)
K1=K1+5
A1(K1)=T(1)
A1(K1+1)=OUTSCA(1)
A1( 2)=( 1)

```



```

FORTRAN IV-PLUS V2-51      12:59:54      82-MAY-88      PAGE 5
PARC96.FTN      /TR:BLOCKS/VR

      INIS(8)=1823
      INIS(11)=INIS(8)
      INIS(12)=2848
      A1(1)=AINV
      A2(1)=AINV
      CLOSE(UNIT=4)

      CREATE INFORMATION FILE

      WRITE(6,*) '***** REAL-TIME PARC ALGORITHM *****'
      WRITE(6,*) 'PARAMETERS FOR THE ADAPTIVE FILTER:'
      WRITE(6,*) 'A=', SCAL1(2), 'B=', SCAL1(3)
      WRITE(6,*) 'TRANSFER BLOCK SIZE =', TBLK
      WRITE(6,*) 'FILTERING BLOCK SIZE =', FBLK
      WRITE(6,*) 'CORRELATION BLOCK SIZE =', KBLK
      WRITE(6,*) 'RUN LENGTH =', SCAL2(5), 'BIT # FOR ITS CODE =', BITLMT
      WRITE(6,*) 'BIT # FOR LEVEL 1 =', BIT1
      WRITE(6,*) 'PITCH PERIOD SEARCHING RANGE =', ITSTRT, 'TO', ITEND
      WRITE(6,*) 'BIT BLOCK SIZE =', SCAL2(2), 'NULL BIT # =', SCAL2(21)
      WRITE(6,*) 'GAP SIZE =', IGAP, 'PHONY BETA BITS =', SCAL2(6)
      WRITE(6,*) 'RMSMIN =', RMSMIN, 'ALAD =', ALAD, 'ALP =', ALP
      WRITE(6,*) 'SMIN =', SMIN, 'G =', G, 'AINV =', AINV, 'STATE =', STATE
      WRITE(6,*) 'NUMBER OF QUANTIZER LEVELS =', IK
      WRITE(6,*) 'OUTPUT SCALING FACTOR:'
      WRITE(6,*) 'OUTSCA(1), I=1, ILL)
      WRITE(6,*) 'EXPANSION/CONTRACTION FACTORS:'
      WRITE(6,*) 'EXPN(1), I=1, ILL)
      WRITE(6,*) 'QUANTIZER THRESHOLDS:'
      WRITE(6,*) 'T(1), I=1, ILL)
      WRITE(6,*) 'OF BITS CORRESPONDING TO EACH LEVEL:'
      WRITE(6,*) 'UB(1), I=1, IK)
      WRITE(6,*) 'UPPER BETA LIMIT =', UBETA, 'LOWER BETA LIMIT =', SCAL1(11)
      WRITE(6,*) 'OF QUANTIZING LEVELS FOR BETA =', IOBETA
      WRITE(6,*) 'GAP THRESHOLD =', INIS(1), 'FILTER THRESHOLD =', INIS(2)
      WRITE(6,*) 'ADAM SAMPLING FREQ =', FREADM, 'AOM SAMPLING FREQ =', FREADM
      WRITE(6,*) 'INV. GAIN FACTOR FOR MASKING LBSB =', 1./((SCAL3(24)*2.**13)
      WRITE(6,*) 'MAX. O-BUFFER LENGTH =', SCAL3(25)
      WRITE(6,*) 'GAIN FACTOR AT ADAM INPUT =', SCAL3(24)*SCAL3(26)

      C----- SET UP ROUTINE FOR IOS TRANSFER
      C
      CALL ASSIGN(3, 'SY:IOSDC.CSP')
      CALL FPMID(IOSPGM, SIZE, MSGERR)
      IF (MSGERR.EQ.0) GO TO 88
      TYPE = '*** ERROR *** MSGERR, ' IN IOS PROGRAM'
      STOP
      CONTINUE
88
      C----- SET IOS DATA RATE
      C
      CNTS=256.-(1.536E6/(4.*2.*FREADM))
      CNTD=256.-(1.536E6/(8.*1.5.*FREADM))
      CNTIOS=256.-CNTD+CNTS
      IF (CNTIOS.GT.32767.) CNTIOS=CNTIOS-65536.
      IOSPGM(2)=CNTIOS

```

FORTMAN IV-PLUS V02-51 12:09:54 02-MAY-80 PAGE 6
PARC96.FTN /TR:BLOCKS/VR

C ----- SET UP HEADER AND TRAILER BLOCK FOR IOS FUNCTION

```

#182 DO 200 I=1,SIZE
#183 I1=SIZE+1-I
#184 IOSPGM(I1+2)=IOSPGM(I1)
#185
#186 IOSPGM(1)=SIZE
IOSPGM(2)=256*BRC+BTR1
#187
#188 IOSPGM(SIZE+2+1)=B
#189 IOSPGM(SIZE+2+2)=2
#190 IOSPGM(SIZE+2+3)=256*B+BTR2
#191 IOSPGM(SIZE+2+4)=B
#192 IOSPGM(SIZE+2+5)=B
#193 IOSPGM(SIZE+2+6)=1
#194 IOSPGM(SIZE+2+7)=-1
#195 IOSPGM(SIZE+2+8)=-1
#196 IOSPGM(SIZE+2+9)=-1
IOSPGM(SIZE+2+10)=-1

```

MAP BUFFER COFIGURATION

```

#197 CONTINUE
#198 M=MPOPN(1)
#199 M=MPIBC(B)
#200 M=MPCLB(1BUS3+ADM1,3072..126..REAL,CONTIG,SHORT)
#201 M=MPCLB(1BUS3+ADM2,3072..126..126..REAL,CONTIG,SHORT)
#202 M=MPCLB(1BUS2+AOM1,3584..252..REAL,CONTIG,SHORT)
#203 M=MPCLB(1BUS2+AOM2,3840..252..REAL,CONTIG,SHORT)
#204 M=MPCLB(1BUS2+AOM3,3584..126..REAL,CONTIG,SHORT)
#205 M=MPCLB(1BUS2+AOM4,3840..126..REAL,CONTIG,SHORT)
#206 M=MPCLB(1BUS3+SAMP,B..1024..REAL,CONTIG,SHORT)
#207 M=MPCLB(1BUS2+TXVHAT,B..1024..REAL,CONTIG,SHORT)
#208 M=MPCLB(1BUS3+T1SHAT,1024..1024..REAL,CONTIG,SHORT)
#209 M=MPCLB(1BUS2+TXPARA,3072..80..REAL,CONTIG,LONG)
#210 M=MPCLB(1BUS3+RCVHAT,2048..1024..REAL,CONTIG,SHORT)
#211 M=MPCLB(1BUS3+RCVHT2,3068..4..REAL,CONTIG,SHORT)
#212 M=MPCLB(1BUS2+R1SHAT,2048..1024..REAL,CONTIG,SHORT)
#213 M=MPCLB(1BUS2+RCPARA,4096..80..REAL,CONTIG,LONG)
#214 M=MPCLB(1BUS1+IDAOM,23000..250..FIXED,CONTIG,LONG)
#215 M=MPCLB(1BUS1+IDADM,23250..250..FIXED,CONTIG,LONG)
#216 M=MPCLB(1BUS1+IDIOS,23500..200..FIXED,CONTIG,LONG)
#217 M=MPCLB(1BUS1+ENTRB,24000..100..FIXED,CONTIG,LONG)
#218 M=MPCLB(1BUS1+ENTRT,23900..30..FIXED,CONTIG,LONG)
#219 M=MPCLB(1BUS1+DCRCT,24100..256..FIXED,CONTIG,LONG)
#220 M=MPCLB(1BUS1+DCRCB,24400..128..FIXED,CONTIG,LONG)
#221 M=MPCLB(1BUS1+QTR1,25000..600..FIXED,CONTIG,LONG)
#222 M=MPCLB(1BUS1+BTR1,25700..200..FIXED,CONTIG,LONG)
#223 M=MPCLB(1BUS1+QRC1,26000..600..FIXED,CONTIG,LONG)
#224 M=MPCLB(1BUS1+QTR2,27000..600..FIXED,CONTIG,LONG)
#225 M=MPCLB(1BUS1+BTR2,27700..200..FIXED,CONTIG,LONG)
#226 M=MPCLB(1BUS1+QRC2,28000..600..FIXED,CONTIG,LONG)
#227 M=MPCLB(1BUS1+BRC,29000..1024..FIXED,CONTIG,LONG)

```

FORTRAN IV-PLUS V02-51 12:59:54 82-MAY-88 PAGE 7
PARC96.FTN /TR:BLOCKS/VR

```

C
C
C      INITIALIZE THESE BUFFERS
M=VSMAL(ADM1,B,ADM1,B)
M=VSMAL(ADM2,B,ADM2,B)
M=VSMAL(AOM1,B,AOM1,B)
M=VSMAL(AOM2,B,AOM2,B)
M=VSMAL(AOM3,B,AOM3,B)
M=VSMAL(AOM4,B,AOM4,B)
M=VSMAL(SAMP,B,SAMP,B)
M=VSMAL(TXVHAT,B,TXVHAT,B)
M=VSMAL(TISHAT,B,TISHAT,B)
M=VSMAL(RCVHAT,B,RCVHAT,B)
M=VSMAL(RISHAT,B,RISHAT,B)

C
C
C      INITIALIZE Q-BUFFERS FOR ENCODER
DO 98 I=1,138
  DECT(I)=2
  DECT(I)=1
  DECT(128)=-12
  DECT(129)=-12
  DECT(138)=-12
M=MPVDB(OTR1,DECT(1),2,B,DECT(138))
M=MPVDB(OTR2,DECT(1),2,B,DECT(138))

C
C
C      CONSTRUCT DECODE TABLES
1.DECT - SIZE:256
DO 188 I=1,256
  DECT(I)=B
  CONTINUE
188 C

DO 182 I=1,14
  II=DECTAB(1,I)+1
  DECT(II)=DECTAB(2,I)+2
  CONTINUE
182 C

2.DECB - SIZE:128
DO 185 I=1,128
  DECB(I)=48
  CONTINUE
185 C

DO 186 I=1,97
  II=ENCB(1)+1
  DECB(II)=I-1
  CONTINUE
186 C

DO 187 I=1,8
  II=PHBTA(1)+1
  DECB(II)=-1
  CONTINUE
187 C

C
C
C      INITIALIZE FUNCTION LIST SELECTOR

```

FORTRAN IV-PLUS V02-51 12:59:54 02-MAY-80 PAGE 8
PARC96.FTN /TR:BLOCKS/VR

```

C
#265      ISEL(1)=B
#266      ISEL(2)=-1
#267      ISEL(3)=56

C----- OFFSET
C
#268      OFFSET=2

C----- INITIALIZE ENCODE-DECODE TABLES
C
#269      M=MPVDB(ENTRT,ENCT(1),2,B,ENCT(3B))
#270      M=MPVDB(ENTRB,ENCB(1),2,B,ENCB(97))
#271      M=MPVDB(DCRCT,DECT(1),2,B,DECT(256))
#272      M=MPVDB(DCRB,DECB(1),2,B,DECB(128))
C      INITIALIZE PARAMETERS, SCALARS FOR PARC
C
#273      M=MPVST(5B,SCAL1(1),15,1)
#274      M=MPVST(6B,SCAL2(1),29,1)
#275      M=MPVST(9A,SCAL3(1),26,1)
#276      M=MPVST(5B,INIS(1),14,1)
#277      M=MPVDB(TXPARA,A1(1),4,1,A1(8B))
#278      M=MPVST(IFSEL,ISEL(1),3,B)
#279      M=MPVDB(RCPARA,A2(1),4,1,A2(8B))

C----- SET UP IOS TRANSFER ROUTINE AND LOAD IOS
C
#280      M=MPVDB(IDIOS,IOSPGM(1),2,B,IOSPGM(2B))
#281      M=MPLDS(IOS,IOS2,IDIOS)

C----- LOAD ADAM, AOM FUNCTIONS
C
#282      M=ADMSD(IDADM,2,B,1,B,2,CHAN1,ADM2,ADM1)
#283      M=AOMID(IDAOM,1,B,2,AOM2,AOM1,OFFSET)
#284      M=MPLDS(ADM,IOS2,IDAOM)
#285      M=MPLDS(AOM,IOS2,IDAOM)

C----- FUNCTION LISTS
C
C----- FUNCTION LIST FOR Q-LEVEL LOOP-BACK OPERATION
C
#286      M=MPBFL(IFPRC)
#287      M=MPVT(2,ADM1)
#288      M=PICH3(ITBTR1,ADM1)
#289      M=PCTX(ITBTR1+2,QTR1)
#290      M=MPVT(1,1)
#291      M=RC(ITBTR1,59)
#292      M=TX(56)
#293      M=PCRC(ITBTR1+1,QTR1,AOM3)
#294      M=MPVT(2,ADM2)
#295      M=PICH3(ITBTR2,ADM2)
#296      M=PCTX(ITBTR2+2,QTR2)
#297      M=MPVT(1,1)
#298      M=RC(ITBTR2,59)

```

188 ---> 94
1 SAME AS ISEL(3)
189 ---> 95

191 ---> 96


```

FORTAN IV-PLUS V82-S1      12:59:54      82-MAY-88      PAGE 18
PARC96.FTN /TR:BLOCKS/VR

#337 M=PICH3(ITBTR2,ADM2)
#338 M=PCIX(ITBTR2+2,QTR2)
#339 M=DECD(BRC,IFSEL,ORC1,ITBRC1,ORC2,ITBRC2,DCRCT,DCRCB,IGAP)
#340 M=MPVT(1,1)
#341 M=TX(56)
#342 M=MPEFL(B)

C ----- PARC FULL OP FUNC LIST - 1
C
#343 M=MPBFL(IFREC1)
#344 M=MPVT(2,ADM1)
#345 M=RC(ITBRC1,59)
#346 M=PICH3(ITBTR1,ADM1)
#347 M=ENCD(QTR2,BTR2,ENTRB,ENTRT,ITBTR2,ITBRC1,59)
#348 M=PCIX(ITBTR1+2,QTR1)
#349 M=PCRC(ITBRC1+1,ORC1,ADM3)
#350 M=DECD(BRC,IFSEL,ORC1,ITBRC1,ORC2,ITBRC2,DCRCT,DCRCB,IGAP)
#351 M=MPVT(1,1)
#352 M=TX(56)
#353 M=MPEFL(B)

C ----- PARC FULL OP FUNC LIST - 2
C
#354 M=MPBFL(IFREC2)
#355 M=MPVT(2,ADM2)
#356 M=RC(ITBRC2,59)
#357 M=PICH3(ITBTR2,ADM2)
#358 M=ENCD(QTR1,BTR1,ENTRB,ENTRT,ITBTR1,ITBRC2,59)
#359 M=PCIX(ITBTR2+2,QTR2)
#360 M=PCRC(ITBRC2+1,ORC2,ADM4)
#361 M=DECD(BRC,IFSEL,ORC1,ITBRC1,ORC2,ITBRC2,DCRCT,DCRCB,IGAP)
#362 M=MPVT(1,1)
#363 M=TX(56)
#364 M=MPEFL(B)

C ----- FUNCTION LIST TO RUN PARC WITH Q-LEVEL LOOP-BACK
C
#365 M=MPBFL(IFPOL)
#366 M=INI(58,NRUNL)
#367 M=MPRNS(105,1052,B)
#368 M=MPRAA(ADM,B,ADM,B)
#369 M=MPVT(2,ADM2)
#370 M=RCINI(61)
#371 M=MPVHL(B,B,IFPRC)
#372 M=MPEFL(B)

C ----- READY TO RUN. SELECT MODE AND GO
C
#373 CONTINUE
#374 TYPE ' ' - SELECT OPERATION MODE:
#375 TYPE ' ' - STOP PROGRAM.
#376 TYPE ' ' 1 - Q-LEVEL LOOP-BACK IN MEMORY. (NO 105)
#377 TYPE ' ' 2 - BIT STREAM LOOP-BACK THRU 105. (FULL DUP OP.)
#378 ACCEPT ' ' LMODE
#379 GO TO(75B,58B,68B) LMODE+1

```

FORTRAN IV-PLUS V02-51 12:59:54 82-MAY-88 PAGE 11
PARC96.FTH /TR:BLOCKS/VR

```

C ----- OPERATION MODE 1
C
8388 M-MPXFL(IFPOL)
8389 TYPE '...' PARC IN Q-LEVEL LOOP-BACK MODE (NO IOS)'
8390 TYPE '...'
8391 GO TO 788
C ----- OPERATION MODE 2
C
8394 M-MPXFL(IFPIOS)
8395 TYPE '...' PARC IN FULL DUPLEX MODE (THRU IOS)'
8396 TYPE '...'
C ----- STOP ADM,ADM,IOS, CLOSE MAP, AND STOP.
C
788 CONTINUE
      PAUSE ' TYPE RES... TO CONTINUE'
      M-MPSAA(ADM)
      M-MPSAA(ADM)
      M-MPCLS(B)
      GO TO 85
8397 CONTINUE
8398 M-MPCLS(B)
8399 STOP
8400 END

```

PAGE 12

12:59:54 82-MAY-88

FORTRAN IV-PLUS V82-51
PARC96.FTN /TR:BLOCKS/VR

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	812752 2885	RV,I.CON,LCL
2	SFOATA	881838 268	RV,D.CON,LCL
3	SIDATA	882824 522	RV,D.CON,LCL
4	SVARS	885756 1527	RV,D.CON,LCL
5	STEMPS	888884 2	RV,D.CON,LCL

VARIABLES

NAME	TYPE	ADDRESS	NAME	SIZE	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ADM	I*2	4-885448	ADM1		I*2	4-882334	ADM2	I*2	4-882336	AINV	R*4	4-885664
ALP	R*4	4-885658	ALT		I*2	4-885534	AOM	I*2	4-885442	AOM1	I*2	4-882418
AOM3	I*2	4-882414	AOM4		I*2	4-882416	BITLMT	R*4	4-885618	BITI	R*4	4-885684
BOUT2	I*2	4-882432	BRC		I*2	4-885426	BTRI	I*2	4-885416	BTR2	I*2	4-885428
CNTD	R*4	4-885748	CNTIOS		R*4	4-885744	CNTS	R*4	4-885734	CONTIG	I*2	4-885532
DCRCT	I*2	4-885486	ENTR8		I*2	4-885482	ENTRT	I*2	4-885484	FBLK	I*2	4-882436
FIXED	I*2	4-885524	PREADM		R*4	4-885726	G	R*4	4-885668	GAP	R*4	4-885634
I	I*2	4-885786	IBUS1		I*2	4-885536	IBUS2	I*2	4-885548	IBUS3	I*2	4-885542
IDAOM	I*2	4-885452	IDIOS		I*2	4-885454	IFMODE	I*2	4-885552	IFPIOS	I*2	4-885558
IFPRC	I*2	4-885546	IFRCVR		I*2	4-885556	IFREC1	I*2	4-885564	IFREC2	I*2	4-885566
IFSYN1	I*2	4-885554	IFSYN1		I*2	4-885568	IFSYN2	I*2	4-885562	IGAP	I*2	4-885632
IK	I*2	4-885676	IL		I*2	4-885788	ILL	I*2	4-885782	IOS	I*2	4-885444
IOBETA	I*2	4-885724	ITBRC1		I*2	4-885434	ITBRC2	I*2	4-885436	ITBTR1	I*2	4-885438
ITEND	I*2	4-885628	ITSTRT		I*2	4-885616	J	I*2	4-885784	KBLK	I*2	4-885752
K11	I*2	4-885712	LMODE		I*2	4-885754	LONG	I*2	4-885526	M	I*2	4-885718
N	I*2	4-885674	NRUNL		I*2	4-885514	OFFSET	I*2	4-885488	OBETA	R*4	4-885728
ORC2	I*2	4-885424	QTR1		I*2	4-885412	QTR2	I*2	4-885414	RCPARA	I*2	4-882426
RCVHT2	I*2	4-882448	REAL		I*2	4-885528	RKBLK	R*4	4-885574	RMSMIN	R*4	4-885648
R2SHAT	I*2	4-882424	SAMP		I*2	4-882348	SHORT	I*2	4-885538	SIZE	I*2	4-883262
STATE	R*4	4-885678	TBLK		I*2	4-882434	TEND	R*4	4-885626	TSTRT	R*4	4-885622
TXPARA	I*2	4-882358	TXVHAT		I*2	4-882342	TISHAT	I*2	4-882344	T2SHAT	I*2	4-882346

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
A1	R*4	4-888438	888588	(88)
A2	R*4	4-888118	888588	(88)
B	R*4	4-882218	888128	(28)
CHAN1	I*2	4-885468	888848	(16)
DECB	I*2	4-884662	888488	(128)
DECT	I*2	4-883662	888188	(256)
DECTAB	I*2	4-885382	888878	(2, 14)
ENCB	I*2	4-883368	888382	(97)
ENCT	I*2	4-883264	888874	(38)
EXPN	R*4	4-881758	888128	(28)
INIS	I*2	4-882352	888834	(14)
IOSPGM	I*2	4-882442	888628	(288)
ISEL	I*2	4-885372	888886	(3)
OUTSCA	R*4	4-881638	888128	(28)

```

TRC:FTN
/ TR:BLOCKS/VR
*** HOST SUPPORT MODULES FOR PARC ***
THERE ARE SIX HOST SUPPORT MODULES FOR PARC AS FOLLOWS:
INI(ISA,LRUN) --- INITIALIZATION PROGRAM FOR ALL APS MODULES.
TX(ISA) --- UPDATE PROGRAM FOR THE APS MODULES OF PARC-TRANSMITTER
RC(ISA,ISA) --- UPDATE PROGRAM FOR THE APS MODULES OF PARC-RECEIVER
PIC3(SA,U) --- FOR BETA, T CALCULATION AND FILTER, GAPPING OPERATIONS.
PC3(SA,U) --- PARC-TRANSMITTER
PCRC(SA,U,V) --- PARC-RECEIVER
RCINI(ISA) --- INITIALIZATION PROGRAM FOR PARC-RECEIVER

INTEGER FUNCTION INI(ISA,LRUN)
INI(ISA,LRUN) --- INITIALIZATION PROGRAM FOR ALL APS MODULES.
FCB = 115
CALL FORMAT:
MAP=INI(ISA,LRUN)
WHERE ISA --- THE STARTING INTEGER SCALAR LOCATION
LRUN --- RUN LENGTH FOR ENCODER

INTEGER FCBRG,FCBSZ,HVS,FCB,MPDCB,HRD,LEVEL
INTEGER ISA,LRUN
COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL

INI=0
DO 100 I=1,11
FCBRG(I)=0
FCBRG(2)=115
CHECK ARGUMENT
IF(ISA.LT.0.OR.ISA.GT.127)INI=-1

IF(INI.EQ.0)GO TO 200
CALL MPERR(INI)
RETURN

FCBRG(3)=ISA
FCBRG(5)=LRUN

INI=RUNMP(FCBRG(1),FCBSZ(1,4))

RETURN
END

```

FORTAN IV-PLUS V02-51
TRRC.FTN /TR:BLOCKS/VR

12:47:20 15-APR-88

PAGE 2

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	000222	73
3	SIDATA	000014	6
4	SVAR	000006	3
6	MPZZZ	000312	101

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
INI	I*2	1-000000						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	I*2	4-000002	HR1	R*4	6-000304	HVS	I*2	4-000000
LEVEL	I*2	6-000310	LRUN	I*2	F-000004*	ISA	I*2	F-000002*

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
FCB	I*2	6-000260	000014	6
FCBRG	I*2	6-000000	000026	11
FCBSZ	I*2	6-000026	000232	77
MPDCB	I*2	6-000274	000010	4

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000144		

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 000556 183

```

FORTRAN IV-PLUS V02-51      12:47:28      15-APR-80      PAGE 3
TRNC.FTN /TR:BLOCKS/WR

      C
      C      INTEGER FUNCTION TX(ISA)
      C 11. TX(ISA) --- UPDATE PROGRAM FOR APS MODULES FOR PARC-TRANSMITTER
      C
      C      FCB = 114
      C
      C      CALL FORMAT:
      C
      C      MAP=TX(ISA)
      C      WHERE ISA --- THE STARTING INTEGER SCALAR LOCATION
      C
      C      INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRD,LEVEL
      C      INTEGER ISA
      C      COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
      C
      C      TX=0
      C      DO 100 I=1,11
      C      FCBRG(I)=0
      C      FCBRG(2)=114
      C      CHECK ARGUMENT
      C      IF(ISA.LT.0.OR. ISA.GT. 127)TX=-1
      C
      C      IF(TX.EQ.0)GO TO 200
      C      CALL MPERR(TX)
      C      RETURN
      C
      C      FCBRG(3)=ISA
      C      TX=RUNMP(FCBRG(1),FCBSZ(1,3))
      C
      C      RETURN
      C      END

```

FORTAN IV-PLUS V02-51
TRAC.FTM /TR:BLOCKS/VR

12:47:20 15-APR-88

PAGE 4

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	000214 78	RV,I,CON,LCL
3	SIDATA	000014 6	RV,D,CON,LCL
4	SVARS	000006 3	RV,D,CON,LCL
6	MPZZZ	000312 101	RV,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
TX	1*2	1-000000						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	1*2	4-000002	HRI	R*4	6-000304	HVS	1*2	4-000000
LEVEL	1*2	6-000310				I	1*2	4-000004
						ISA	1*2	F-000002*

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
FCB	1*2	6-000260	000014	6 (6)
FCBRC	1*2	6-000000	000026	11 (11)
FCBSZ	1*2	6-000026	000232	77 (11,7)
MPDCB	1*2	6-000274	000010	4 (4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000144		

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 000550 100

```

FORTRAN IV-PLUS V02-51      12:47:34      15-APR-88      PAGE 5
TRRC.FTN /TR:BLOCKS/VR

C
C
0001      INTEGER FUNCTION RC(SA,ISA)
C
C 111.  RC(SA,ISA) --- UPDATE PROGRAM FOR APS MODULES OF PARC-RECEIVER
C
C      FCB = 113
C
C      CALL FORMAT:
C
C      MAP=RC(SA,ISA)
C      WHERE SA --- SCALAR LOCATION FOR T
C      ISA --- THE STARTING INTEGER SCALAR LOCATION
C
C      INTEGER FCBRG,FCBSZ,HVS,FCB,MPDCB,HRD,LEVEL
C      INTEGER SA,ISA
C      COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),MRI,LEVEL
C
C      RC=0
C      DO 100 I=1,11
C      FCBRG(I)=0
C      FCBSZ(2)=113
C      CHECK ARGUMENTS
C      IF(SA.LT.0.OR.SA.GT.127)RC=-1
C      IF(ISA.LT.0.OR.ISA.GT.127)RC=-2
C
C      IF(RC.EQ.0)GO TO 200
C      CALL MPERR(RC)
C      RETURN
C
C      FCBRG(9)=SA
C      FCBRG(11)=ISA
C      RC=RUNMP(FCBRG(1),FCBSZ(1,5))
C
C      RETURN
C      END
0017
0018

```

FORTRAN IV-PLUS V02-51
TRNC.FTN /TR:BLOCKS/VR

12:47:34 15-APR-80

PAGE 6

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	83	RV,I.COM,LCL
3	SIDATA	6	RV,D.COM,LCL
4	SVARS	3	RV,D.COM,LCL
6	MPZZZ	101	RV,D.OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
RC	1*2	1-000000						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	1*2	4-000002	HR1	R*4	6-000304	HVS	1*2	4-000000
LEVEL	1*2	6-000310	SA	1*2	F-000002*	ISA	1*2	4-000004
								F-000004*

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
FCB	1*2	6-000260	000014	6 (6)
FCBGC	1*2	6-000000	000026	11 (11)
FCBSZ	1*2	6-000026	000232	77 (11,7)
MPDCB	1*2	6-000274	000010	4 (4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000170		

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED - 000602 193

```

FORTRAN IV-PLUS V02-51      12:47:41    15-APR-88      PAGE 7
TRAC.FTN                    /TR:BLOCKS/VR

      C
      C
0001      C      INTEGER FUNCTION PICH3(SA,U)
      C IV.    PICH3 -- FOR BETA, T COMPUTATION AND FILTER, GAPPING OPERATIONS
      C
      C      FCB = 248
      C
      C      CALL FORMAT:
      C      MAP=PICH3(SA,U)
      C
      C      WHERE SA --- SCALAR LOCATION FOR ENCD. BETA, BETA AND T
      C      U --- INPUT ADAM BUFFER
      C
      C      INTEGER SA,U
      C      IDUMY=8
      C      PICH3=FCB6N(248,B.SA,U,IDUMY,B,B,B,6)
      C      RETURN
      C      END
0002
0003
0004
0005
0006

```

FORTRAM IV-PLUS V02-51
TRRC.FTN /TR:BLOCKS/VR

12:47:41 15-APR-88

PAGE 8

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	000102	RV,I,CON,LCL
2	SPDATA	000014	RV,D,CON,LCL
3	SIDATA	000038	RV,D,CON,LCL
4	SVARS	000002	RV,D,CON,LCL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
PICH3	I*2	1-000000						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
IDUHV	I*2	4-000000	SA	I*2	F-000002	U	I*2	F-000004

FUNCTIONS AND SUBROUTINES REFERENCED

FCBGN

TOTAL SPACE ALLOCATED = 000158 52

FORTRAN IV-PLUS V02-51 12:47:45 15-APR-68 PAGE 9
TRRC.FTN /TR:BLOCKS/VR

```

0001      C      INTEGER FUNCTION PCTX(SA,U)
          C V.    PCTX -- PARC-TRANSMITTER
          C      FCB = 242
          C      CALL FORMAT:
          C      MAP=PCTX(SA,U)
          C      WHERE SA --- THE SCALAR LOCATION FOR BETA
          C      U --- OUTPUT QUANTIZING BUFFER
          C
          C      INTEGER SA,U
          C      IDUMY=0
          C      PCTX=FCBGN(242,B,SA,U,IDUMY,B,B,B,6)
          C      RETURN
          C      END
0002
0003
0004
0005
0006

```

FORTMAN IV-PLUS V02-S1
TRRC.FTN /TR:BLOCKS/WR

12:47:45 15-APR-88

PAGE 18

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	000102	33 RV,I.CON,LCL
2	SPDATA	000014	6 RV,D.CON,LCL
3	SIDATA	000038	12 RV,D.CON,LCL
4	SVARS	000002	1 RV,D.CON,LCL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
PCTX	I*2	1-000000						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
IDUHY	I*2	4-000000	SA	I*2	F-000002	U	I*2	F-000004

FUNCTIONS AND SUBROUTINES REFERENCED

FCBGN

TOTAL SPACE ALLOCATED = 000158 52

```

FORTRAN IV-PLUS V02-51          PAGE 11
TRRC.FTN                        12:47:48    15-APR-68

C
C
0001      INTEGER FUNCTION PCRC(SA,U,V)
C
C VI.      PCRC ---- PARC-RECEIVER
C
C          FCB = 243
C
C          CALL FORMAT:
C          MAP=PCRC(SA,U,V)
C
C          WHERE SA --- SCALAR NUMBER FOR INPUT ENCD. BETA AND T
C          V --- AOM BUFFER
C          U --- INPUT QUANTIZING BUFFER
C
C          INTEGER SA,U,V
C          PCRC=FCBGN(243,B,SA,U,B,V,B,B,7)
C          RETURN
C          END
0002
0003
0004
0005

```

FORTAN IV-PLUS V82-S1
TRAC.FTH

12:47:48

15-APR-88

PAGE 12

/TR:BLOCKS/VR

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	34	RV,I,CON,LCL
2	SPDATA	6	RV,D,CON,LCL
3	SIDATA	12	RV,D,CON,LCL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
PCRC	I*2	1-888888						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
SA	I*2	F-888882*	U	I*2	F-888884*	V	I*2	F-888886*

FUNCTIONS AND SUBROUTINES REFERENCED

FCBGN

TOTAL SPACE ALLOCATED = 888158 52

```

FORTRAN IV-PLUS V02-51      12:47:52      15-APR-88      PAGE 13
TRAC.FTN      /TR:BLOCKS/VR

C
0001      INTEGER FUNCTION RCINI(ISA)
C
C VII. RCINI(ISA) --- INITIALIZATION PROGRAM FOR APS MODULES OF RECEIVER.
C
C
C FCB = 116
C
C CALL FORMAT:
C
C MAP-RCINI(ISA)
C WHERE ISA --- THE STARTING INTEGER SCALAR LOCATION
C
C INTEGER FCBRG,FCBSZ,HVS,FCB,MPDCB,HRD,LEVEL
C INTEGER ISA
C COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL

C
0002      RCINI=0
0003      DO 100 J=1,11
0004      FCBRG(1)=0
0005      FCBRG(2)=116
0006      CHECK ARGUMENT
0007      IF(ISA .LT. 0 .OR. ISA .GT. 127)RCINI=-1
C
0008      IF(RCINI .EQ. 0)GO TO 200
0009      CALL MPERR(RCINI)
0010      RETURN
C
0011      FCBRG(3)=ISA
C
0012      RCINI=RUNMP(FCBRG(1),FCBSZ(1,3))
C
0013      RETURN
0014      END
0015
0016

```

FORTRAN IV-PLUS V02-51
TRRC.FTN /TR:BLOCKS/JR

12:47:52 15-APR-88

PAGE 14

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	71	RV,I,CON,LCL
3	STDATA	6	RV,D,CON,LCL
4	SVARS	3	RV,D,CON,LCL
6	MPZZZ	101	RV,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
RCINI	I*2	1-000000						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	I*2	4-000002	HR1	R*4	6-000304	HVS	I*2	4-000000
LEVEL	I*2	6-000310				I	I*2	4-000004
						ISA	I*2	F-000002*

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
FCB	I*2	5-000260	000014	6 (6)
FCBRG	I*2	6-000000	000026	11 (11)
FCBSZ	I*2	6-000026	000232	77 (11,7)
MPDCB	I*2	6-000274	000010	4 (4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000106		

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 000552 101

FORTRAN IV-PLUS V#2-B1
TRAC.FTN /TR:BLOCKS/VR PAGE 15

12:47:58 15-APR-88

C
C

.TRAC/NOSP=TRRC

```

FORTRAN IV-PLUS V02-S1      12:48:04      15-APR-80      PAGE 1
ENCD.FTN /TR:BLOCKS/VR

      C
0001      INTEGER FUNCTION ENCD(QHAT,OUTB,BETA,ENCT,ISTB,SID,ISID)
      C ----- HOST SUPPORT MODULE FOR SOURCE CODING THE Q-OUTPUT OF
      C           THE PARC ALGORITHM.
      C
      C FCB=110
      C
      C CALLING SEQUENCE:
      C M=ENCD(QHAT,OUTB,BETA,ENCT,ISTB,SID,ISID)
      C
      C WHERE:
      C QHAT = Q-OUTPUT BUFFER, LENGTH=400
      C OUTB = OUTPUT BIT BUFFER, LENGTH=200
      C BETA = CODING TABLE FOR 99 BETA VALUES, LENGTH=100
      C ENCT = SOURCE CODE BUFFER, LENGTH=30
      C           CONTAINS FOR EACH LEVEL:
      C             1ST WORD: CODE LENGTH-1
      C             2ND WORD: CODE
      C
      C ISTB = SID IN INTEGER TABLE WHERE T,BETA ARE LOCATED
      C SID = SCALAR ID FOR RCVR UPDATE
      C ISID = INT. SCALAR ID FOR RCVR UPDATE
      C
      C INTEGER FCBRG,FCBSZ,HVS,FCB,MPDCB,HRD,LEVEL
      C INTEGER QHAT,OUTB,BETA,ENCT,ISTB,SID,ISID
      C COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
      C
      C ENCD=0
      C DO 100 I=1,11
      C   FCBRG(I)=0
      C   CONTINUE
      C   FCBRG(2)=110
      C
      C ----- CHECK BID VALUES
      C
      C IF(QHAT.LT.1.OR.QHAT.GT.63) ENCD=-1
      C IF(OUTB.LT.1.OR.OUTB.GT.63) ENCD=-2
      C IF(BETA.LT.1.OR.BETA.GT.63) ENCD=-3
      C IF(ENCT.LT.1.OR.ENCT.GT.63) ENCD=-4
      C
      C ----- CHECK SCALAR VALUES
      C
      C IF(ISTB.LT.1.OR.ISTB.GT.127) ENCD=-5
      C IF(SID.LT.1.OR.SID.GT.127) ENCD=-6
      C IF(ISID.LT.1.OR.ISID.GT.127) ENCD=-7
      C
      C IF(ENCD.EQ.0) GO TO 150
      C CALL MPERR(ENCD)
      C RETURN
      C
      C CONTINUE
      C FCBRG(3)=ISTB
      C FCBRG(4)=OUTB
      C FCBRG(5)=QHAT
      C FCBRG(6)=BETA
      C FCBRG(7)=ENCT

```

PAGE 2

15-APR-88

12:48:04

/TR:BLOCKS/VR

FORTRAN IV-PLUS V02-51
ENCD.FTN0026 FCORG(9)=SID
0027 FCORG(11)=ISID

0028 C ENCD=RUNMP(FCORG(1),FCBSZ(1,5))

0029 C RETURN
0030 END

FORTRAM IV-PLUS V02-51
ENDC.FTN /TR:BLOCKS/VR

12:48:04 15-APR-88

PAGE 3

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	988478	156
3	SIDATA	888814	6
4	SVARS	888886	3
6	MP22Z	888312	181

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ENDC	1*2	1-888888									

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
BETA	1*2	F-888886*	ENCT	1*2	F-888818*	HRD	1*2	4-888882	HR1	R*4	6-8888384
I	1*2	4-888884	ISID	1*2	F-888816*	ISTB	1*2	F-888812*	LEVEL	1*2	6-8888318
QHAT	1*2	F-888882*	SID	1*2	F-888814*				OUTB	1*2	F-888884*

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
FCB	1*2	6-8888268	888814	6 (6)
FCBGR	1*2	6-8888888	888826	11 (11)
FCBSZ	1*2	6-888826	888232	77 (11.7)
MPDCB	1*2	6-8888274	888818	4 (4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
188	**	158	1-8888354				

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 881824 266

FORTAN IV-PLUS V82-51
ENCD.FTN /TR:BLOCKS/VR

C

.ENCD/NOSP=ENCD

12:48:15

15-APR-88

PAGE 4

```

FORTRAN IV-PLUS V02-51      12:48:21    15-APR-88      PAGE 1
TRBUF.FTN      /TR:BLOCKS/VP

C
0001      INTEGER FUNCTION TRBUF(DBUF1,DBUF2,INIT,CBUF)
C
C ----- HOST SUPPORT MODULE FOR DOUBLE BUFFER TO CIRC BUFF TRANSFER
C PROGRAM.
C ----- IOS SIMULATOR IN CSPU. (ON/OFF HOOK SIMULATION)
C
C      FCB=112
C
C ARGUMENTS:
C      DBUF1 = BID OF DOUBLE BUFFER 1
C      DBUF2 = BID OF DOUBLE BUFFER 2
C      INIT = INT. SCALAR ID FOR INIT. SELECT SWITCH
C      INIT+1 = INT. SCALAR ID FOR IOS MODE
C      # - OFF HOOK
C      1 - ON HOOK
C      CBUF = BID FOR CIRCULAR BUFFER
C
C      INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRI,LEVEL,RUNMP
C      INTEGER DBUF1,DBUF2,INIT,CBUF
C      COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
C
C      TRBUF=#
C      DO 100 I=1,11
C      FCBRG(I)=#
C      CONTINUE
C      FCBRG(2)=112
C
C ----- CHECK ARGUMENTS
C
C      IF(DBUF1.LT.1.OR.DBUF1.GT.63) TRBUF=-1
C      IF(DBUF2.LT.1.OR.DBUF2.GT.63) TRBUF=-2
C      IF(INIT.LT.1.OR.INIT.GT.127) TRBUF=-3
C      IF(CBUF.LT.1.OR.CBUF.GT.63) TRBUF=-4
C
C      IF(TRBUF.EQ.#) GO TO 150
C      CALL MPERR(TRBUF)
C      RETURN
C
C      CONTINUE
C      FCBRG(3)=INIT
C      FCBRG(4)=DBUF1
C      FCBRG(5)=DBUF2
C      FCBRG(7)=CBUF
C
C      TRBUF=RUNMP(FCBRG(1),FCBSZ(1,4))
C
C      RETURN
C      END

```

FORTRAM IV-PLUS V02-51
TRBUF.FTN /TR:8BLOCKS/VR

12:48:21 15-APR-88

PAGE 2

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	88326	187
3	SIDATA	88814	6
4	SVARS	88884	2
6	MPZZZ	88318	188

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
TRBUF	1*2	1-888888						

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CBUF	1*2	F-888818*	DBUF1	1*2	F-888882*	DBUF2	1*2	F-888884*
1	1*2	4-888882	INIT	1*2	F-888886*	LEVEL	1*2	6-888886
						HR1	1*2	6-888884
						HVS	1*2	4-888888

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
FCB	1*2	6-888268	88814	6 (6)
FCRG	1*2	6-888888	88826	11 (11)
FCBSZ	1*2	6-888826	88832	77 (11,7)
MPDCB	1*2	6-888274	88818	4 (4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
188	**	158	1-888252		

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 888656 215

NO FPP INSTRUCTIONS GENERATED

FORTAN IV-PLUS V#2-51
TRBUF.FTN /TR:BLOCKS/VR
PAGE 3

12:48:29 15-APR-88

C

.TRBUF/NOSP=TRBUF

```

FORTAN IV-PLUS V02-S1          12:48:36    15-APR-88          PAGE 1
DECD.FTN      /TR:BLOCKS/VR

0001      INTEGER FUNCTION DECD(ICB,ISL,OOB1,ITB1,OOB2,ITB2,DECT,DECB,IG)
C----- MOST SUPPORT MODULE FOR DECODING BIT STREAM INPUT FOR THE
C      PARC RECEIVER.
C
C      FCB=111
C
C      CALLING SEQUENCE:
C      M=DECD(ICB,ISL,OOB1,ITB1,OOB2,ITB2,DECT,DECB,IG)
C
C      WHERE:
C      ICB = INPUT BIT BUFFER, CIRCULAR, LENGTH=1024
C      ISL = FUNCTION LIST SELECTOR
C      ISL1 = 0 - INITIALIZE, SYNCHRONIZE
C      -1 - SYNCHRONIZE
C      +1 - DECODE
C      ISL2 = -1 - FUNC. LIST 1
C      +1 - FUNC. LIST 2
C      ISL3 = INT. SCAL ID FOR XMTR UPDATE
C      OOB1 = OUTPUT QHAT BUFFER 1, LENGTH=400
C      ITB1 = INT. SC. TABLE LOC. FOR T.BETA 1, LENGTH=2
C      OOB2 = OUTPUT QHAT BUFFER 2, LENGTH=400
C      ITB2 = INT. SC. TABLE LOC. FOR T.BETA 2, LENGTH=2
C      DECT = Q-LEVEL DECODE TABLE, LENGTH=256
C      DECB = BETA DECODE TABLE, LENGTH=128
C      IG = GAP SIZE IN INTEGER FORMAT
C
0002      INTEGER FCBRG,FCBSZ,HVS,FCB,MPDCB,HRI,LEVEL,RUNMP
0003      INTEGER ICB,ISL,OOB1,ITB1,OOB2,ITB2,DECT,DECB
0004      COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
C
C      DECD=0
C      DO 100 I=1,11
0005      FCBRG(I)=0
0006      CONTINUE
0007      FCBRG(2)=111
C
C----- CHECK BID VALUES
C
0008      IF(ICB.LT.1.OR.ICB.GT.63) DECD=-1
0009      IF(OOB1.LT.1.OR.OOB1.GT.63) DECD=-3
0010      IF(OOB2.LT.1.OR.OOB2.GT.63) DECD=-5
0011      IF(DECT.LT.1.OR.DECT.GT.63) DECD=-7
0012      IF(DECB.LT.1.OR.DECB.GT.63) DECD=-8
C
C----- CHECK SID VALUES
C
0013      IF(ISL.LT.1.OR.ISL.GT.127) DECD=-2
0014      IF(ITB1.LT.1.OR.ITB1.GT.127) DECD=-4
0015      IF(ITB2.LT.1.OR.ITB2.GT.127) DECD=-6
C
C----- CHECK GAP SIZE
C
0016      IF((IC.LT.1) OR (IG.GT.125)) DECD=-9
0017
0018

```

FORTAN IV-PLUS V02-51
 DECD.FTN /TR:BLOCKS/VR PAGE 2

12:48:36 15-APR-80

```

C
0019 IF(DECD.EQ.0) GO TO 150
0020 CALL MPERR(DECD)
0021 RETURN
C
0022 CONTINUE
0023 FCBRG(3)=ISL
0024 FCBRG(4)=OQB2
0025 FCBRG(5)=OQB1
0026 FCBRG(6)=ITB2
0027 FCBRG(7)=ITB1
0028 FCBRG(8)=DECT
0029 FCBRG(9)=DECB
0030 FCBRG(10)=IG
0031 FCBRG(11)=ICB
C
0032 DECD=RUNMP(FCBRG(1),FCBSZ(1,5))
C
0033 RETURN
0034 END

```

FORTAN IV-PLUS V02-S1
DECD.FTN /TR:BLOCKS/VR

12:48:36 15-APR-88

PAGE 3

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	SCODE1	177	RV.I.CON.LCL
3	SIDATA	6	RV.D.CON.LCL
4	SVARS	2	RV.D.CON.LCL
6	MPZZZ	188	RV.D.OVR.GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
DECD	I*2	1-000000									

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
DECB	I*2	F-000028*	DECT	I*2	F-000016*	HRI	I*2	6-000304	I	I*2	4-000002
ICB	I*2	F-000002*	IG	I*2	F-000022*	ISL	I*2	F-000000	ITB2	I*2	F-000014*
LEVEL	I*2	6-000306	OQB1	I*2	F-000006*	OQB2	I*2	F-000012*			

ARRAYS

NAME	TYPE	ADDRESS	SIZE	DIMENSIONS
FCB	I*2	6-000268	000014	6
FCBRG	I*2	6-000000	000026	11
FCBSZ	I*2	6-000026	000232	77
MPDCB	I*2	6-000274	000010	4

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	150	1-0000430		

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 001072 205

NO FPP INSTRUCTIONS GENERATED

FORTAN IV-PLUS VB2-51
DECD.FTN /TR:BLOCKS/VR
PAGE 4

12:48:47 15-APR-88

C

.DECD/NOSP-DECD

```

(00001) *
(00002) *
(00003) *
(00004) *
(00005) *
(00006) *
(00007) *
(00008) *
(00009) *
(00010) *
(00011) *
(00012) *
(00013) *
(00014) *
(00015) *
(00016) *
(00017) *
(00018) *
(00019) *
(00020) *
(00021) *
(00022) *
(00023) *
(00024) *
(00025) *
(00026) *
(00027) *
(00028) *
(00029) *
(00030) *
(00031) *
(00032) *
(00033) *
(00034) *
(00035) *
(00036) *
(00037) *
(00038) *
(00039) *
(00040) *
(00041) *

```

```

MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980
*****
* REAL TIME IMPLEMENTATION OF THE P A R C ALGORITHM *
* *****

```

```

PROGRAM: PARC.MAP
JAN. 30, 1980

```

```

MAVLIN YEH
ARVIND S ARORA

```

P A R C (PITCH EXTRACTED, ADAPTIVE, PREDICTIVE RESIDUAL ENCODER) IS A SPEECH COMPRESSION ALGORITHM DEVELOPED AT THE DEPARTMENT OF ELECTRICAL ENGINEERING, UNIVERSITY OF NOTRE DAME UNDER CONTRACT FROM THE DEFENCE COMMUNICATION AGENCY. THE ALGORITHM HAS BEEN CURRENTLY DEVELOPED AND OPTIMIZED FOR DIGITAL TRANSMISSION OF SPEECH AT 9600 BAUD.

THE REAL TIME IMPLEMENTATION OF P A R C HAS BEEN DONE ON THE MAP-300 ARRAY PROCESSOR. SEVEN OF THE EIGHT PROGRAM MODULES USED IN THE IMPLEMENTATION ARE CONTAINED IN THIS FILE:

1. INITIALIZATION AND UPDATING.
2. ESTIMATION OF THE PITCH PERIOD AND THE CORRELATION COEFFICIENT REQUIRED FOR PITCH REDUNDANCY REMOVAL (PRR).
3. PARC-TRANSMITTER.
4. PARC-RECEIVER.
5. DOUBLE-BUFFER (189*2) TO CIRCULAR-BUFFER (1024) TRANSFER PROGRAM. THIS IS USED TO SIMULATE THE FUNCTION OF THE IOS2.
6. THE SOURCE ENCODER PREPARES THE DIGITAL BIT STREAM BY ENCODING THE QUANTIZER LEVELS AND THE PRR SIDE INFORMATION PUT OUT BY THE PARC TRANSMITTER. IT USES A VARIABLE-LENGTH TO VARIABLE-LENGTH ENCODING SCHEME.
7. THE DECODER PERFORMS TWO FUNCTIONS. IT FIRST ACQUIRES FRAME SYNCHRONIZATION ON THE INCOMING BIT STREAM. IT THEN DECODES THE BIT STREAM TO THE QUANTIZER LEVELS AND THE PRR INFORMATION FOR USE BY THE PARC RECEIVER.

EJECT

PAGE 3: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

(000000) EJECT

PAGE 7: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

B483F F05405B3 (00206)      MOVHR      R5,ISVT1S(R2)
B4841 4A4A (00207)          ANDRR      R4,R5
B4842 E040A21 (00208)      MOVHR      R4,DASTX
B4844 F040A25 (00209)      MOVHR      R4,VHSTX
B4846 4C46 (00210)          ADDR      R4,R3
B4847 4A4A (00211)          ANDRR      R4,R5
B4848 E040A25 (00212)      MOVHR      R4,VHSTX
B484A F040A1D (00213)      MOVHR      R4,SSTX
B484C 4C46 (00214)          ADDR      R4,R3
B484D F061FFCE (00215)      MOVHR      R6,SYSSFLGS
B484F 5A6C0004 (00216)      ANDKR      R6,R6,S4
B4851 00104958 (00217)      JMP          CONT2S,EQ
B4853 F00DFCE (00218)      MOVLM      $6,SYSSFLGS
B4855 F0640504 (00219)      MOVHR      R6,ISVT2S(R2)
B4857 4C4C (00220)          ADDR      R4,R6
B4858 4A4A (00221)          ANDRR      R4,R5
B4859 E040A1D (00222)      MOVHR      R4,SSTX
B485B 0E70 (00223)          RETURN
                                EVEN
                                EJECT
                                (00224)
                                (00225)
                                (00226)

```

;MASK FOR SAMPLE BUFFER
 ;
 ;OUTPUT TRANSFERING POINTER
 ;
 ;ADD N OF TX
 ;MASKING
 ;OUTPUT VBAT POINTER
 ;
 ;ADD N OF TX
 ;GET SYSTEM FLAG
 ;CHECK G2
 ;JUMP IF G2=0
 ;RESET G2
 ;GET PITCH REPETITION SIZE
 ;ADD PITCH REPETITION SIZE
 ;MASKING
 ;OUTPUT S POINTER

MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

PAGE

INIS MODULE TO INITIALIZE APS PARAMETERS.
 IT INITIALIZES THE FOLLOWING PARAMETERS:
 PITCH REPETITION THRESHOLD, FILTER THRESHOLD, FILTERING SIZE, ISTART,
 CORRELATION SIZE, PITCH REPETITION SIZE, NSTX, SSTX, DASTX, VHSTX
 INTEGER SCALAR SEQUENCE: PITCH REPETITION THRESHOLD, FILTER THRESHOLD,
 FILTER SIZE - 1, ISTART - 1, KBLK - 1, PITCH REPETITION SIZE
 FCB FORMAT (16 BIT WORD FORMAT SHOWN)

WORD	LEFT BYTE	RIGHT BYTE
1	115	INTEGER SCALAR
2	B	B
3	B	B
4	B	B
5	B	B

MOVW R2,R1,MSKSRBYT ;GET INTEGER SCALAR
 MOVW R3,ISVTS(R2) ;PITCH REPETITION THRESHOLD

00227) *
 00228) *
 00229) *
 00230) *
 00231) *
 00232) *
 00233) *
 00234) *
 00235) *
 00236) *
 00237) *
 00238) *
 00239) *
 00240) *
 00241) *
 00242) *
 00243) *
 00244) *
 00245) *
 00246) *
 00247) *
 00248) *
 00249) *
 00250) *
 00251) *
 00252) *
 00253) *
 00254) *
 00255) *
 00256) *
 00257) *
 00258) *
 00259) *
 00260) *
 00261) *
 00262) *
 00263) *
 00264) *
 00265) *
 00266) *
 00267) *
 00268) *
 00269) *
 00270) *
 00271) *
 00272) *

00273) *
 00274) *
 00275) *
 00276) *
 00277) *

PAGE 18: MAP MODULES FOR THE P A R C ALGORITHM ---- JAN. 38, 1988

RCINIS MODULE TO INITIALIZE APS PARAMETERS OF RECEIVER.
 IT INITIALIZES THE FOLLOWING PARAMETERS:
 NSRC, VHSRC, SHSRC, DASRC AND G3
 INTEGER SCALAR SEQUENCE: STARTING LOCATION OF RECEIVER BUFFER
 TRANSFER BLOCK SIZE
 FCB FORMAT (16 BIT WORD FORMAT SHOWN)

WORD	LEFT BYTE	RIGHT BYTE
1	116	INTEGER SCALAR #
2	#	#
3	#	#
4	#	#
5	#	#

RCINIS
 MOVW R2,R1,MSKSRBYT
 MOVW R3,ISVTS(R2)
 MOVW R3,VHSRC

RCINIS
 MOVW R2,R1,MSKSRBYT
 MOVW R3,ISVTS(R2)
 MOVW R3,VHSRC

RCINIS
 MOVW R2,R1,MSKSRBYT
 MOVW R3,ISVTS(R2)
 MOVW R3,VHSRC

RCINIS
 MOVW R2,R1,MSKSRBYT
 MOVW R3,ISVTS(R2)
 MOVW R3,VHSRC

PAGE 11: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

04896 E0304FF (00349)	MOVW	R3, SHSRC	
04898 F0440504 (00350)	MOVW	R4, ISVT2S(R2)	
0489A 4E38 (00351)	SUBRR	R3, R4	
0489B E0304ED3 (00352)	MOVW	R3, DASRC	
0489D F00FFCE (00353)	MOVLW	\$7, SYSSFLGS	
0489F 4E36 (00354)	SUBRR	R3, R3	
048A1 E030466 (00355)	MOVW	R3, NSRC	
048A2 0E70 (00356)	RETURN		
048A3 0000 (00357)	EVEN		
000048A4 (00358)			
000048A4 (00359)	CONIS = 0L		
000048A4 (00360)			
000048A4 (00361)	EJECT		

TRANSFER SIZE
RESET G3

PAGE 12:

MAP MODULES FOR THE P A R C ALGORITHM

--- JAN. 30, 1980

```

(00362) *
(00363) *
(00364) *
(00365) *
(00366) *
(00367) *
(00368) *
(00369) *
(00370) *
(00371) *
(00372) *
(00373) *
(00374) *
(00375) *
(00376) *
(00377) *
(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) *
*****
*
*      ADAPTIVE FILTERS AND
*      PITCH PERIOD AND CORRELATION CALCULATION
*
*****
IT USES V3.5 MAP-EXECUTIVE

* DESCRIPTION:
THIS PROGRAM CALCULATES THE PITCH PERIOD AND CORRELATION COEFFICIENT
OF A BLOCK SAMPLES. ALSO, DEPENDING ON THE CONDITION OF THE SAMPLE
BUFFER IN TRANSMITTER, IT WILL EMPLOY ADAPTIVE FILTER AND/OR PITCH
REPETITION.

* THIS PROGRAM EMPLOYS THE FOLLOWING BUFFERS:

1. ADAM BUFFER: ADAM SAMPLES INPUT ANALOG SIGNALS AND PUTS THE
CORRESPONDING QUANTIZED DIGITAL SIGNALS INTO THIS BUFFER.
--- DOUBLE BUFFERING --- SHORT FLOATING FORMAT ---
2. SAMPLE BUFFER: THE INPUT DATA COME FROM ADAM BUFFER. THE PURPOSE
OF THIS BUFFER IS TO PROVIDE BUFFER CONTROL TECHNIQUE.
--- CIRCULAR BUFFER --- SHORT FLOATING FORMAT ---
3. WHAT BUFFER: THIS BUFFER CONTAINS RECONSTRUCTED REDUCED SPEECH
SAMPLES FROM RARC-TRANSMITTER. ( THIS PROGRAM ONLY DEFINES THE
STARTING LOCATION OF WHAT POINTER.)

* SYMBOLIC DEFINITIONS:

TBLK: TRANSFERING BLOCK SIZE FROM ADAM BUFFER TO SAMPLE BUFFER
FBLK: FILTERING BLOCK SIZE
KBLK: CORRELATION BLOCK SIZE
IEND: UPPER LIMIT OF PITCH PERIOD
SSTX: DATA POINTER IN SAMPLE BUFFER FOR USING IN PARC-TRANSMITTER
DASTX: DATA POINTER IN SAMPLE BUFFER FOR THE TRANSFER FROM ADAM
      BUFFER
VHSTX: WHAT POINTER IN WHAT BUFFER
INSGTH: LOCATION OF PITCH REPETITION THRESHOLD
INSFTH: LOCATION OF FILTER THRESHOLD
INSFBS: LOCATION OF FBLK-1
INCSST: LOCATION OF LOWER LIMIT OF PITCH PERIOD
INCSZ: LOCATION OF CORRELATION BLOCK SIZE -1
INSGSZ: LOCATION OF PITCH REPETITION SIZE

```

PAGE 13: MAP MODULE FOR THE P A R C ALGORITHM --- JAN. 38, 1988

* RESTRICTIONS:

1. TBLK SHOULD BE MULTIPLE OF 2. THIS RESTRICTION CAN BE REMOVED IF THOSE STATEMENTS MARKED **A** ARE SLIGHTLY MODIFIED. THE PURPOSE OF THIS RESTRICTION IS TO REDUCE THE DATA TRANSFERING TIME.
2. IT IS ASSUMED THAT DASTX NEVER OVERRIDES SSTX. THAT IS CONTROLLED BY 3. BY PITCH REPETITION THRESHOLD AND SIZE.
3. KBLK SHOULD BE MULTIPLE OF 4.
4. THE ABSOLUTE VALUES OF UPPER LIMIT AND LOWER LIMIT OF CORRELATION COEFFICIENT SHOULD BE THE SAME.
5. THE SEQUENCE OF INPUT SCALARS ARE AS FOLLOWS:
TBLK/2-A.B.FBLF-B.5.KBLK-B.5.SEARCHING SIZE-B.5.2*-15,
IEND-B.5.B.8881,1START,LB,UB,HL/(UB*2**15),HL/(UB*2**24),
2**24*UB/HL
6. THE LOCATIONS OF ARRAYS ARE AS FOLLOWS:
SAMPLE BUFFER: B(3) ... 1823(3)
WHAT BUFFER: B(2) ... 1823(2) --- THE SIZE AND STARTING LOCATION OF THIS BUFFER CAN BE CHANGED BY SLIGHTLY MODIFICATIONS IN TRANSMITTER PROGRAMS.

* CALL THIS FUNCTION:

MAP = PICH3(SA,U)
WHERE SA --- SCALAR * FOR PITCH PERIOD
U --- BUFFER * FOR ADAM BUFFER

CONTAINS FOR ASSEMBLY

```

MSS=B
DHYS=$794
BUSIS=$8881
TOESPRT=$288
CSPUSNOS=$21FC
AFDTS=$8E8

SVT588=SVTS+188
SVT518=SVT588+2
SVT548=SVT518+6
SVT568=SVT548+4
SVT638=SVT568+18
SVT688=SVT638+18
SVT788=SVT688+4
SVT748=SVT788+8

DUMMY OUTPUT SPACE
:BUS 1
:TOP OF EXEC.
:LOCATION OF NO-CSPU SUPPORT
:STARTING ADDRESS OF AFDT
:SCALAR TABLE 58

:ADD TWO SCALAR

```

PAGE 14: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

00000426 (00450) SVT79S=SVT74S+12 ;ADD ONE SCALAR
0000042E (00451) SVT83S=SVT79S+8
00000430 (00452) SVT98S=SVT83S+24
00000432 (00453) SVT99S=SVT98S+4 ;ADD UPSAMPLING
00000434 (00454) SVT111S=SVT99S+26 ;ADD ONE SCALAR
00000436 (00455) NSTX=SVT83S+2
00000438 (00456) NSRC=SVT111S+2
0000043A (00457) SVT115S=NSRC+2
0000043C (00458) SVT117S=SVT115S+4
0000043E (00459) SVT118S=SVT117S+2
00000440 (00460) DISPATCH TABLE ENTRY
00000442 (00461) *
00000444 (00462) *
00000446 (00463) *
00000448 (00464) *
0000044A (00465) ADDR PICH3S(R7,1) ;APU=PICH3
0000044C (00466) ADDR VPICH3S(R7,1) ;APS=VPICH3
0000044E (00467) ADDR CSPUSNOS(1,0) ;NO CPU
00000450 (00468) *
00000452 (00469) *
00000454 (00470) *
00000456 (00471) *
00000458 (00472) *
0000045A (00473) *
0000045C (00474) *
0000045E (00475) *
00000460 (00476) *
00000462 (00477) *
00000464 (00478) *
00000466 (00479) PICH3S BEGIN APU(PICH3)
00000468 (00480) * #M=3 ;STARTING ADDRESS
0000046A (00481) * #A=B ;MODULE SIZE
0000046C (00482) * DATA TRANSFER FROM ADAM BUFFER TO SAMPLE BUFFER
0000046E (00483) *
00000470 (00484) *
00000472 (00485) *
00000474 (00486) *
00000476 (00487) *
00000478 (00488) *
0000047A (00489) *
0000047C (00490) *
0000047E (00491) *
00000480 (00492) *
00000482 (00493) PICH3SSA K(1)
00000484 (00494) *
00000486 (00495) *
00000488 (00496) *
0000048A (00497) *
0000048C (00498) *
0000048E (00499) *
00000490 (00500) *
00000492 (00501) *
00000494 (00502) *
00000496 (00503) *
00000498 (00504) *
0000049A (00505) *
0000049C (00506) *
0000049E (00507) *
00000500 (00508) *
00000502 (00509) *
00000504 (00510) *
00000506 (00511) *
00000508 (00512) *
0000050A (00513) *
0000050C (00514) *
0000050E (00515) *
00000510 (00516) *
00000512 (00517) *
00000514 (00518) *
00000516 (00519) *
00000518 (00520) *
0000051A (00521) *
0000051C (00522) *
0000051E (00523) *
00000520 (00524) *
00000522 (00525) *
00000524 (00526) *
00000526 (00527) *
00000528 (00528) *
0000052A (00529) *
0000052C (00530) *
0000052E (00531) *
00000530 (00532) *
00000532 (00533) *
00000534 (00534) *
00000536 (00535) *
00000538 (00536) *
0000053A (00537) *
0000053C (00538) *
0000053E (00539) *
00000540 (00540) *
00000542 (00541) *
00000544 (00542) *
00000546 (00543) *
00000548 (00544) *
0000054A (00545) *
0000054C (00546) *
0000054E (00547) *
00000550 (00548) *
00000552 (00549) *
00000554 (00550) *
00000556 (00551) *
00000558 (00552) *
0000055A (00553) *
0000055C (00554) *
0000055E (00555) *
00000560 (00556) *
00000562 (00557) *
00000564 (00558) *
00000566 (00559) *
00000568 (00560) *
0000056A (00561) *
0000056C (00562) *
0000056E (00563) *
00000570 (00564) *
00000572 (00565) *
00000574 (00566) *
00000576 (00567) *
00000578 (00568) *
0000057A (00569) *
0000057C (00570) *
0000057E (00571) *
00000580 (00572) *
00000582 (00573) *
00000584 (00574) *
00000586 (00575) *
00000588 (00576) *
0000058A (00577) *
0000058C (00578) *
0000058E (00579) *
00000590 (00580) *
00000592 (00581) *
00000594 (00582) *
00000596 (00583) *
00000598 (00584) *
0000059A (00585) *
0000059C (00586) *
0000059E (00587) *
00000600 (00588) *
00000602 (00589) *
00000604 (00590) *
00000606 (00591) *
00000608 (00592) *
0000060A (00593) *
0000060C (00594) *
0000060E (00595) *
00000610 (00596) *
00000612 (00597) *
00000614 (00598) *
00000616 (00599) *
00000618 (00600) *
0000061A (00601) *
0000061C (00602) *
0000061E (00603) *
00000620 (00604) *
00000622 (00605) *
00000624 (00606) *
00000626 (00607) *
00000628 (00608) *
0000062A (00609) *
0000062C (00610) *
0000062E (00611) *
00000630 (00612) *
00000632 (00613) *
00000634 (00614) *
00000636 (00615) *
00000638 (00616) *
0000063A (00617) *
0000063C (00618) *
0000063E (00619) *
00000640 (00620) *
00000642 (00621) *
00000644 (00622) *
00000646 (00623) *
00000648 (00624) *
0000064A (00625) *
0000064C (00626) *
0000064E (00627) *
00000650 (00628) *
00000652 (00629) *
00000654 (00630) *
00000656 (00631) *
00000658 (00632) *
0000065A (00633) *
0000065C (00634) *
0000065E (00635) *
00000660 (00636) *
00000662 (00637) *
00000664 (00638) *
00000666 (00639) *
00000668 (00640) *
0000066A (00641) *
0000066C (00642) *
0000066E (00643) *
00000670 (00644) *
00000672 (00645) *
00000674 (00646) *
00000676 (00647) *
00000678 (00648) *
0000067A (00649) *
0000067C (00650) *
0000067E (00651) *
00000680 (00652) *
00000682 (00653) *
00000684 (00654) *
00000686 (00655) *
00000688 (00656) *
0000068A (00657) *
0000068C (00658) *
0000068E (00659) *
00000690 (00660) *
00000692 (00661) *
00000694 (00662) *
00000696 (00663) *
00000698 (00664) *
0000069A (00665) *
0000069C (00666) *
0000069E (00667) *
00000700 (00668) *
00000702 (00669) *
00000704 (00670) *
00000706 (00671) *
00000708 (00672) *
0000070A (00673) *
0000070C (00674) *
0000070E (00675) *
00000710 (00676) *
00000712 (00677) *
00000714 (00678) *
00000716 (00679) *
00000718 (00680) *
0000071A (00681) *
0000071C (00682) *
0000071E (00683) *
00000720 (00684) *
00000722 (00685) *
00000724 (00686) *
00000726 (00687) *
00000728 (00688) *
0000072A (00689) *
0000072C (00690) *
0000072E (00691) *
00000730 (00692) *
00000732 (00693) *
00000734 (00694) *
00000736 (00695) *
00000738 (00696) *
0000073A (00697) *
0000073C (00698) *
0000073E (00699) *
00000740 (00700) *
00000742 (00701) *
00000744 (00702) *
00000746 (00703) *
00000748 (00704) *
0000074A (00705) *
0000074C (00706) *
0000074E (00707) *
00000750 (00708) *
00000752 (00709) *
00000754 (00710) *
00000756 (00711) *
00000758 (00712) *
0000075A (00713) *
0000075C (00714) *
0000075E (00715) *
00000760 (00716) *
00000762 (00717) *
00000764 (00718) *
00000766 (00719) *
00000768 (00720) *
0000076A (00721) *
0000076C (00722) *
0000076E (00723) *
00000770 (00724) *
00000772 (00725) *
00000774 (00726) *
00000776 (00727) *
00000778 (00728) *
0000077A (00729) *
0000077C (00730) *
0000077E (00731) *
00000780 (00732) *
00000782 (00733) *
00000784 (00734) *
00000786 (00735) *
00000788 (00736) *
0000078A (00737) *
0000078C (00738) *
0000078E (00739) *
00000790 (00740) *
00000792 (00741) *
00000794 (00742) *
00000796 (00743) *
00000798 (00744) *
0000079A (00745) *
0000079C (00746) *
0000079E (00747) *
00000800 (00748) *
00000802 (00749) *
00000804 (00750) *
00000806 (00751) *
00000808 (00752) *
0000080A (00753) *
0000080C (00754) *
0000080E (00755) *
00000810 (00756) *
00000812 (00757) *
00000814 (00758) *
00000816 (00759) *
00000818 (00760) *
0000081A (00761) *
0000081C (00762) *
0000081E (00763) *
00000820 (00764) *
00000822 (00765) *
00000824 (00766) *
00000826 (00767) *
00000828 (00768) *
0000082A (00769) *
0000082C (00770) *
0000082E (00771) *
00000830 (00772) *
00000832 (00773) *
00000834 (00774) *
00000836 (00775) *
00000838 (00776) *
0000083A (00777) *
0000083C (00778) *
0000083E (00779) *
00000840 (00780) *
00000842 (00781) *
00000844 (00782) *
00000846 (00783) *
00000848 (00784) *
0000084A (00785) *
0000084C (00786) *
0000084E (00787) *
00000850 (00788) *
00000852 (00789) *
00000854 (00790) *
00000856 (00791) *
00000858 (00792) *
0000085A (00793) *
0000085C (00794) *
0000085E (00795) *
00000860 (00796) *
00000862 (00797) *
00000864 (00798) *
00000866 (00799) *
00000868 (00800) *
0000086A (00801) *
0000086C (00802) *
0000086E (00803) *
00000870 (00804) *
00000872 (00805) *
00000874 (00806) *
00000876 (00807) *
00000878 (00808) *
0000087A (00809) *
0000087C (00810) *
0000087E (00811) *
00000880 (00812) *
00000882 (00813) *
00000884 (00814) *
00000886 (00815) *
00000888 (00816) *
0000088A (00817) *
0000088C (00818) *
0000088E (00819) *
00000890 (00820) *
00000892 (00821) *
00000894 (00822) *
00000896 (00823) *
00000898 (00824) *
0000089A (00825) *
0000089C (00826) *
0000089E (00827) *
00000900 (00828) *
00000902 (00829) *
00000904 (00830) *
00000906 (00831) *
00000908 (00832) *
0000090A (00833) *
0000090C (00834) *
0000090E (00835) *
00000910 (00836) *
00000912 (00837) *
00000914 (00838) *
00000916 (00839) *
00000918 (00840) *
0000091A (00841) *
0000091C (00842) *
0000091E (00843) *
00000920 (00844) *
00000922 (00845) *
00000924 (00846) *
00000926 (00847) *
00000928 (00848) *
0000092A (00849) *
0000092C (00850) *
0000092E (00851) *
00000930 (00852) *
00000932 (00853) *
00000934 (00854) *
00000936 (00855) *
00000938 (00856) *
0000093A (00857) *
0000093C (00858) *
0000093E (00859) *
00000940 (00860) *
00000942 (00861) *
00000944 (00862) *
00000946 (00863) *
00000948 (00864) *
0000094A (00865) *
0000094C (00866) *
0000094E (00867) *
00000950 (00868) *
00000952 (00869) *
00000954 (00870) *
00000956 (00871) *
00000958 (00872) *
0000095A (00873) *
0000095C (00874) *
0000095E (00875) *
00000960 (00876) *
00000962 (00877) *
00000964 (00878) *
00000966 (00879) *
00000968 (00880) *
0000096A (00881) *
0000096C (00882) *
0000096E (00883) *
00000970 (00884) *
00000972 (00885) *
00000974 (00886) *
00000976 (00887) *
00000978 (00888) *
0000097A (00889) *
0000097C (00890) *
0000097E (00891) *
00000980 (00892) *
00000982 (00893) *
00000984 (00894) *
00000986 (00895) *
00000988 (00896) *
0000098A (00897) *
0000098C (00898) *
0000098E (00899) *
00000990 (00900) *
00000992 (00901) *
00000994 (00902) *
00000996 (00903) *
00000998 (00904) *
0000099A (00905) *
0000099C (00906) *
0000099E (00907) *
00000A00 (00908) *
00000A02 (00909) *
00000A04 (00910) *
00000A06 (00911) *
00000A08 (00912) *
00000A0A (00913) *
00000A0C (00914) *
00000A0E (00915) *
00000A10 (00916) *
00000A12 (00917) *
00000A14 (00918) *
00000A16 (00919) *
00000A18 (00920) *
00000A1A (00921) *
00000A1C (00922) *
00000A1E (00923) *
00000A20 (00924) *
00000A22 (00925) *
00000A24 (00926) *
00000A26 (00927) *
00000A28 (00928) *
00000A2A (00929) *
00000A2C (00930) *
00000A2E (00931) *
00000A30 (00932) *
00000A32 (00933) *
00000A34 (00934) *
00000A36 (00935) *
00000A38 (00936) *
00000A3A (00937) *
00000A3C (00938) *
00000A3E (00939) *
00000A40 (00940) *
00000A42 (00941) *
00000A44 (00942) *
00000A46 (00943) *
00000A48 (00944) *
00000A4A (00945) *
00000A4C (00946) *
00000A4E (00947) *
00000A50 (00948) *
00000A52 (00949) *
00000A54 (00950) *
00000A56 (00951) *
00000A58 (00952) *
00000A5A (00953) *
00000A5C (00954) *
00000A5E (00955) *
00000A60 (00956) *
00000A62 (00957) *
00000A64 (00958) *
00000A66 (00959) *
00000A68 (00960) *
00000A6A (00961) *
00000A6C (00962) *
00000A6E (00963) *
00000A70 (00964) *
00000A72 (00965) *
00000A74 (00966) *
00000A76 (00967) *
00000A78 (00968) *
00000A7A (00969) *
00000A7C (00970) *
00000A7E (00971) *
00000A80 (00972) *
00000A82 (00973) *
00000A84 (00974) *
00000A86 (00975) *
00000A88 (00976) *
00000A8A (00977) *
00000A8C (00978) *
00000A8E (00979) *
00000A90 (00980) *
00000A92 (00981) *
00000A94 (00982) *
00000A96 (00983) *
00000A98 (00984) *
00000A9A (00985) *
00000A9C (00986) *
00000A9E (00987) *
00000AA0 (00988) *
00000AA2 (00989) *
00000AA4 (00990) *
00000AA6 (00991) *
00000AA8 (00992) *
00000AAA (00993) *
00000AAC (00994) *
00000AAE (00995) *
00000AB0 (00996) *
00000AB2 (00997) *
00000AB4 (00998) *
00000AB6 (00999) *
00000AB8 (01000) *
00000ABA (01001) *
00000ABC (01002) *
00000ABE (01003) *
00000AC0 (01004) *
00000AC2 (01005) *
00000AC4 (01006) *
00000AC6 (01007) *
00000AC8 (01008) *
00000ACA (01009) *
00000ACC (01010) *
00000ACE (01011) *
00000ACF (01012) *
00000AD0 (01013) *
00000AD2 (01014) *
00000AD4 (01015) *
00000AD6 (01016) *
00000AD8 (01017) *
00000ADA (01018) *
00000ADC (01019) *
00000ADE (01020) *
00000ADF (01021) *
00000AE0 (01022) *
00000AE2 (01023) *
00000AE4 (01024) *
00000AE6 (01025) *
00000AE8 (01026) *
00000AEA (01027) *
00000AEC (01028) *
00000AEE (01029) *
00000AEF (01030) *
00000AF0 (01031) *
00000AF2 (01032) *
00000AF4 (01033) *
00000AF6 (01034) *
00000AF8 (01035) *
00000AFA (01036) *
00000AFC (01037) *
00000AFE (01038) *
00000AFF (01039) *
00000B00 (01040) *
00000B02 (01041) *
00000B04 (01042) *
00000B06 (01043) *
00000B08 (01044) *
00000B0A (01045) *
00000B0C (01046) *
00000B0E (01047) *
00000B10 (01048) *
00000B12 (01049) *
00000B14 (01050) *
00000B16 (01051) *
00000B18 (01052) *
00000B1A (01053) *
00000B1C (01054) *
00000B1E (01055) *
00000B20 (01056) *
00000B22 (01057) *
00000B24 (01058) *
00000B26 (01059) *
00000B28 (01060) *
00000B2A (01061) *
00000B2C (01062) *
00000B2E (01063) *
00000B30 (01064) *
00000B32 (01065) *
00000B34 (01066) *
00000B36 (01067) *
00000B38 (01068) *
00000B3A (01069) *
00000B3C (01070) *
00000B3E (01071) *
00000B40 (01072) *
00000B42 (01073) *
00000B44 (01074) *
00000B46 (01075) *
00000B48 (01076) *
00000B4A (01077) *
00000B4C (01078) *
00000B4E (01079) *
00000B50 (01080) *
00000B52 (01081) *
00000B54 (01082) *
00000B56 (01083) *
00000B58 (01084) *
00000B5A (01085) *
00000B5C (01086) *
00000B5E (01087) *
00000B60 (01088) *
00000B62 (01089) *
00000B64 (01090) *
00000B66 (01091) *
00000B68 (01092) *
00000B6A (01093) *
00000B6C (01094) *
00000B6E (01095) *
00000B70 (01096) *
00000B72 (01097) *
00000B74 (01098) *
00000B76 (01099) *
00000B78 (01100) *
00000B7A (01101) *
00000B7C (01102) *
00000B7E (01103) *
00000B80 (01104) *
00000B82 (01105) *
00000B84 (01106) *
00000B86 (01107) *
00000B88 (01108) *
00000B8A (01109) *
00000B8C (01110) *
00000B8E (01111) *
00000B90 (01112) *
00000B92 (01113) *
00000B94 (01114) *
00000B96 (01115) *
00000B98 (01116) *
00000B9A (01117) *
00000B9C (01118) *
00000B9E (01119) *
00000BA0 (01120) *
00000BA2 (01121) *
00000BA4 (01122) *
00000BA6 (01123) *
00000BA8 (01124) *
00000BAA (01125) *
00000BAC (01126) *
00000BAE (01127) *
00000BB0 (01128) *
00000BB2 (01129) *
00000BB4 (01130) *
00000BB6 (01131) *
00000BB8 (01132) *
00000BBA (01133) *
00000BBC (01134) *
00000BBE (01135) *
00000BC0 (01136) *
00000BC2 (01137) *
00000BC4 (01138) *
00000BC6 (01139) *
00000BC8 (01140) *
00000BCA (01141) *
00000BCC (01142) *
00000BCE (01143) *
00000BCF (01144) *
00000BD0 (01145) *
00000BD2 (01146) *
00000BD4 (01147) *
00000BD6 (01148) *
00000BD8 (01149) *
00000BDA (01150) *
00000BDC (01151) *
00000BDE (01152) *
00000BDF (01153) *
00000BE0 (01154) *
00000BE2 (01155) *
00000BE4 (01156) *
00000BE6 (01157) *
00000BE8 (01158) *
00000BEA (01159) *
00000BEC (01160) *
00000BEE (01161) *
00000BEF (01162) *
00000BF0 (01163) *
00000BF2 (01164) *
00000BF4 (01165) *
00000BF
```

PAGE 15: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

```

A01 B48A0 B0F00000 (B0494)
A02 B48A0 B0E00000 (B0495)
A03 B48AC B0E90000 (B0496)
A04 B48AE B0970000 (B0497)
A05 B48B0 B0E00000 (B0498)
A06 B48B2 B0E00000 (B0499)
A07 B48B4 B0E00000 (B0500)
A08 B48B6 B0E10000 (B0501)
A09 B48B8 B0E3A000 (B0502)
A0A B48BA B0910000 (B0503)
A0B B48BC B0E32000 (B0504)
A0C B48BE B0E00000 (B0505)
A0D B48C0 B0E00000 (B0506)
A0E B48C2 B0E00000 (B0507)
A0F B48C4 B0E00000 (B0508)
A10 B48C6 B0E00000 (B0509)
A11 B48C8 B0E00000 (B0510)
A12 B48CA B0E10000 (B0511)
      *
      * (B0512) *
      * (B0513) *
      * (B0514) *
      * (B0515) *
      * (B0516) *
      * (B0517) *
      * (B0518) *
      * (B0519) *
      * (B0520) *
      * (B0521) *
      * (B0522) *
      * (B0523) *
      * (B0524) *
      * (B0525) *
      * (B0526) *
      * (B0527) *
      * (B0528) *
      * (B0529) *
      * (B0530) *
      * (B0531) *
      * (B0532) *
      * (B0533) *
      * (B0534) *
      * (B0535) *
      * (B0536) *
      * (B0537) *

A13 B48CC B0E00000 (B0529)
A14 B48CE B0E00000 (B0530)
A15 B48D0 B0E00000 (B0531)
A16 B48D2 B0E00000 (B0532)
A17 B48D4 B0E00000 (B0533)
A18 B48D6 B0E00000 (B0534)
A19 B48D8 B0E00000 (B0535)
A1A B48DA B0E00000 (B0536)
A1B B48DC B0E00000 (B0537)

      MOV(IA0,AB) \ NOP
      MOV(IA0,MB)
      MOV(IA0,M1)
      MOV(R,A7)
      MOV(IA0,M7) \ NOP
      NOP \ MOV(IA0,M7)
      MUL(MB,M7)
      MOV(P,A1)
      ALIGN(A1)
      MOV(R,A1)
      NORM(A1)
      MOV(R,M7)
      MUL(M1,M7)
      SUB(AB,A7) \ NOP
      MOV(R,AB) \ NOP
      MOV(P,OQ) \ NOP
      NOP \ MOV(P,OQ)
      JUMPC(LOOP2,T1)

      ADAPTIVE FILTERS

      REGISTERS AFFECTED:

      ADM1: AB,A1,A2,MB,M6,M7,EXO
      ADM2: AB,A1,MB,M6,EXO

      FLAGS AFFECTED: AF1
      FLAGS SENSED: GB,AF1

      INPUT SEQUENCE: A,S(K-1),B,FBLK,S(K),SF(K-1),S(K),SF(K-1),.....,DUMMY.
      OUTPUT SEQUENCE: SF(K),...

      IT EMPLOYES IN PLACE OPERATION.

      NOP
      JUMPS(PERIOD,GB)
      JUMPC(AWAIT,AF1)
      CLEAR(AF1)
      MOV(IA0,MB) \ NOP
      MOV(IA0,M7) \ NOP
      MUL(MB,M7) \ MOV(IA0,MB)
      MOV(IA0,A2) \ NOP
      MOV(IA0,M6) \ NOP

      ; NO FILTER IS EMPLOYED
      ; WAIT FOR SIGNAL FROM APS
      ; MB=A
      ; M7=S(K-1)
      ; A2=FILTERING BLOCK SIZE
      ; M6=S(K)

```



```

(00582) *
(00583) *
A2E 04902 20492049
A2F 04904 20482040
A30 04906 00F10000 LOOP6
A31 04908 000000F1 (00587)
A32 0490A 00F00000 (00588)
A33 0490C 000000F0 (00589)
(00590) *
A34 0490E 49004900
A35 04910 12851285 (00592)
A36 04912 00F10000 (00593)
A37 04914 000000F1 (00594)
A38 04916 46854685 (00595)
A39 04918 00F00000 (00596)
A3A 0491A 000000F0 (00597)
(00598) *
A3B 0491C 49164916 (00599)
A3C 0491E 12851285 (00600)
A3D 04920 00F10000 (00601)
A3E 04922 000000F1 (00602)
A3F 04924 46854685 (00603)
A40 04926 00F00000 (00604)
A41 04928 000000F0 (00605)
A42 0492A 4F760896 (00606)
A43 0492C 00930898 (00607)
A44 0492E 901E0034 (00608)
(00609) *
A45 04930 00554F00 (00610)
A46 04932 46A00850 (00611)
A47 04934 9009004E (00612)
(00613) *
A48 04936 20292029 (00614)
A49 04938 00530894 (00615)
A4A 0493A 00920892 (00616)
A4B 0493C 00150815 (00617)
A4C 0493E 00160816 (00618)
A4D 04940 1000002F (00619)
(00620) *
A4E 04942 4E560000 (00621)
A4F 04944 00530000 (00622)
A50 04946 00150815 (00623)
A51 04948 00000894 (00624)
A52 0494A 911E0050 (00625)

; FIRST SEARCHING
; SIGNAL APS TO PRODUCE DATA
; A1=S(J-T)
; AB=S(J)
;
; R=S(J)-S(J-T)
; R=ABS(S(J)-S(J-T))
;
; A6=SUM3
;
; R=S(J)-S(J-T)
;
; R=SUM3
; R=SEARCHING SIZE-1
; EXO=KBLK/4
;
; A3=KBLK/4
; A2=IEND-B.5-T
;
; R=SMALL-SUM3
; A3=KBLK/4
;
; ----
; A4=SEARCHING SIZE-1

```


PAGE 19: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 38, 1988

```

A6F B4984 B8964596 (B8678)
A7B B4986 B8CFB88B (B8671)
A71 B4988 B8E9B88B (B8672)
A72 B498A 4F74B894 (B8673)
A73 B498C B888B8CF (B8674)
A74 B498E B888B8E9 (B8675)
A75 B499B B885B885 (B8676)
A76 B4992 B893B898 (B8677)
A77 B4994 9B1EB88B (B8678)
      (B8679) *
      (B8680) *
      (B8681) *
      (B8682) *
      (B8683) *
      (B8684) *
      (B8685) *
      (B8686) *
      (B8687) *
      (B8688) *
      (B8689) *
      (B8690) *
      (B8691) *
A78 B4996 B85B88EA (B8691)
A79 B4998 4AB8B8F4 (B8692)
A7A B499A B8F14A8B (B8693)
A7B B499C B888B8E7 (B8694)
A7C B499E B8884F94 (B8695)
A7D B49AB B888B88E (B8696)
A7E B49A2 B888B854B (B8697)
A7F B49A4 B88F88F7 (B8698)
A8B B49A6 4C34B884 (B8699)
A81 B49AB B8F73A8B (B8700)
A82 B49AA B888B89C (B8701)
A83 B49AC 9B1EB88B (B8702)
      (B8703) *
A94 B49AE 2E8B888B (B8704)
A85 B498B 46A846AB (B8705)
A86 B4982 846B888B (B8706)
A87 B4984 168B888B (B8707)
A88 B4986 B888B885B (B8708)
A89 B4988 48314616 (B8709)
A8A B498A B88E888A (B8710)
A8B B498C 844B8898 (B8711)
A8C B498E B84AB88B (B8712)
A8D B49CB B888B88B (B8713)

MOV(A6),ADD(A4,A6)
MOV(IQ,M7) \ NOP
MOV(IQ,M1) \ NOP
MOV(A4),SUB(A3,A7) \ MOV(R,A4)
NOP \ MOV(IQ,M7)
NOP \ MOV(IQ,M1)
MOV(P,A5)
MOV(R,A3) \ MOV(R,EXO)
JUMPC(LOOP9,T1)

OUTPUT BETA AND T

REGISTERS TO BE USED:

ADM1: A6,A1,A4,A7,M6,M2,M5,M7,EXO
ADM2: A6,A2,A4,A6,A7,M2,M6,EXO

INPUT SEQUENCE: 1/2**15,IEND-B.5,B.88B1,1START,LOWER BETA LIMIT,
UPPER BETA LIMIT
OUTPUT SEQUENCE: T(IN INTEGER)

MOV(EXT,A6) \ MOV(IQ,M2)
ADD(A6,A4) \ MOV(IQ,A4)
MOV(IQ,A1) \ SUB(A4,A2)
NOP \ MOV(IQ,A7)
NOP \ MOV(A4),SUB(A4,A7)
NOP \ MOV(R,M6)
NOP \ MUL(M2,M6)
MOV(R,M7) \ MOV(IQ,A7)
MOV(A4),SUB(A1,A4) \ MOV(P,A4)
MOV(IQ,A7) \ ALIGN(A4)
MOV(R,NULL) \ MOV(R,OQ)
JUMPC(RESET,T1)

RCP(A4) \ NOP
MOV(MB),ADD(A5,A6) \ ADD(A5,A6) \ MB=FB
MUL(MB,M7) \ NOP
MOV(EXO),K(2) \ NOP
MOV(P,A6) \ MOV(EXT,A6)
MOV(A1),SUB(A1,A6) \ MOV(A6),ADD(A6,A6) \ R=2-A6
MOV(R,M6) \ MOV(R,M2)
MUL(MB,M6) \ MOV(R,EXO)
MOV(EXT,M2) \ NOP
MOV(P,EXO) \ NOP

M2=1./(2.**15)
A4=IEND-B.5
A7=1START
ENCODE T
M6=T
A7=LOWER BETA

R=SUM1
A6=SUM1
R=SUM1
M2=SUM1
EXO=SUM1

```



```

PAGE 21:      MAP MODULES FOR THE P A R C ALGORITHM      --- JAN. 30, 1980

      00000000 (00750) PICH3SSZ-0A-PICH3SSA
      04A06    (00759)      END
              (00760)      *
              (00761)      EJECT

      ! COMPUTE THE SIZE OF THE MODULE

```



```

PAGE 23:          AP MODULES FOR THE P A R C ALGORITHM      --- JAN. 30, 1980

A15 04A30 2A210039 (00006)
A16 04A3A 2CA10001 (00007)
A17 04A3C 2E0A0001 (00008)
A18 04A3E 30210039 (00009)
A19 04A40 32A10001 (00010)
A1A 04A42 34200004 (00011)
A1B 04A44 362020F8 (00012)
A1C 04A46 380A0001 (00013)
A1D 04A48 3A210039 (00014)
A1E 04A4A 3CA10001 (00015)
A1F 04A4C 3E201C01 (00016)
      (00017) *
      (00018) *
      (00019) *
      (00020) *
      (00021) *
      (00022) *
      (00023) *
      (00024) *
      (00025) *
      (00026) *
      (00027) *
      (00028) *
      (00029) *
      (00030) *
      (00031) *
      (00032) *
      (00033) *
      (00034) *
      (00035) *
      (00036) *
      (00037) *
      (00038) *
      (00039) *
      (00040) *
      (00041) *
      (00042) *
      (00043) *
      (00044) *
      (00045) *
      (00046) *
      (00047) *
      (00048) *
      (00049) *

      VLOOP3
      VLOOP4

      ADDL(BV2,1)
      AND8(BV2,BW,TF)
      ADD(BR3,1,TF)
      ADDL(BV2,1)
      AND8(BV2,BW,TF)
      SUBL(BR2,4),JUMPP(VLOOP2)
      ADDL(BR2,3),JUMPN(CKP)
      ADD(BR3,1,TF)
      ADDL(BV2,1)
      AND8(BV2,BW,TF)
      SUBL(BR2,1),JUMPP(VLOOP4)

      CHECK ADAPTIVITY (INCLUDING FILTERING AND PITCH REPETITION)

      REGISTERS PERMANENTLY USED BY THIS PROGRAM:

      BV2=THE POINTER FOR ADAM OUTPUT ADDRESS
      BV1=THE POINTER FOR PARC INPUT ADDRESS

      REGISTERS AFFECTED: BR0,BR2,BR3
      FLAGS AFFECTED: G2,AF1

      INSGTH=0L+1
      LOAD(BR3,650)
      MOV8(BR2,BV2)
      SUB8(BR2,BV1),JUMPP(CONT1)
      ADD(BR2,1024)
      INSGTH=0L+1
      LOAD(BR0,500)
      SUB8(BR2,BR3),JUMPN(NEXT1)
      SET(G2)
      ADD8(BR2,BR3)
      SUB8(BR2,BR0),JUMPN(NOTR)
      SET(AF1)

      FILTER

      REGISTERS AFFECTED: BR0,BR2,BR3,BW0

      LOAD(BR0,SVT510),L,TF)
      MOV8(BR2,BV1,TF)
      ADD(BR0,2,TF)

      ; OUTPUT ADDRESS IN SAMPLE BUFFER
      ;
      ; OUTPUT ADDRESS IN SAMPLE BUFFER
      ;
      ; OUTPUT ADDRESS IN SAMPLE BUFFER
      ;
      ; IF > 0 -> NO FILTER IS EMPLOYED
      ; SIGNAL APU --- FILTERING

      ; INITIALIZE PITCH REPETITION THRESHOLD
      ; PITCH REPETITION THRESHOLD
      ;
      ;
      ; INITIALIZE FILTER THRESHOLD
      ; NO FILTER THRESHOLD
      ; IF < 0 -> PITCH REPETITION
      ; -> PITCH REPETITION
      ;
      ; IF > 0 -> NO FILTER IS EMPLOYED
      ; SIGNAL APU --- FILTERING

```

PAGE 24: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

A20 04A60 5A9A0002 (00050)
      0000A60 (00051)
A21 04A6A 5C40004F (00052)
      0000A6A (00053)
A22 04A6C 5E7003FF (00054)
      0000A6C (00055)
A23 04A6E 60010015 (00056)
      0000A6E (00057)
A24 04A70 622A0001 (00058)
      0000A70 (00059)
A25 04A72 64A90007 (00060)
      0000A72 (00061)
A26 04A74 66220001 (00062)
      0000A74 (00063)
A27 04A76 68A90007 (00064)
      0000A76 (00065)
A28 04A78 6A2A0002 (00066)
      0000A78 (00067)
A29 04A7A 6C900007 (00068)
      0000A7A (00069)
A30 04A7C 6E220001 (00070)
      0000A7C (00071)
A31 04A7E 70A90007 (00072)
      0000A7E (00073)
A32 04A80 72010039 (00074)
      0000A80 (00075)
A33 04A82 74010007 (00076)
      0000A82 (00077)
A34 04A84 76093501 (00078)
      0000A84 (00079)
A35 04A86 78003600 (00080)
      0000A86 (00081)
A36 04A88 7A300024 (00082)
      0000A88 (00083)
A37 04A8A 7CC203EE (00084)
      0000A8A (00085)
A38 04A8C 7E9A0002 (00086)
      0000A8C (00087)
A39 04A8E 8000A00F (00088)
      0000A8E (00089)
A40 04A90 80400013 (00090)
      0000A90 (00091)
A41 04A92 82010011 (00092)
      0000A92 (00093)
A42 04A94 84310011 (00094)
      0000A94 (00095)
A43 04A96 860006FE (00096)
      0000A96 (00097)
A44 04A98 8800060E (00098)
      0000A98 (00099)
A45 04A9A 8A0043A8 (00100)
      0000A9A (00101)
A46 04A9C 8C200020 (00102)
      0000A9C (00103)

      ADD(BR0,2,TF)
      INSFBS=0L+1
      LOAD(BR0,79)
      LOAD(BR3,1023)
      MOV8(BV0,BR2)
      ADD(BR2,1)
      AND8(BR2,BR3,TF)
      SUB(BR2,1)
      AND8(BR2,BR3,TF)
      ADD(BR2,2)
      AND8(BR2,BR3,TF)
      SUB(BR2,1)
      AND8(BR2,BR3,TF)
      ADDL(BV0,1)
      AND8(BV0,BR3,TF)
      SUBL(BR0,1),JUMPP(VLOOP1)
      JUMP(NOFT+1)

      FILTERING BLOCK
      ;INI. FILT. BLK. SIZE
      ;FILTERING SIZE
      ;MASK**
      ;SF(K-1) POINTER
      ; S(K)
      ; SF(K-1)
      ; S(K)
      ; SF(K)
      ; LOOPING

      PITCH PERIOD AND CORRELATION COEFFICIENT CALCULATIONS
      REGISTERS AFFECTED: BR0,BV0,BV3
      FLAGS AFFECTED: G0
      SET(G0)
      LOAD(BR0,SVTS4S(1),L,TF)
      ADD(BR0,2,TF)
      INSCST=0L+1
      LOAD(BR0,19)
      MOV8(BV0,BR0)
      MOV8(BV3,BR0)
      SEARCHING T
      REGISTERS AFFECTED: BR0,BR2,BR3,BV0
      FLAGS AFFECTED: AF0,AF3
      FLAGS SENSED: AF0,AF2,AF3
      JUMPS(FINISH,AF2)
      JUMPS(RECORD,AF3)
      JUMPC(IN4,AF0)
      CLEAR(AF0)

      ; SIGNAL APU --- NO FILTERING
      ;KBLK/4
      ;CORRELATION SIZE
      ;INI. LOWER LIMIT OF T
      ;STARTING 0 OF CORRELATION
      ;INI VALUE OF T
      ;JUMP IF FINISH PITCH PERIOD SEARCHING
      ;RECORD T
      ;WAIT FOR SIGNAL

```


PAGE 27: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(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) *
(01001) *
(01002) *
(01003) *
(01004) *
(01005) *
(01006) *
(01007) *
(01008) *
(01009) *
(01010) *
(01011) *
(01012) *
(01013) *
(01014) *
(01015) *
(01016) *
(01017) *

```

PITCH EXTRACTION ADAPTIVE RESIDUAL CODER
P A R C

IT USES V3.5 MAP-EXECUTIVE

* DESCRIPTION:

THIS IS A SPEECH CODING PROGRAM IN MAP-300 ARITHMETIC PROCESSOR FOR DIGITAL TRANSMISSION OF SPEECH AT A RATE OF 9600 BITS PER SECOND. THE ALGORITHM COMBINES PITCH EXTRACTION LOOP, PITCH COMPENSATING ADAPTIVE QUANTIZER, SEQUENTIALLY ADAPTIVE PREDICTOR.

1. THE REDUNDANCY OF SPEECH IS REMOVED IN THE PITCH EXTRACTION LOOP BLOCK BY BLOCK.
2. THE SEQUENTIALLY ADAPTIVE PREDICTOR USING BACKWARD ADAPTION IS USED TO FORM AN ESTIMATE OF PITCH REDUCED SIGNAL.
3. THE ERROR IN THIS ESTIMATE IS NORMALIZED AND QUANTIZED BY THE PITCH COMPENSATING ADAPTIVE QUANTIZER.
4. IF PITCH REPETITION MODE HAPPENS, A BLOCK OF SHAT IS DUPLICATED.

* SYMBOLIC ABBREVIATIONS:

INSTX LOCATION OF PITCH REPETITION SIZE IN APS MODULE

* BUFFERS:

1. SAMPLE BUFFER: IT CONTAINS INPUT SPEECH SAMPLES AND FILTERED INPUT SPEECH SAMPLES.
--- CIRCULAR BUFFER --- SHORT REAL FORMAT ---
2. WHAT BUFFER: IT CONTAINS RECONSTRUCTED REDUCED SPEECH SAMPLES.
--- CIRCULAR BUFFER --- SHORT REAL FORMAT ---
3. SHAT BUFFER: IT CONTAINS RECONSTRUCTED SPEECH SAMPLES.
--- CIRCULAR BUFFER --- SHORT REAL FORMAT ---
4. Q BUFFER: IT CONTAINS OUTPUT QUANTIZING LEVELS.
--- DOUBLE BUFFERS --- INTEGER * 2 FORMAT ---
5. PARAMETER BUFFER: IT CONTAINS PARAMETERS FOR QUANTIZER INCLUDING A(8),(T(J),OUT(J),EXP(N(J),B(J)) (IN SEQUENCE)

* RESTRICTION:

PAGE 32: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

A30 0486A 08314831 (01194)
A39 0486C 080E080E (01195)
A3A 0486E 054084C0 (01196)
A3B 04870 300000AF (01197)
A3C 04872 300000AF (01198)
      (01199) *
      (01200) *
      (01201) *
      (01202) *
      (01203) *
      (01204) *
      (01205) *
      (01206) *
      (01207) *
      (01208) *
      (01209) *
      (01210) *
      (01211) *
      (01212) *
      (01213) *
      (01214) *
      (01215) *
A3D 04874 080E0800 (01216)
A3E 04876 080E0800 (01217)
A3F 04878 04EE4780 (01218)
A40 0487A 16000800 (01219)
A41 0487C 08000800 (01220)
A42 0487E 08F408F1 (01221)
A43 04880 080008F6 (01222)
A44 04882 08F00800 (01223)
A45 04884 08F10800 (01224)
A46 04886 08060800 (01225)
A47 04888 4E290894 (01226)
A48 0488A 080E0800 (01227)
      (01228) *
      (01229) *
      (01230) *
      (01231) *
      (01232) *
      (01233) *
      (01234) *
      (01235) *
      (01236) *
      (01237) *

MOV(A1),SUB(A1,A3) \ MOV(A1),SUB(A1,A3) \ R=2-P1
MOV(R,M6) \ MOV(R,M6) \ M6=2-P1
MUL(M2,M6) \ MUL(M1,M6) \ P=FB*(2-P1)
CALL(RCP1) \ CALL SUBROUTINE RCP1
CALL(RCP1)

CALCULATE REDUCED SPEECH

REGISTERS USED:
ADM1:
ADM2: A4,A7

REGISTERS AFFECTED:
ADM1: A0,A1,A4,A6,M1,M5,M6,M7
ADM2: A1,A4,A6,M0,M4,M7

INPUT SEQUENCE: SHAT(K-T),BETA,SMIN,G,NQ,RUNCOUNT,COMPANSATION,S(K),1/2

FUNCTIONS: V(K) = S(K) - BETA * SHAT(K-T)
SAMPLE COUNT = SAMPLE COUNT + 1

MOV(IGA,M7) \ NOP \ M7=SHAT(K-T)
MOV(IGA,M1) \ NOP \ M1=BETA
MOV(M6),MUL(M1,M7) \ ADD(A4,A7) \ M6=1/2*SIZE
K(1) \ MOV(IGA,M0) \
NOP \ MOV(IGA,M7) \
MOV(IGA,A4) \ MOV(IGA,A1) \
NOP \ MOV(IGA,A6) \
MOV(IGA,A0) \ NOP \ A0=COMPANSATION
MOV(P,A6) \ MOV(R,M4) \ A1=S(K)
MOV(M1),SUB(A1,A6) \ MOV(R,A4) \ A6=BETA*SHAT(K-T)
MOV(IGA,M5) \ NOP \ M5=1/2*14

M0=SMIN
M7=G
A1=NQ
A6=RUNLENGTH

QUANTIZER

REGISTERS USED:
ADM1: A0,A2,A4,A5,M1,M5,M6
ADM2: A1,A2,A5,A6,A7,M0,M7

REGISTERS AFFECTED:
ADM1: A1,A2,A3,A4,M1,M2,M7,EX0

```

```

ADM2: AB,A1,A3,A5,M1,M6,M7,EX0
      INPUT SEQUENCE: T(J),OUT(J),EXPN(J),B(J)
      OUTPUT SEQUENCE: VHAT(K),L
      FLAGS AFFECTED: GB,AFB,AF1,AF2,AF3
      FLAGS SENSED: GB,AFB,AF3
      FUNCTIONS: E(K) = V(K) - P(K)
                  ABS(E(K)/SIZE) C.F. T(J)
                  RMS = SMIN
                  G/(RMS**2)
                  GET QUANTIZER LEVEL
                  SIZE = EXPN(J)
                  EHAT(K) = +- OUT(J) * SIZE
                  VHAT(K) = EHAT(K) + P(K)
                  SIZE = MAX(SIZE + EXPN(J), RMS * SMIN)
                  OUTPUT VHAT(K) AND QUANTIZER LEVEL
                  CHECK RUN LENGTH CONDITION
                  UPDATE RUN LENGTH COUNTER AND BIT COUNTER
      MOV(R,A1) \ R(A2)
      SUB(A1,A5) \ NOP
      MOV(R,M2) \ MOV(P,M1)
      JUMPC(CON2,T1)
      SET(GB)
      NOP \ MOV(R,M6)
      ABSMUL(M2,M6) \ MUL(MB,M6)
      MOV(P,A1) \ MOV(P,A3)
      NOP \ MUL(M1,M7)
      QUANTIZER LOOP
      NOP
      JUMPS(QUAN1,AFB)
      SET(AFB)
      MOV(IGA,A3) \ NOP
      MAX(A1,A3) \ SUB(A1,A7)
      MOV(R,NULL) \ MOV(R,A1)
      JUMPS(QUAN6,T1)
      SET(AF3)
      JUMPC(QUAN1,T2)
      SET(AF2)
      ;
      ;
      ; SIGNAL APS TO PRODUCE T(K) ADDRESS
      ; A3=T(K)
      ; ERROR > T(K)?
      ; WAIT FOR T1 FLAG
      ; ERROR < T(K) GO TO QUAN6
      ; L=1
      ; COUNTER > B GO TO QUAN1
      ; SIGNAL APS TO STOP PRODUCING T(K) ADDRESS

```

A5C 04002 00004F20 (01282)	NOP \ SUB(A1,A7)				
A5D 04004 00000200 (01293)	NOP \ MOV(EXO),R(A5)				
A5E 04006 00510000 (01284)	MOV(EXT,A1) \ NOP				
A5F 04008 00000000 (01285)	NOP \ MOV(R,EXO)				
A60 0400A 91040007 (01286)	JUMPS(NEGA,G0)				
	(01287) *				
	(01288) *				
	(01289) *				
A61 0400C 004A00E9 (01290)	MOV(EXT,M2) \ MOV(IGA,M1)				
A62 0400E 00EF008E (01291)	MOV(IGA,M7) \ MOV(R,M6)				
A63 0400F 056004CF (01292)	MUL(M2,M7) \ MOV(M7),MUL(M1,M6)				
A64 04010 00000300 (01293)	MOV(P,EXO) \ MOV(P,EXO)				
A65 04012 00530051 (01294)	MOV(EXT,A3) \ MOV(EXT,A1)				
A66 04014 00A00000 (01295)	MUL(M1,M5) \ NOP				
A67 04016 45600320 (01296)	ADD(A3,A5) \ MAX(A1,A3)				
A68 04018 00500000 (01297)	MOV(EXT,EXO) \ MOV(R,A5)				
A69 0401A 00930049 (01298)	MOV(R,A3) \ MOV(EXT,M1)				
	(01299) *				
A6A 0401C 009C0000 (01300)	MOV(R,OQ) \ NOP				
A6B 0401E 00000075 (01301)	JUMPC(QUAN4,AF3)				
A6C 04020 490002C0 (01302)	SUB(A4,A1) \ R(A6)				
A6D 04022 20200200 (01303)	CLEAR(AF3)				
A6E 04024 00A00000 (01304)	MOV(P,NULL) \ NOP				
A6F 04026 00090000 (01305)	MOV(R,M1) \ NOP				
A70 04028 04A00000 (01306)	MUL(M1,M5) \ NOP				
A71 0402A 00840000 (01307)	MOV(P,A4) \ NOP				
A72 0402C 3A000000 (01308)	ALIGN(A4) \ NOP				
A73 0402E 009C0000 (01309)	MOV(R,OQ) \ MOV(R,A0)				
A74 04030 10000070 (01310)	JUMPC(QUAN5)				
A75 04032 00840000 (01311)	MOV(P,A4) \ SUB(A0,A7)				
A76 04034 3A000000 (01312)	ALIGN(A4) \ NOP				
A77 04036 009C0000 (01313)	MOV(R,OQ) \ MOV(R,A0)				
A78 04038 001F0070 (01314)	JUMPC(QUAN5,T2)				
A79 0403A 420002C0 (01315)	ADD(A0,A2) \ R(A6)				
A7A 0403C 00920000 (01316)	MOV(R,A2) \ MOV(R,A0)				
A7B 0403E 00F10000 (01317)	MOV(IGA,A1) \ NOP				
A7C 04040 49A00000 (01318)	SUB(A2,A1) \ NOP				
A7D 04042 00920000 (01319)	MOV(R,A2) \ NOP				
A7E 04044 001E0000 (01320)	JUMPC(ADAP1,T1)				
A7F 04046 20A002A0 (01321)	SET(AF3)				
	(01322) *				
	(01323) *				
	(01324) *				
	(01325) *				

ADAPTATION

REGISTERS USED:

	A1-1		
	!CHECK ERROR	EXO=STATE VARIABLE	
	!IF ERROR<0 GO TO MEGA		
	M2=STATE VARIABLE	M1=OUT(J)	
	M7=EXP(N(J))	M6=STATE VARIABLE(SI	
	!SIZE=EXP(N(J))	M7=P1 P=OUT(J)=SIZE	
	!EXO=SIZE*EXP(N(J))	EXO=OUT(J)=SIZE	
	!A3=+-OUT(J)=SIZE	A1=SIZE*EXP(N(J))	
	!P=1/2**14		
	!OUT(J)=SIZE*PRE	CHOOSE NEW SIZE	
	!EXO=+-OUT(J)=SIZE	RENEW SIZE	
	!A3=VHAT(K)	M1=EQ	
	!OUTPUT VHAT(K)		
	!JUMP IF Q(L)=1	A6=RUNLENGTH	
	!CALCULATE L		
	!CLEAR AF3		
	!M1=L		
	!READY TO ALIGN		
	!OUTPUT L	A0=RUNCOUNT	
	!OUTPUT L=1		
	!DECREMENT A0		
	!JUMP IF NO ENOUGH FOR RUN LENGTH	A6=RUNLENGTH	
	!COMPASATING	A0=RUNCOUNT	
	!A1=B(K)		
	!UPDATE BITCOUNT		
	!JUMP IF BITCOUNT > 0		
	!SIGNAL END OF PARC		

```

(01326) *
(01327) *
(01328) *
(01329) *
(01330) *
(01331) *
(01332) *
(01333) *
(01334) *
(01335) *
(01336) *
(01337) *
(01338) *
(01339) *
(01340) *
(01341) *
(01342) *
(01343) *
(01344) *
(01345) *
(01346) *
(01347) ADAP1
(01348)
(01349)
(01350)
(01351) *
(01352)
(01353)
(01354)
(01355)
(01356)
(01357)
(01358)
(01359)
(01360)
(01361)
(01362)
(01363)
(01364)
(01365)
(01366)
(01367) ADAP2
(01368)
(01369)

ADM1: A3,A6
ADM2: A7,M1,M7

REGISTERS AFFECTED:

ADM1: A0,A1,A4,A5,A6,M1,M2,M6
ADM2: A3,M1,M6,M7,EXO

INPUT SEQUENCE: AINV*(1/ALAD-1),ALAD,N-2.5,NULL BITS,
OUTPUT SEQUENCE: SHAT(K-1),A(1),VHAT(K-1)

FLAGS AFFECTED: G0

FUNCTIONS: ERR = G * EHAT(K) / (RMS ** 2)
          A(1) = (A(1) + AINV*(1/ALAD-1)) * ALAD + VHAT(K-1)*ERR
          SHAT(K) = VHAT(K) + BETA*SHAT(K-1)
          OUTPUT SHAT(K) AND A(1)
          A(1) = A(1)*ALAD + VHAT(K-1)*ERR
          OUTPUT A(1)

MOV(IGA,A4) \ NOP
MOV(IGA,M6) \ NOP
NOP \ MOV(IGA,A3)
MOV(IGA,A0) \ NOP

MOV(IGA,A5) \ MUL(M1,M7)
ADD(A5,A4) \ NOP
MOV(M2),ADD(A3,A6) \ MOV(P,M1)
MUL(M2,M6) \ MOV(IGA,M7)
MOV(IGA,M1) \ MUL(M1,M7)
MOV(R,00) \ NOP
MOV(A1),MUL(M1,M6) \ MOV(P,EXO)
MOV(EXT,A6) \ MOV(IGA,M6)
ADD(A6,A1) \ MUL(M1,M6)
CLEAR(G0)
MOV(R,00) \ NOP
MOV(P,A1) \ MOV(P,EXO)
MOV(EXT,A6) \ NOP
ADD(A6,A1) \ NOP
MOV(R,00) \ NOP
MOV(IGA,M1) \ NOP
MUL(M1,M6) \ MOV(IGA,M7)
NOP \ MUL(M1,M7)

;A4=AINV*(1/ALAD-1)
;M6=ALAD
A3=N-2.5
;A0=NULL BIT
;A5=A(1)
;A(1)+A4
;M2=R1
;P2=ALAD*A1
;M1=A(2)
;OUTPUT SHAT(K)
;A1=P2
;A6=VHAT(K-1)*ERR
;R=NEW A(1)
;OUTPUT A(1)
;A1=ALAD*A(2)
;A6=VHAT(K-2)*ERR
;R=NEW A(2)
;OUTPUT A(2)
;M1=A(1)
;P=A(1)*ALAD
M7=VHAT(K-1)
P=VHAT(K-1)*ERR

P1=G*EQ/RMS**2
M1=P1
M7=VHAT(K-1)
P3=VHAT(K-1)*P1
EXO=P3
M6=VHAT(K-2)
P=VHAT(K-2)*ERR
EXO=VHAT(K-2)*ERR
M7=VHAT(K-1)
P=VHAT(K-1)*ERR

```

```

A96 0AC26 00010000 (01370)
A97 0AC20 0056AF60 (01371)
A98 0AC2A 41C00000 (01372)
A99 0AC2C 009C0000 (01373)
A9A 0AC2E 901F0000 (01374)
A9B 0AC30 00000000 (01375)
      (01376) *
      (01377) *
      (01378) *
      (01379) *
      (01380) *
A9C 0AC32 910000A2 (01381)
A9D 0AC34 90050011 (01382)
      (01383) *
      (01384) *
      (01385) *
      (01386) *
      (01387) *
      (01388) *
      (01389) *
      (01390) *
      (01391) *
      (01392) *
      (01393) *
      (01394) *
      (01395) *
      (01396) *
      (01397) *
      (01398) *
      (01399) *
      (01400) *
      (01401) *
      (01402) *
      (01403) *
A9E 0AC36 00400000 (01404) NULL1
A9F 0AC38 00920000 (01405)
AA0 0AC3A 911E00A2 (01406)
AA1 0AC3C 00120000 (01407)
AA2 0AC3E 00000000 (01408) FEND
AA3 0AC40 00000000 (01409)
AA4 0AC42 02400A00 (01410)
AA5 0AC44 0000025C (01411)
AA6 0AC46 009C0000 (01412)
AA7 0AC48 00020000 (01413)

MOV(P,A1) \ MOV(P,EXO)
MOV(EXT,A6) \ SUB(A3,A7)
ADD(A6,A1) \ NOP
MOV(R,00) \ MOV(R,A3)
JUMPC(ADAP2,T2)
NOP
END OF PROCESSING ?
FLAGS SENSED: G1,AF3
JUMPS(FEND,AF3)
JUMPC(PRE3,G1)
NULL MODE
REGISTERS USED:
ADM1: A0,A2
ADM2: A2,A5,A7,M4
REGISTERS AFFECTED:
ADM1: A2
ADM2: M3,EXO
FLAGS AFFECTED: G1,AF3,RA
INPUT SEQUENCE: 1/2**15
OUTPUT SEQUENCE: END CODE,BIT COUNT,RMS,STATE VARIABLE, OF SAMPLES
FUNCTIONS: IF MORE THAN TWO NULL CODES, SET BIT COUNT = 0
            OTHERWISE UPDATE BITCOUNT
            OUTPUT END CODE, BITCOUNT, RMS, SIZE AND SAMPLE COUNTER

SUB(A2,A0) \ NOP
MOV(R,A2) \ NOP
JUMPS(FEND,T1)
MOV(ZERO,A2) \ NOP
NOP \ MOV(IQA,M3)
NOP \ MUL(M3,M4)
R(A2) \ NEG(A7)
NOP \ MOV(OQ),R(A2)
MOV(R,00) \ MOV(P,EXO)
MOV(EXT,A2) \ MOV(OQ),R(A5)

;A1=A(1)*ALAD
;A6=VHAT(K-1)*ALAD
;R=NEW A(1)
;OUTPUT NEW A(1)
;END ?

;JUMP IF BITCOUNT < 0
;JUMP IF NOT RUN OUT OF SAMPLE

;UPDATE BITCOUNT
;
;JUMP IF BITCOUNT < 0
;RESET BITCOUNT TO SYNC. TRANSMISSIONS
;M3=1./(2.**15)
;
;
;OUTPUT END CODE
;OUTPUT BITCOUNT
;OUTPUT RMS

```

NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORI
APR 80 J L MELSA, D L COHN, A ARORA OCA

THM AT 9600 --ETC(U)

APR 80 J L MELSA, D L COHN, A ARORA

DCA100-79-C-0005

ML

3 of 3

AD
A088534

END
DATE
FILMED
8-80
DTIC


```

(B1448) *
(B1449) *
(B1450) *
(B1451)
(B1452)
(B1453)
(B1454)
(B1455)
(B1456)
(B1457)
(B1458) *
(B1459) VPCTXS
(B1460) *
(B1461) *
(B1462) *
(B1463) *
(B1464) *
(B1465) *
(B1466) *
(B1467) *
(B1468) *
(B1469) *
(B1470) *
(B1471) *
(B1472) *
(B1473) *
(B1474) *
(B1475) *
(B1476) *
(B1477) *
(B1478) *
(B1479) *
(B1480)
(B1481)
(B1482)
(B1483)
(B1484)
(B1485)
(B1486)
(B1487)
(B1488)
(B1489) *
(B1490) *
(B1491) *

      B4C7C 00004082
      B4C7E 0000ACF4
      B4C80 0001
      B4C81 0100
      B4C82 0000AC92

      B4C84 00300032
      B4C86 02C20404
      B4C88 048A0002
      B4C8A 068A0002
      B4C8C 088A0002
      B4C8E 0A8A0002
      B4C90 0CC2046E
      B4C92 0E401000
      B4C94 10220000
      B4C96 12020000
      B4C98 14310011

      APS MODULE OF PCTX PROGRAM

      EVEN
      ADDR VPCTXSI
      DATA VPCTXS+2*VPCTXSX
      DATA 1
      DATA VPCTXSZ
      ADDR VPCTXSA
      EVEN

      THE FOLLOWING REGISTERS ARE PERMANENTLY USED BY THIS PROGRAM

      BR1: ADDRESS OF VHAT(K)
      BR2: ADDRESS OF S(K-1)
      BR3: ADDRESS OF SHAT(K-T-1)
      BV0: 1023
      BV1: ADDRESS OF SHAT(K-1)
      BV2: SD(K+N-1)
      BV3: QUANTIZER OUTPUT ADDRESS - 1

      INITIALIZATION

      REGISTERS AFFECTED: BR0,BR2,BV3

      INPUT SEQUENCE: BIT COUNT, BIT BUFFER SIZE, RMS, SIZE, RUN LENGTH,
                      PITCH REPETITION SIZE

      SET(RA)

      LOAD(BR0,SVT63S(1),L,TF)
      ADD(BR0,2,TF)
      ADD(BR0,2,TF)
      ADD(BR0,2,TF)
      ADD(BR0,2,TF)
      LOAD(BR0,SVT118S(1),L,TF)
      LOAD(BR0,[1])
      SUB(BR2,MSS)
      SUB(BR0,MSS)
      MOVB(BV3,BR0)

      ;BITCOUNT
      ;BIT BUFFER SIZE
      ;RMS
      ;STATE VARIABLE
      ;RUN LENGTH
      ;O-BUFFER LIMIT
      ;O BUFFER

      ;
      ;ADDRESS POINTER OF Q BUFFER - 1

      CHECK PITCH REPETITION

```

(01492) *		FLAG SENSED: G2	
(01493) *		REGISTER AFFECTED: BR0, BR2, BR3, BR0, BV1, BV3	
(01494) *			
(01495) *			
(01496) *			
(01497) *	START		
A08 04C9A 17310039		ADDL(BV3, 1, TE)	: PHONY BETA ADDRESS
A09 04C9C 10290012		MOV8(BR2, BV1)	: STARTING ADDRESS OF INPUT SPEECH(K-1)
A10 04C9E 1A7E0400		LOAD(BR3, 1024(3), S)	: STARTING LOCATION OF SHAT BUFFER **
A11 04CA0 1C39002A		ADD8(BR3, BV1)	:
A12 04CA2 1E110017		MOV8(BV1, BR3)	: RESERVE FOR PITCH REPETITION TO USE
A13 04CA4 2B390020		SUB8(BR3, BV0)	
A14 04CA6 224003FF		LOAD(BR0, 1023)	: SAMPLE BUFFER SIZE - 1 **
A15 04CA8 24810011		MOV8(BV0, BR0)	
A16 04CAA 260072A6		JUMPC(VCHECK, G2)	: JUMP IF NO PITCH REPETITION
A17 04CAC 28C2040E		LOAD(BR0, SVT68S(1), L, TF)	: PHONY BETA BITS
A18 04CAE 2A0A0000		ADD(BR0, 2, TF)	: PITCH REPETITION SIZE
A19 04CB0 2C00003C		INSTX=01, 1	: INITIALIZE PITCH REPETITION SIZE
A20 04CB2 2E110021		LOAD(BR0, 60)	: PITCH REPETITION SIZE
A21 04CB4 30390021		SUB8(BV1, BR0)	: BACK A PITCH REPETITION
A22 04CB6 32F20001		SUB8(BR3, BR0)	: BACK A PITCH REPETITION
A23 04CB8 343A0001		SUB(BR0, 1)	: PITCH REPETITION-1
A24 04CBA 36390000		ADD(BR3, 1)	:
A25 04CBC 380A0400		AND8(BR3, BV0)	: SHAT(K-T-G) **
A26 04CCE 3A110039		ADD(BR3, 1024, TF)	: RESTRICT SHAT BUFFER TO START FROM 1024
A27 04CC0 3C110000		ADDL(BV1, 1)	:
A28 04CC2 3E110028		AND8(BV1, BV0)	:
A29 04CC4 40910039		ADDL(BV1, 1, TF)	: PITCH REPETITION
A30 04CC6 42091A01		SUBL(BR0, 1), JUMPP(VCAP)	:
A31 04CC8 44007200		JUMPC(VCHECK)	:
(01522) *		PREDICTOR	
(01523) *			
(01524) *		FLAGS AFFECTED: AFB	
(01525) *			
(01526) *		REGISTERS AFFECTED: BR0, BR1	
(01527) *			
(01528) *			
(01529) *	VLOOP		
A32 04CCA 46C20012		LOAD(BR0, SVT70S(1), L, TF)	: RMSMIN
A33 04CCC 480A0002		ADD(BR0, 2, TF)	: N-2.5
A34 04CC0 4A0A0002		ADD(BR0, 2, TF)	: 1-ALP
A35 04CC2 4C0A0002		ADD(BR0, 2, TF)	: ALP
A36 04CC4 4E200020		CLEAR(AFB)	:
A37 04CC6 50C40C00		LOAD(BR0, 3072(2), L, TF)	: STARTING LOCATION OF A(I) **
A38 04CC8 52120001		SUB(BR1, 1)	:

PAGE 41: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

A47 #4012 9E004CEA (#1500)
A48 #4014 900046A8 (#1501)
A49 #4016 92200028 (#1502)
A4A #4018 940A000A (#1503)
A4B #401A 96004660 (#1504)
A4C #401C 980A000A (#1505)
A4D #401E 9A20002A (#1506)
A4E #4020 9C000010 (#1507)
A4F #4022 9E0A0002 (#1508)
A50 #4024 A0200029 (#1509)
A51 #4026 A20A0002 (#1500)
A52 #4028 A40055E4 (#1591)
A53 #402A A60A0002 (#1592)
A54 #402C A8005560 (#1593)
A55 #402E AA0A0004 (#1594)
A56 #4030 AC110013 (#1595)
A57 #4032 AE910000 (#1596)
A58 #4034 B1310039 (#1597)
A59 #4036 B2C20426 (#1603)
A5A #4038 B48A0002 (#1604)
A5B #403A B68A0002 (#1605)
A5C #403C B88A0002 (#1606)
A5D #403E BA120001 (#1607)
A5E #4040 BC4A0C00 (#1608)
A5F #4042 BE990000 (#1609)
A60 #4044 C0120001 (#1610)
A61 #4046 C20A0002 (#1611)
A62 #4048 C40E0400 (#1612)
A63 #404A C6090020 (#1613)
A64 #404C C8100011 (#1614)
A65 #404E CA990000 (#1615)
A66 #4050 CC120001 (#1616)
A67 #4052 CE4A0C00 (#1617)
A68 #4054 D0910011 (#1618)
A69 #4056 D291003A (#1619)
A6A #4058 D4C40C04 (#1620)
A6B #405A D6990000 (#1621)
A6C #405C D891003A (#1622)
A6D #405E DA120001 (#1623)

JUMPS(VQUAN2,AF2)
JUMPC(VQUAN1,AF0)
CLEAR(AFB)
ADD(BR0,10,TF)
JUMP(VQUAN1)
ADD(BR0,10,TF)
CLEAR(AF2)
JUMP(SHIFT)
ADD(BR0,2,TF)
CLEAR(AF1)
ADD(BR0,2,TF)
JUMPS(VNEGA,GB)
ADD(BR0,2,TF)
JUMP(VNEGA+1)
ADD(BR0,4,TF)
MOV8(BV1,BR1)
AND8(BV1,BV0,TF)
ADDL(BV3,1,TE)

ADAPTION

REGISTERS AFFECTED: BR0,BR1,BV1

LOAD(BR0,SVT79S(1),L,TF)
ADD(BR0,2,TF)
ADD(BR0,2,TF)
ADD(BR0,2,TF)
SUB(BR1,1)
LOAD(BR0,3072(2),L,TF)
AND8(BR1,BV0,TF)
SUB(BR1,1)
ADD(BR0,2,TF)
LOAD(BR0,1024(3),S)
ADD8(BR0,BR2)
MOV8(BV1,BR0,TF)
AND8(BR1,BV0,TF)
SUB(BR1,1)
LOAD(BR0,3072(2),L)
MOV8(BV1,BR0,TF)
ADDL(BV1,2,TF)
LOAD(BR0,3076(2),L,TF)
AND8(BR1,BV0,TF)
ADDL(BV1,2,TF)
SUB(BR1,1)

;ERROR > ALL T(K)
;WAIT FOR SIGNAL
;SEND T(K)
;SEND OUT(J)
;OUT(J)
;EXPN(J)
;B(K)
;VHAT(K) ADDRESS
;L ADDRESS

;AINV*(1/ALAD-1) **
;ALAD
;N-2.5
;NULL BITS
;GET ADDRESS POINTER FOR VHAT BUFFER
;STARTING ADDRESS OF A(I) **

;SHAT(K) ADDRESS
;VHAT(K-2) ADDRESS
;A(1) ADDRESS
;A(2) ADDRESS

```

```

PAGE 42: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 38, 1988

A6E 84068 DC8A8882 (81624) ADD(BR8,2,TF)
A6F 84062 DE998888 (81625) AND8(BR1,8W8,TF)
A78 84064 E891883A (81626) ADDL(BV1,2,TF)
A71 84066 E21A8885 (81627) ADD(BR1,5)
      (81628) *
      (81629) * END OF BUFFER ?
      (81630) *
      (81631) *
      (81632) *
      (81633) *
      (81634) *
      (81635) *
      (81636) *
      (81637) *
      (81638) *
      (81639) *
      (81640) *
      (81641) *
      (81642) *
      (81643) *
      (81644) *
      (81645) *
      (81646) *
      (81647) *
      (81648) *
      (81649) *
      (81650) *
      (81651) *
      (81652) *
      (81653) *
      (81654) *
      (81655) *
      (81656) *
      (81657) *
      (81658) *
      (81659) *
      (81660) *
      (81661) *
      (81662) *
      (81663) *
      (81664) *
      (81665) *
      (81666) *

A72 84068 E48823E8 (81633) VCHECK
A73 8406A E68875E8 (81634) JUMPS(VLOOP,AF8)
A74 8406C E88872A5 (81635) JUMPS(VEND,AF3)
      (81636) *
      (81637) *
      (81638) *
      (81639) *
      (81640) *
      (81641) *
      (81642) *
      (81643) *
      (81644) *
      (81645) *
      (81646) *
      (81647) *
      (81648) *
      (81649) *
      (81650) *
      (81651) *
      (81652) *
      (81653) *
      (81654) *
      (81655) *
      (81656) *
      (81657) *
      (81658) *
      (81659) *
      (81660) *
      (81661) *
      (81662) *
      (81663) *
      (81664) *
      (81665) *
      (81666) *

A75 8406E E8318839 (81641) VEND
A76 84078 ECC2842E (81642) LOAD(BR8,SVT83S(1),L,TF)
A77 84072 EE528484 (81643) LOAD(BR1,SVT63S(1),L)
A78 84074 F8918813 (81644) MOV8(BV1,8R1,TF)
A79 84076 F291883C (81645) ADDL(BV1,4,TF)
A7A 84078 F491883A (81646) ADD(BR8,2)
A7B 8407A F68A8882 (81647) MOV8(BV1,8R8,TE)
A7C 8407C F9118811 (81648) CLEAR(R1)
A7D 8407E FA288831 (81649) NOP
A7E 84088 FC888828 (81650) NOP
      (81651) *
      (81652) *
      (81653) *
      (81654) *
      (81655) *
      (81656) *
      (81657) *
      (81658) *
      (81659) *
      (81660) *
      (81661) *
      (81662) *
      (81663) *
      (81664) *
      (81665) *
      (81666) *

84082 88884C92 88884C92 VPCTXSA=0C
      (81653) *
      (81654) *
      (81655) *
      (81656) *
      (81657) *
      (81658) *
      (81659) *
      (81660) *
      (81661) *
      (81662) *
      (81663) *
      (81664) *
      (81665) *
      (81666) *

84082 88888888 88888888 VPCTXSI DATA IF'B.B'
      (81660) *
      (81661) *
      (81662) *
      (81663) *
      (81664) *
      (81665) *
      (81666) *

88888888 88888888 VPCTXSZ=0L-VPCTXS
      (81662) *
      (81663) *
      (81664) *
      (81665) *
      (81666) *

88888888 88888888 CON3S=0L
      (81664) *
      (81665) *
      (81666) *

EJECT

;JUMP IF AF8 SET --- PARC LOOP NOT END
;JUMP IF BITCOUNT < 8
;WAIT FOR SIGNAL

;OUTPUT END CODE
;1./(2.*15)
;
;BITCOUNT
;RMS
;STATE VARIABLE
;
;# OF SAMPLES

;CHAIN ANCHOR

;COMPUTE MODULE SIZE

```

```

(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) *

```

PITCH EXTRACTION ADAPTIVE RESIDUAL CODER

P A R C
RECEIVER

* DESCRIPTION:

THIS IS A SPEECH CODING PROGRAM IN MAP-300 ARITHMETIC PROCESSOR FOR DIGITAL TRANSMISSION OF SPEECH AT A RATE OF 9600 BITS PER SECOND. THE ALGORITHM COMBINES PITCH EXTRACTION LOOP, PITCH COMPENSATING ADAPTIVE QUANTIZER, SEQUENTIALLY ADAPTIVE PREDICTOR.

1. THE SEQUENTIALLY ADAPTIVE PREDICTOR USING BACKWARD ADAPTION IS USED TO FORM AN ESTIMATE OF PITCH REDUCED SIGNAL.
2. THE ERROR OF THIS ESTIMATE IS RECONSTRUCTED BY INVERSE QUANTIZER.
3. THE REDUNDANCE OF THE CURRENT SAMPLE IS CALCULATED BY PITCH EXTRACTION LOOP.
4. THE RECONSTRUCTED SPEECH IS THE SUM OF THIS ERROR, THE ESTIMATE AND THE REDUNDANCE.

* SYMBOLIC ABBREVIATIONS:

SHSRC --- THE LOCATION OF SHAT POINTER
TSRC --- THE LOCATION OF PITCH PERIOD FOR RECEIVER
VHSRC --- THE LOCATION OF WHAT POINTER
INSRC --- THE LOCATION OF (PITCH REPETITION SIZE - 1)

* BUFFERS:

1. Q BUFFER: IT CONTAINS INPUT QUANTIZING LEVELS.
--- IN 16-BIT INTEGER*2 FORMAT --- DOUBLE BUFFERS ---
2. WHAT BUFFER: IT CONTAINS RECONSTRUCTED REDUCED SPEECH.
--- CIRCULAR BUFFER --- IN SHORT-REAL FORMAT ---
3. SHAT BUFFER: IT CONTAINS RECONSTRUCTED SPEECH.
--- CIRCULAR BUFFER --- IN SHORT-REAL FORMAT ---
4. ADM BUFFER: IT CONTAINS UPSAMPLED SHAT DATA.
--- DOUBLE BUFFER --- SHORT REAL FORMAT ---
5. PARAMETER BUFFER: IT CONTAINS THE PARAMETERS FOR INVERSE QUANTIZER.
SEQUENCE : A(8),(ENPN(J),OUT(J)) --- IN LONG-REAL FORMAT ---

(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) *

* RESTRICTION:

1. THE ORDER OF THE PREDICTOR IS 4.
2. INITIALLY, THE SHAT BUFFER SHOULD BE IN FULL CONDITION.
3. THE LOCATIONS OF SOME ARRAYS ARE AS FOLLOWS:
 SHAT BUFFER --- 2048(2) ... 3071(2)
 VBAT BUFFER --- 2048(3) ... 3071(3)
 AOM BUFFER --- 3584(2)-4095(2)
 PARAMETER BUFFER --- 4096(2) ...
 WHERE QUANTIZER LEVELS ARE DEFINED AS FOLLOWS:
 FROM LOWEST TO HIGHEST
 Q(N),Q(N-1),...,Q(N/2+2),Q(1),Q(2),...,Q(N/2+1)

* CALL TO THIS FUNCTION:

MAP = PCRC(SA,U,V)
 WHERE SA --- SCALAR LOCATION OF ENCD. BETA
 U --- BUFFER # FOR INPUT Q BUFFER
 V --- AOM BUFFER #

DISPATCH TABLE ENTRY

FCB243=243
 #L=AFDTs*3*2*(FCB243-128)
 ADDR PCRCs(R7,1)
 ADDR VPCRCs(R7,1)
 ADDR CSPUSNOS(1,0)
 APU & APS MODULE
 #L=CON3S
 APU MODULE
 EVEN PCRCSSA
 DATA PCRCSSZ
 BEGIN APU(PCRC)
 #M=3
 #A=0

:FCB=243
 :STARTING LOCATION OF AFDT FOR 243

:APU=PCRC
 :APS=VPCRC
 :NO CGPU SUPPORT

:START ON WORD BOUNDARY
 :STARTING ADDRESS
 :SIZE
 :START OF APU MODULE

000000F3
 0000009A

0009A 001E4D86
 0009C 001E4EC2
 0009E 001021FC

00004D84

04D84 0000
 04D85 009A

00000003
 00000000

```

PAGE 45: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

(01755) *
(01756) *
(01757) *
(01758) *
(01759) *
(01760) *
(01761) *
(01762) *
(01763) *
(01764) PCRCSSA K(0.5)
A00 04D86 16601600 MOV(10A,A5) \ NOP
A01 04D88 08F50000 MOV(10A,A6) \ NOP
A02 04D8A 08F60000 MOV(M1),K(1)
A03 04D8C 16891600 MOV(10A,A2) \ NOP
A04 04D8E 08F20000 MOV(10A,A7) \ NOP
A05 04D90 08F00000 MOV(A7),MIN(A5,A6) \ MOV(R,A7)
A06 04D92 6D170000 MOV(A7),MAX(A5,A6) \ MOV(R,A7)
A07 04D94 66100000 MOV(10A,A1) \ NOP
A08 04D96 08F10000 MIN(A1,A5) \ NOP
A09 04D98 6D200000 MOV(A1),MAX(A1,A6) \ NOP
A10 04D9A 66310000 MOV(R,EX0) \ NOP
A11 04D9C 08980000 MOV(A1),ADD(A5,A1) \ NOP
A12 04D9E 41110000 MOV(R,M7) \ MOV(EX1,EX0)
A13 04DA0 84E00000 MUL(M1,M7) \ NOP
A14 04DA2 4F400000 SUB(A2,A7) \ NOP
A15 04DA4 08500000 MOV(EX1,A5) \ NOP
A16 04DA6 08500000 MOV(P,00) \ NOP
A17 04DA8 088C0000 MOV(R,A2) \ MOV(EX1,00)
A18 04DAA 08920000 JUMPC(TRAN,T1)
A19 04DAC 901E0000

INITIALIZATION
(01785) *
(01786) *
(01787) *
(01788) *
(01789) *
(01790) *
(01791) *
(01792) *
(01793) *
(01794) *
(01795) *
(01796) *
(01797) *
(01798) *

REGISTERS AFFECTED:
ADM1: A0,A1,A2,M0,M1,M3,M5,M6,M7
ADM2: A0,A2,A3,A4,A6,A7,M0,M2,M6

INPUT SEQUENCE: ENCD,BETA,RMS,STATE VARIABLE,UB*2**15/HL,RMSMIN,N-2.5
ALAD,SMIN,UB,G,1-ALP,ALP,AINV*(1/ALAD-1),2**14,PITCH REP

ADM1 *****
ADM2 *****
ADM1 *****
ADM2 *****

MOV(10A,A1) \ NOP
;A1=ENCD. BETA

DATA TRANSFER FROM SHAT BUFFER TO ADM BUFFER

REGISTERS USED:
ADM1: A0,A1,A2,A5,A6,A7,M1,M7,EX0
ADM2: A0,A1,A7,M1,M7,EX0

INPUT SEQUENCE: 2047/16**3,-2048/16**3,TRANSFER SIZE/2-0.5,SHAT(K-1),SHA
OUTPUT SEQUENCE: (SHAT(K-1)*SHAT(K))/2,SHAT(K)

;A5=2047/(16**3)
;A6=-2048/(16**3)
;A2=TRANSFER SIZE/2-0.5
;SHAT(K-1)
;LIMITING
;LIMITING

;LOOPING

```


PAGE 49: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 38, 1988

```

A6D 84E68 8588888C (81931)
A6E 84E62 88888894 (81932) *
      (81933) *
      (81934) *
      (81935) *
      (81936) *
      (81937) *
      (81938) *
      (81939) *
      (81940) *
      (81941) *
      (81942) *
      (81943) *
      (81944)
A6F 84E64 88F58849 (81944)
A7B 84E66 88A88888 (81945)
A71 84E68 888A8888 (81946)
A72 84E6A 852888EF (81947)
A73 84E6C 88818888 (81948)
A74 84E6E 88EA84E8 (81949)
A75 84E78 889C8888 (81950)
A76 84E72 885388ED (81951)
A77 84E74 416884A8 (81952)
A78 84E76 889C8888 (81953)
A79 84E78 88818888 (81954)
A7A 84E7A 88538828 (81955)
A7B 84E7C 41688888 (81956)
A7C 84E7E 889C8893 (81957)
      ADAP
A7D 84E88 88EA8888 (81958)
A7E 84E82 852888EF (81959)
A7F 84E84 888884E8 (81960)
A8B 84E86 88818888 (81961)
A81 84E88 88534F68 (81962)
A82 84E8A 41688888 (81963)
A83 84E8C 889C8893 (81964)
A84 84E8E 881F887D (81965) *
      (81966) *
      (81967) *
      (81968) *
      (81969) *
      (81970) *
      (81971) *
      (81972) *
      (81973) *
A85 84E98 888888F5 (81974) CKEND

      MOV(M2,M4) \ MOV(R,M4)
      MOV(P,EXO) \ MOV(R,A4)
      ADAPTATION
      REGISTERS AFFECTED:
      ADM1: A1,A3,A5,M2
      ADM2: A3,M1,M5,M7,EXO
      INPUT SEQUENCE: A(I), VHAT(K-1)
      OUTPUT SEQUENCE: A(I)
      MOV(IQA,A5) \ MOV(EXI,M1)
      ADD(A5,A8) \ NOP
      MOV(R,M2) \ NOP
      MUL(M2,M5) \ MOV(IQA,M7)
      MOV(P,A1) \ NOP
      MOV(IQA,M2) \ MUL(M1,M7)
      MUL(M2,M5) \ MOV(P,EXO)
      MOV(EXI,A3) \ MOV(IQA,M5)
      ADD(A3,A1) \ MUL(M1,M5)
      MOV(R,OQ) \ NOP
      MOV(P,A1) \ MOV(P,EXO)
      MOV(EXI,A3) \ R(A8)
      ADD(A3,A1) \ NOP
      MOV(R,OQ) \ MOV(R,A3)
      MOV(IQA,M2) \ NOP
      MUL(M2,M5) \ MOV(IQA,M7)
      NOP \ MUL(M1,M7)
      MOV(P,A1) \ MOV(P,EXO)
      MOV(EXI,A3) \ SUB(A3,A7)
      ADD(A3,A1) \ NOP
      MOV(R,OQ) \ MOV(R,A3)
      JUMPC(ADAP,T2)
      CHECK END OF RECEIVER
      REGISTERS AFFECTED:
      ADM1:
      ADM2: A5
      NOP \ MOV(IQA,A5)
      A5=L

```

```

      A5=A(1)
      R1=A(1)+ANIV*8.88787
      M2=R1
      P2=ALAD*R1
      M1=A(2)
      A1=P2
      A3=VHAT(K-1)*ERR
      R=NEW A(1)
      OUTPUT A(1)
      A1=ALAD*A(2)
      A3=VHAT(K-2)*ERR
      R=NEW A(2)
      OUTPUT A(2)
      M1=A(1)
      P=A(1)*ALAD
      A1=A(1)*ALAD
      A3=VHAT(K-1)*ALAD
      R=NEW A(1)
      OUTPUT NEW A(1)
      END ?

```

```

      P1=C*EQ/RMS**2
      M1=P1
      M7=VHAT(K-1)
      P3=VHAT(K-1)*P1
      EXO=P3
      M5=VHAT(K-2)
      P=VHAT(K-2)*ERR
      EXO=VHAT(K-2)*ERR
      R=N-2.5
      A3=N-2.5
      M7=VHAT(K-1)
      P=VHAT(K-1)*ERR
      EXO=VHAT(K-1)*ERR
      COUNTER=COUNTER-1

```


PAGE 54: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

A43 04F48 0699002A (02146)
A44 04F4A 00120001 (02147)
A45 04F4C 00A00002 (02148)
A46 04F4E 0C190000 (02149)
A47 04F50 0E99002A (02150)
A48 04F52 001A0004 (02151)
A49 04F54 02190000 (02152)
A4A 04F56 0419002A (02153)
      (02154) *
      (02155) *
      (02156) *
      (02157) *
      (02158) *
      (02159) *
      (02160) *
      (02161) *
      (02162) *
      (02163) *
      (02164) *
      (02165) *
      (02166) *
      (02167) *
      (02168) *
      (02169) *
      (02170) *
      (02171) *
      (02172) *
      (02173) *
      (02174) *
      (02175) *
      (02176) *
      (02177) *
      (02178) *
      (02179) *
      (02180) *
      (02181) *
      (02182) *
      (02183) *
      (02184) *
      (02185) *
      (02186) *
      (02187) *
      (02188) *
      (02189) *

A48 04F58 0644100A (02164)
A4C 04F5A 000001E9 (02165)
A4D 04F5C 0A004C08 (02166)
A4E 04F5E 0C200028 (02167)
A4F 04F60 0E0A0004 (02168)
A50 04F62 00004C60 (02169)
A51 04F64 020A0002 (02170)
A52 04F66 04200029 (02171)
A53 04F68 060A0002 (02172)
      (02173) *
      (02174) *
      (02175) *
      (02176) *
      (02177) *
      (02178) *
      (02179) *
      (02180) *
      (02181) *
      (02182) *
      (02183) *
      (02184) *
      (02185) *
      (02186) *
      (02187) *
      (02188) *
      (02189) *

A54 04F6A 003A0001 (02178)
A55 04F6C 0A390000 (02179)
A56 04F6E 0C09002A (02180)
A57 04F70 0E010013 (02181)
A58 04F72 00310039 (02182)
A59 04F74 02310000 (02183)
A5A 04F76 0401002A (02184)
      (02185) *
      (02186) *
      (02187) *
      (02188) *
      (02189) *

      ADD8(BR1,BV1,TF)
      SUB(BR1,1)
      ADD(BR0,2,TF)
      AND8(BR1,BV0)
      ADD8(BR1,BV1,TF)
      ADD(BR1,4)
      AND8(BR1,BV0)
      AND8(BR1,BV1)

      PRODUCING ADDRESSES FOR QUANTIZER

      CONTROLLED BY HANDSHAKE SIGNAL (AF0,AF1) FROM APU PROGRAM

      REGISTERS AFFECTED: BR0

      FLAGS AFFECTED: AF0,AF1
      FLAGS SENSED: AF0,AF1

      LOAD(BR0,4106(2),L)
      JUMPS(VQ3,AF1)
      JUMPC(VQ1,AF0)
      CLEAR(AF0)
      ADD(BR0,4)
      JUMP(VQ1)
      ADD(BR0,2,TF)
      CLEAR(AF1)
      ADD(BR0,2,TF)

      ADDRESSES OF SHAT(K-T),VHAT(K),SHAT(K)

      REGISTERS AFFECTED: BR3,BV2,BV3

      ADD(BR3,1)
      AND8(BR3,BV0)
      ADD8(BR3,BV1,TF)
      MOV8(BV2,BR1,TF)
      ADDL(BV3,1)
      AND8(BV3,BV0)
      ADD8(BV3,BV1,TF)

      ADAPTION

      USING CIRCULAR BUFFER

      ;ADDRESS OF EXPN(1) -6
      ;FIND CORRESPONDING QUANTIZING LEVEL
      ;WAIT FOR SIGNAL
      ;
      ;EXPN(3)
      ;OUT(3)
      ;SHAT(K-T)
      ;VHAT(K)
      ;
      ;SHAT(K)

```

PAGE 55: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

A58 04F70 06C41000 (02190) *
A59 04F71 06C41000 (02191) *
A60 04F72 06C41000 (02192) *
A61 04F73 06C41000 (02193) *
A62 04F74 06C41000 (02194) *
A63 04F75 06C41000 (02195) *
A64 04F76 06C41000 (02196) *
A65 04F77 06C41000 (02197) *
A66 04F78 06C41000 (02198) *
A67 04F79 06C41000 (02199) *
A68 04F80 06C41000 (02200) *
A69 04F81 06C41000 (02201) *
A70 04F82 06C41000 (02202) *
A71 04F83 06C41000 (02203) *
A72 04F84 06C41000 (02204) *
A73 04F85 06C41000 (02205) *
A74 04F86 06C41000 (02206) *
A75 04F87 06C41000 (02207) *
A76 04F88 06C41000 (02208) *
A77 04F89 06C41000 (02209) *
A78 04F90 06C41000 (02210) *
A79 04F91 06C41000 (02211) *
A80 04F92 06C41000 (02212) *
A81 04F93 06C41000 (02213) *
A82 04F94 06C41000 (02214) *
A83 04F95 06C41000 (02215) *
A84 04F96 06C41000 (02216) *
A85 04F97 06C41000 (02217) *
A86 04F98 06C41000 (02218) *
A87 04F99 06C41000 (02219) *
A88 04FA0 06C41000 (02220) *
A89 04FA1 06C41000 (02221) *
A90 04FA2 06C41000 (02222) *
A91 04FA3 06C41000 (02223) *
A92 04FA4 06C41000 (02224) *
A93 04FA5 06C41000 (02225) *
A94 04FA6 06C41000 (02226) *
A95 04FA7 06C41000 (02227) *
A96 04FA8 06C41000 (02228) *
A97 04FA9 06C41000 (02229) *
A98 04FAA 06C41000 (02230) *
A99 04FAB 06C41000 (02231) *
A00 04FAC 06C41000 (02232) *
A01 04FAE 06C41000 (02233) *

; STARTING ADDRESS OF A(1)
; OUTPUT A(1)
; INPUT L
; JUMP IF NOT FINISH
; WAIT FOR APU SIGNAL

REGISTERS AFFECTED: BR0,BR1,BV2
LOAD(BR0,4096(2),L,TF)
SUB(BR1,1)
ANDB(BR1,BV0)
ADDR(BR1,BV1,TF)
MOVB(BV2,BR0,TF)
SUB(BR1,1)
ADD(BR0,2,TF)
ANDB(BR1,BV0)
ADDR(BR1,BV1,TF)
ADDL(BV2,2,TF)
SUB(BR1,1)
ADD(BR0,2,TF)
ANDB(BR1,BV0)
ADDR(BR1,BV1,TF)
ADDL(BV2,2,TF)
SUB(BR1,1)
ADD(BR0,2,TF)
ANDB(BR1,BV0)
ADDR(BR1,BV1,TF)
ADDL(BV2,2,TF)
ADD(BR1,5)

END OF RECEIVER ?

REGISTERS AFFECTED: BR2
FLAGS SENSED: AF0,AF3
ADD(BR2,1,TF)
NOP
JUMPS(VPRE1,AF0)
JUMPC(VLOP2,AF3)

END OF RECEIVER ?

REGISTERS AFFECTED: BR0,BV3
FLAGS AFFECTED: AF3
LOAD(BR1,SVT111S(1),L,TF)
CLEAR(AF3)
LOAD(BR0,SVT99S(1),L)

```

```

PAGE 56:      MAP MODULES FOR THE P A R C ALGORITHM      --- JAN. 30, 1980

A77 04F00 EEB10011 (02234)      MOV8(BV3,BR,TF)      ;STATE VARIABLE
A78 04F02 F0B10032 (02235)      SUBL(BV3,2,TF)      ;RMS
A79 04F04 F21A0002 (02236)      ADD(BR1,2)      ;# OF SAMPLES
A7A 04F06 F5310013 (02237)      MOV8(BV3,BR1,TE)
A7B 04F08 F6200031 (02239)      CLEAR(RI)
A7C 04F0A F8000020 (02239)      NOP
A7D 04F0C FAD00060 (02240)      JUMP(B)
      (02241)
      (02242) VPCRCSA=0C      ;CHAIN ANCHOR
      (02243)
      (02244)      END
      (02245)      EVEN
      (02246)
      (02247)
      (02248)      STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
      (02249)
      (02250) VPCRC51 DATA 2F'B.B'
      ...
      (02251)
      (02252) VPCRC52=0L-VPCRC5      ;COMPUTE MODULE SIZE
      (02253)
      (02254)      TOP OF EXEC
      (02255)
      (02256)      TOES=0L
      (02257)      0L=TOESPR
      (02258)      ADDR      TOES(.BUS1S)
      (02259)      END
      (02260)
      (02261)      EVEN
      (02262)      EJECT

```

PAGE 57: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(02263) *
(02264) *
(02265) *
(02266) *
(02267) *
(02268) *
(02269) *
(02270) *
(02271) *
(02272) *
(02273) *
(02274) *
(02275) *
(02276) *
(02277) *
(02278) *
(02279) *
(02280) *
(02281) *
(02282) *
(02283) *
(02284) *
(02285) *
(02286) *
(02287) *
(02288) *
(02289) *
(02290) *
(02291) *
(02292) *
(02293) *
(02294) *
(02295) *
(02296) *
(02297) *
(02298) *
(02299) *
(02300) *
(02301) *
(02302) *
(02303) *
(02304) *
(02305) *
(02306) *

```

ARVIND S. ARORA

NAME: ENCLS FCB: 110

THIS PROGRAM TAKES A BLOCK OF QUANTIZER OUTPUTS FROM THE PARC ALGORITHM AND ENCODES THEM INTO A BIT STREAM FOR DIGITAL TRANSMISSION. EACH OUTPUT BLOCK HAS 189 BITS CONSISTING OF 1 SYNC BIT, 6 BITS FOR T, 7 BITS FOR BETA, 7 BITS FOR 'PHONEY' BETA IF APPLICABLE, 18 BITS FOR PARITY CHECK, AND THE REST (157 BITS) FOR QUANTIZER OUTPUT INFORMATION.

ENCODER CONSTRAINTS:

```

1-- THE NUMBER OF QUANTIZER OUTPUTS IN THE INPUT BUFFER MUST GENERATE NO
    MORE THAN 196 BITS.
1-- NONE OF THE BUFFERS IN THIS PROGRAM MUST OCCUPY MEMORY LOCATIONS
    HIGHER THAN 32K.
1-- THERE ARE SEVERAL CONSTRAINTS ON THE SOURCE CODE:
    1. THE CODE FOR LEVEL 1 IS 0.
    2. THE CODE FOR 'PHONEY BETA' IS 000,000.
    3. RUN LENGTH IS FIXED AT 14. (UNLESS LOCATION LSRUN IS MODIFIED)
    4. THE CODE 111,111 IS AN ALTERNATE NULL CODE.
    5. BECAUSE THE LSB OF THE CODE-WORD IS TRANSMITTED FIRST, THE RECEIVER
       GETS AN INVERTED CODE, EG. 110 --> 011. TO COMPENSATE FOR THIS,
       THE ENCODING TABLE MUST CONTAIN INVERTED CODES, RIGHT JUSTIFIED.
    6. LOCATION 0 IN THE ENCODING TABLE CONTAINS THE CODE FOR RUN LENGTH.
    7. LOCATION 2*12 IN THE ENCODING TABLE CONTAINS THE CODE FOR NULL CODE.
1-- ALTHOUGH THE CODES FOR Q-LEVELS ARE INVERTED BY THE ALGORITHM, THE
    CODES FOR T, BETA, 'PHONEY BETA' ARE NOT.
1-- THE VARIOUS INPUTS TO THIS PROGRAM NEED TO BE PROVIDED IN THE
    FOLLOWING FORMATS:
    N=2          FIXED FORMAT (ENCODED T)
    0<T<63      FIXED FORMAT (ENCODED T)
    0<BETA<96    FIXED FORMAT (ENCODED BETA)
    -VE NO.      FIXED FORMAT
    1. Q-LEVELS
    2. T
    3. BETA
    4. 'PHONEY BETA'

```

PAGE 58: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

FIXED FORMAT

-VE 12

(#2307) ; 5. END CODE

(#2308) ;

(#2309) ;

(#2310)

EJECT

```

(02311) ;
(02312) ;
(02313) ;
(02314) ;
(02315) ;
(02316) ;
(02317) ;
(02318) ;
(02319) ;
(02320) ;
(02321) ;
(02322) ;
(02323) ;
(02324) ;
(02325) ;
(02326) ;
(02327) ;
(02328) ;
(02329) ;
(02330) ;
(02331) ;
(02332) ;
(02333) ;
(02334) ;
(02335) ;
(02336) ;
(02337) ;
(02338) ;
(02339) ;
(02340) ;
(02341) ;
(02342) ;
(02343) ;
(02344) ;
(02345) ;
(02346) ;
(02347) ;
(02348) ;
(02349) ;
(02350) ;
(02351) ;
(02352) ;
(02353) ;
(02354) ;

```

THE FUNCTION CONTROL BLOCK IS AS FOLLOWS:

```

-----
|                                     |
| FCB # 110 | ISTB |
|-----|-----|
| OUTB | QHAT |
|-----|-----|
| BETA (TABLE) | ENCT (TABLE) |
|-----|-----|
| SCAL ID(RC UPDATE) |
|-----|
| INT SC ID(RC UPD) |
|-----|

```

*** NOTE: ENABLE LINES WITH * IN COL. 1 TO ASSEMBLE THIS PROGRAM BY ITSELF.

--- SYMBOL DEFINITION TO INTERFACE WITH SNAP-11 EXEC. REL 3.05

```

BCTSBA=$502
FDT$UFCB=$8C4
ISVTS=$502
SVSSFLG=$1FFCE
TOESOLD=$5000
TOESPTR=$200
SVTS=$302
--- TOP OF MEMORY POINTER
*OL=TOESPTR
ADDR TOESNEW(,1)
--- DISPATCH TABLE ENTRIES
*OL=FDT$UFCB
ADDR ENCD$

```

PAGE 68: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 38, 1988

```

(2355) ;
(2356) ;---- CSPU MODULE FOR THE ENCODER
(2357) ;
(2358) ;OL=TOESOLD
(2359) ;---- I. BASE ADDRESS FOR 4 BUFFERS AND INDEX FOR T.BETA
(2360) ;
(2361) ;---- INDEX FOR T.BETA
(2362) ;
(2363) ENCD5 MOVW R2,R1,S8FF ;SID FOR T.BETA
(2364) MOVW R2,I1D5TB ;SAVE IN LOC I1D5TB
(2365) ;
(2366) ;---- UPDATE RECEIVER
(2367) ;
(2368) ; CALL R7.RCS
(2369) ;
(2370) ;---- ADDRESS FOR QHAT
(2371) NOP
(2372) ;
(2373) ; INCR R1.1
(2374) MOVW R6,R1,S8FF ;BID FOR QHAT
(2375) CALL .BASES ;SUBROUTINE TO RETRIEVE BASE ADDR.
(2376) MOVW R4,AD5QHAT ;SAVE IN LOC AD5QHAT
(2377) ;
(2378) ;---- ADDRESS FOR OUTB
(2379) ;
(2380) MOVW R6,R1
(2381) LRS R6.8 ;BID FOR OUTB
(2382) CALL .BASES ;SUB. TO GET BASE ADDR.
(2383) MOVW R4,AD5OUTB ;SAVE IN LOC AD5OUTB
(2384) ADDR R5,S8FFD ;BASE ADDR + 189
(2385) MOVW R4,AD5OUTT ;SAVE IN LOC AD5OUTT 8. ADDR+189
(2386) ;
(2387) ;---- ADDRESS FOR ENCT, INDEXED BY R1
(2388) ;
(2389) INCR R1.1
(2390) MOVW R6,R1,S8FF ;BID FOR ENCT
(2391) CALL .BASES ;SUBROUTINE TO GET BASE ADDR.
(2392) IORIR R4,S8FFD2 ;INSERT INDEX REG.1
(2393) MOVW R4,AD5ENCT ;SAVE IN LOC AD5ENCT
(2394) ;
(2395) ;---- ADDRESS FOR BETA, INDEXED BY R1
(2396) ;
(2397) MOVW R6,R1

```

84FC2 782288FF
84FC4 E8285112

84FC6 2881
84FC7 2888

84FC8 2611
84FC9 786288FF
84FCB 868850FF
84FCD 84485186

84FCF 6862
84FDB 3C68
84FD1 868850FF
84FD3 84485188
84FD5 9C58888D
84FD7 84485118

84FDB 2611
84FDA 786288FF
84FDC 868850FF
84FDE 96488882
84FEB 8448518A

84FE2 6862

PAGE 63: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(02485) ;--- VIII. OUTPUT REST OF INFORMATION BITS
0504A F0905106 (02486) ENS08 MOVNR R1,0ADSQHAT
0504C 92100002 (02487) CMPIR R1,S0002
0504E 00205054 (02488) JMP ENS01,GTZ
0505B 00105068 (02489) JMP ENS83,EQZ
05052 00005082 (02490) JMP ENS90
05054 C000510A (02491) MOVNRL R4,0ADSENCT
05056 00100001 (02492) MOVNR R1,R5,S0001
05058 1010 (02493) EQZ
05059 447C (02494) SKP
0505A E0905108 (02495) XORR R7,R6
0505C 2661 (02496) MOVNR R1,0ADSOUTB
0505D E0005109 (02497) INCR R6,1
0505F 92600039 (02498) INCM ADSOUTB+1
05061 1020 (02499) CMPIR R6,S0039
05062 060050EF (02500) SKPL LEZ
05064 3C51 (02501) CALL .PAROUTS
05065 0C405056 (02502) LRS R5,1
05067 E5005107 (02503) DJP R4,ENS02
05069 0000504A (02504) INCM ADSQHAT+1
0506B 00005055 (02505) JMP ENS08
0506D F0205115 (02506) ;
0506E E5005107 (02507) ;--- RUN-LENGTH CODE
05070 003050A3 (02508) ;
05072 003050A3 (02509) ENS03 MOVNR R2,LSRUN
05074 92100002 (02510) ENS04 NEG R2
05076 00205092 (02511) MOVNR R1,0ADSQHAT+1
05078 0020506E (02512) INCM ADSQHAT+1
0507A 00205092 (02513) MOVNR R1,0ADSQHAT
0507C 00205092 (02514) JMP ENS88,LTZ
0507E 00205092 (02515) CMPIR R1,S0002
05080 00205092 (02516) JMP ENS86,GTZ
05082 00205092 (02517) IJM R2,ENS84
05084 00205092 (02518) ;--- OUTPUT RUN LENGTH CODE (AT LOC (ADSENCT)+0)
05086 00205092 (02519) ;
05088 00205092 (02520) ;
0508A 00205092 (02521) ;
0508C 00205092 (02522) ;
0508E 00205092 (02523) ;
05090 00205092 (02524) ;
05092 00205092 (02525) ;
05094 00205092 (02526) ;
05096 00205092 (02527) ;
05098 00205092 (02528) ;
0509A 00205092 (02529) ;
0509C 00205092 (02530) ;
0509E 00205092 (02531) ;
050A0 00205092 (02532) ;
050A2 00205092 (02533) ;
050A4 00205092 (02534) ;
050A6 00205092 (02535) ;
050A8 00205092 (02536) ;
050AA 00205092 (02537) ;
050AC 00205092 (02538) ;
050AE 00205092 (02539) ;
050B0 00205092 (02540) ;
050B2 00205092 (02541) ;
050B4 00205092 (02542) ;
050B6 00205092 (02543) ;
050B8 00205092 (02544) ;
050BA 00205092 (02545) ;
050BC 00205092 (02546) ;
050BE 00205092 (02547) ;
050C0 00205092 (02548) ;
050C2 00205092 (02549) ;
050C4 00205092 (02550) ;
050C6 00205092 (02551) ;
050C8 00205092 (02552) ;
050CA 00205092 (02553) ;
050CC 00205092 (02554) ;
050CE 00205092 (02555) ;
050D0 00205092 (02556) ;
050D2 00205092 (02557) ;
050D4 00205092 (02558) ;
050D6 00205092 (02559) ;
050D8 00205092 (02560) ;
050DA 00205092 (02561) ;
050DC 00205092 (02562) ;
050DE 00205092 (02563) ;
050E0 00205092 (02564) ;
050E2 00205092 (02565) ;
050E4 00205092 (02566) ;
050E6 00205092 (02567) ;
050E8 00205092 (02568) ;
050EA 00205092 (02569) ;
050EC 00205092 (02570) ;
050EE 00205092 (02571) ;
050F0 00205092 (02572) ;
050F2 00205092 (02573) ;
050F4 00205092 (02574) ;
050F6 00205092 (02575) ;
050F8 00205092 (02576) ;
050FA 00205092 (02577) ;
050FC 00205092 (02578) ;
050FE 00205092 (02579) ;
050FF 00205092 (02580) ;
05100 00205092 (02581) ;
05102 00205092 (02582) ;
05104 00205092 (02583) ;
05106 00205092 (02584) ;
05108 00205092 (02585) ;
0510A 00205092 (02586) ;
0510C 00205092 (02587) ;
0510E 00205092 (02588) ;
05110 00205092 (02589) ;
05112 00205092 (02590) ;
05114 00205092 (02591) ;
05116 00205092 (02592) ;
05118 00205092 (02593) ;
0511A 00205092 (02594) ;
0511C 00205092 (02595) ;
0511E 00205092 (02596) ;
05120 00205092 (02597) ;
05122 00205092 (02598) ;
05124 00205092 (02599) ;
05126 00205092 (02600) ;
05128 00205092 (02601) ;
0512A 00205092 (02602) ;
0512C 00205092 (02603) ;
0512E 00205092 (02604) ;
05130 00205092 (02605) ;
05132 00205092 (02606) ;
05134 00205092 (02607) ;
05136 00205092 (02608) ;
05138 00205092 (02609) ;
0513A 00205092 (02610) ;
0513C 00205092 (02611) ;
0513E 00205092 (02612) ;
05140 00205092 (02613) ;
05142 00205092 (02614) ;
05144 00205092 (02615) ;
05146 00205092 (02616) ;
05148 00205092 (02617) ;
0514A 00205092 (02618) ;
0514C 00205092 (02619) ;
0514E 00205092 (02620) ;
05150 00205092 (02621) ;
05152 00205092 (02622) ;
05154 00205092 (02623) ;
05156 00205092 (02624) ;
05158 00205092 (02625) ;
0515A 00205092 (02626) ;
0515C 00205092 (02627) ;
0515E 00205092 (02628) ;
05160 00205092 (02629) ;
05162 00205092 (02630) ;
05164 00205092 (02631) ;
05166 00205092 (02632) ;
05168 00205092 (02633) ;
0516A 00205092 (02634) ;
0516C 00205092 (02635) ;
0516E 00205092 (02636) ;
05170 00205092 (02637) ;
05172 00205092 (02638) ;
05174 00205092 (02639) ;
05176 00205092 (02640) ;
05178 00205092 (02641) ;
0517A 00205092 (02642) ;
0517C 00205092 (02643) ;
0517E 00205092 (02644) ;
05180 00205092 (02645) ;
05182 00205092 (02646) ;
05184 00205092 (02647) ;
05186 00205092 (02648) ;
05188 00205092 (02649) ;
0518A 00205092 (02650) ;
0518C 00205092 (02651) ;
0518E 00205092 (02652) ;
05190 00205092 (02653) ;
05192 00205092 (02654) ;
05194 00205092 (02655) ;
05196 00205092 (02656) ;
05198 00205092 (02657) ;
0519A 00205092 (02658) ;
0519C 00205092 (02659) ;
0519E 00205092 (02660) ;
051A0 00205092 (02661) ;
051A2 00205092 (02662) ;
051A4 00205092 (02663) ;
051A6 00205092 (02664) ;
051A8 00205092 (02665) ;
051AA 00205092 (02666) ;
051AC 00205092 (02667) ;
051AE 00205092 (02668) ;
051B0 00205092 (02669) ;
051B2 00205092 (02670) ;
051B4 00205092 (02671) ;
051B6 00205092 (02672) ;
051B8 00205092 (02673) ;
051BA 00205092 (02674) ;
051BC 00205092 (02675) ;
051BE 00205092 (02676) ;
051C0 00205092 (02677) ;
051C2 00205092 (02678) ;
051C4 00205092 (02679) ;
051C6 00205092 (02680) ;
051C8 00205092 (02681) ;
051CA 00205092 (02682) ;
051CC 00205092 (02683) ;
051CE 00205092 (02684) ;
051D0 00205092 (02685) ;
051D2 00205092 (02686) ;
051D4 00205092 (02687) ;
051D6 00205092 (02688) ;
051D8 00205092 (02689) ;
051DA 00205092 (02690) ;
051DC 00205092 (02691) ;
051DE 00205092 (02692) ;
051E0 00205092 (02693) ;
051E2 00205092 (02694) ;
051E4 00205092 (02695) ;
051E6 00205092 (02696) ;
051E8 00205092 (02697) ;
051EA 00205092 (02698) ;
051EC 00205092 (02699) ;
051EE 00205092 (02700) ;
051F0 00205092 (02701) ;
051F2 00205092 (02702) ;
051F4 00205092 (02703) ;
051F6 00205092 (02704) ;
051F8 00205092 (02705) ;
051FA 00205092 (02706) ;
051FC 00205092 (02707) ;
051FE 00205092 (02708) ;
05200 00205092 (02709) ;
05202 00205092 (02710) ;
05204 00205092 (02711) ;
05206 00205092 (02712) ;
05208 00205092 (02713) ;
0520A 00205092 (02714) ;
0520C 00205092 (02715) ;
0520E 00205092 (02716) ;
05210 00205092 (02717) ;
05212 00205092 (02718) ;
05214 00205092 (02719) ;
05216 00205092 (02720) ;
05218 00205092 (02721) ;
0521A 00205092 (02722) ;
0521C 00205092 (02723) ;
0521E 00205092 (02724) ;
05220 00205092 (02725) ;
05222 00205092 (02726) ;
05224 00205092 (02727) ;
05226 00205092 (02728) ;
05228 00205092 (02729) ;
0522A 00205092 (02730) ;
0522C 00205092 (02731) ;
0522E 00205092 (02732) ;
05230 00205092 (02733) ;
05232 00205092 (02734) ;
05234 00205092 (02735) ;
05236 00205092 (02736) ;
05238 00205092 (02737) ;
0523A 00205092 (02738) ;
0523C 00205092 (02739) ;
0523E 00205092 (02740) ;
05240 00205092 (02741) ;
05242 00205092 (02742) ;
05244 00205092 (02743) ;
05246 00205092 (02744) ;
05248 00205092 (02745) ;
0524A 00205092 (02746) ;
0524C 00205092 (02747) ;
0524E 00205092 (02748) ;
05250 00205092 (02749) ;
05252 00205092 (02750) ;
05254 00205092 (02751) ;
05256 00205092 (02752) ;
05258 00205092 (02753) ;
0525A 00205092 (02754) ;
0525C 00205092 (02755) ;
0525E 00205092 (02756) ;
05260 00205092 (02757) ;
05262 00205092 (02758) ;
05264 00205092 (02759) ;
05266 00205092 (02760) ;
05268 00205092 (02761) ;
0526A 00205092 (02762) ;
0526C 00205092 (02763) ;
0526E 00205092 (02764) ;
05270 00205092 (02765) ;
05272 00205092 (02766) ;
05274 00205092 (02767) ;
05276 00205092 (02768) ;
05278 00205092 (02769) ;
0527A 00205092 (02770) ;
0527C 00205092 (02771) ;
0527E 00205092 (02772) ;
05280 00205092 (02773) ;
05282 00205092 (02774) ;
05284 00205092 (02775) ;
05286 00205092 (02776) ;
05288 00205092 (02777) ;
0528A 00205092 (02778) ;
0528C 00205092 (02779) ;
0528E 00205092 (02780) ;
05290 00205092 (02781) ;
05292 00205092 (02782) ;
05294 00205092 (02783) ;
05296 00205092 (02784) ;
05298 00205092 (02785) ;
0529A 00205092 (02786) ;
0529C 00205092 (02787) ;
0529E 00205092 (02788) ;
052A0 00205092 (02789) ;
052A2 00205092 (02790) ;
052A4 00205092 (02791) ;
052A6 00205092 (02792) ;
052A8 00205092 (02793) ;
052AA 00205092 (02794) ;
052AC 00205092 (02795) ;
052AE 00205092 (02796) ;
052B0 00205092 (02797) ;
052B2 00205092 (02798) ;
052B4 00205092 (02799) ;
052B6 00205092 (02800) ;
052B8 00205092 (02801) ;
052BA 00205092 (02802) ;
052BC 00205092 (02803) ;
052BE 00205092 (02804) ;
052C0 00205092 (02805) ;
052C2 00205092 (02806) ;
052C4 00205092 (02807) ;
052C6 00205092 (02808) ;
052C8 00205092 (02809) ;
052CA 00205092 (02810) ;
052CC 00205092 (02811) ;
052CE 00205092 (02812) ;
052D0 00205092 (02813) ;
052D2 00205092 (02814) ;
052D4 00205092 (02815) ;
052D6 00205092 (02816) ;
052D8 00205092 (02817) ;
052DA 00205092 (02818) ;
052DC 00205092 (02819) ;
052DE 00205092 (02820) ;
052E0 00205092 (02821) ;
052E2 00205092 (02822) ;
052E4 00205092 (02823) ;
052E6 00205092 (02824) ;
052E8 00205092 (02825) ;
052EA 00205092 (02826) ;
052EC 00205092 (02827) ;
052EE 00205092 (02828) ;
052F0 00205092 (02829) ;
052F2 00205092 (02830) ;
052F4 00205092 (02831) ;
052F6 00205092 (02832) ;
052F8 00205092 (02833) ;
052FA 00205092 (02834) ;
052FC 00205092 (02835) ;
052FE 00205092 (02836) ;
05300 00205092 (02837) ;
05302 00205092 (02838) ;
05304 00205092 (02839) ;
05306 00205092 (02840) ;
05308 00205092 (02841) ;
0530A 00205092 (02842) ;
0530C 00205092 (02843) ;
0530E 00205092 (02844) ;
05310 00205092 (02845) ;
05312 00205092 (02846) ;
05314 00205092 (02847) ;
05316 00205092 (02848) ;
05318 00205092 (02849) ;
0531A 00205092 (02850) ;
0531C 00205092 (02851) ;
0531E 00205092 (02852) ;
05320 00205092 (02853) ;
05322 00205092 (02854) ;
05324 00205092 (02855) ;
05326 00205092 (02856) ;
05328 00205092 (02857) ;
0532A 00205092 (02858) ;
0532C 00205092 (02859) ;
0532E 00205092 (02860) ;
05330 00205092 (02861) ;
05332 00205092 (02862) ;
05334 00205092 (02863) ;
05336 00205092 (02864) ;
05338 00205092 (02865) ;
0533A 00205092 (02866) ;
0533C 00205092 (02867) ;
0533E 00205092 (02868) ;
05340 00205092 (02869) ;
05342 00205092 (02870) ;
05344 00205092 (02871) ;
05346 00205092 (02872) ;
05348 00205092 (02873) ;
0534A 00205092 (02874) ;
0534C 00205092 (02875) ;
0534E 00205092 (02876) ;
05350 00205092 (02877) ;
05352 00205092 (02878) ;
05354 00205092 (02879) ;
05356 00205092 (02880) ;
05358 00205092 (02881) ;
0535A 00205092 (02882) ;
0535C 00205092 (02883) ;
0535E 00205092 (02884) ;
05360 00205092 (02885) ;
05362 00205092 (02886) ;
05364 00205092 (02887) ;
05366 00205092 (02888) ;
05368 00205092 (02889) ;
0536A 00205092 (02890) ;
0536C 00205092 (02891) ;
0536E 00205092 (02892) ;
05370 00205092 (02893) ;
05372 00205092 (02894) ;
05374 00205092 (02895) ;
05376 00205092 (02896) ;
05378 00205092 (02897) ;
0537A 00205092 (02898) ;
0537C 00205092 (02899) ;
0537E 00205092 (02900) ;
05380 00205092 (02901) ;
05382 00205092 (02902) ;
05384 00205092 (02903) ;
05386 00205092 (02904) ;
05388 00205092 (02905) ;
0538A 00205092 (02906) ;
0538C 00205092 (02907) ;
0538E 00205092 (02908) ;
05390 00205092 (02909) ;
05392 00205092 (02910) ;
05394 00205092 (02911) ;
05396 00205092 (02912) ;
05398 00205092 (02913) ;
0539A 00205092 (02914) ;
0539C 00205092 (02915) ;
0539E 00205092 (02916) ;
053A0 00205092 (02917) ;
053A2 00205092 (02918) ;
053A4 00205092 (02919) ;
053A6 00205092 (02920) ;
053A8 00205092 (02921) ;
053AA 00205092 (02922) ;
053AC 00205092 (02923) ;
053AE 00205092 (02924) ;
053B0 00205092 (02925) ;
053B2 00205092 (02926) ;
053B4 00205092 (02927) ;
053B6 00205092 (02928) ;
053B8 00205092 (02929) ;
053BA 00205092 (02930) ;
053BC 00205092 (02931) ;
053BE 00205092 (02932) ;
053C0 00205092 (02933) ;
053C2 00205092 (02934) ;
053C4 00205092 (02935) ;
053C6 00205092 (02936) ;
053C8 00205092 (02937) ;
053CA 00205092 (02938) ;
053CC 00205092 (02939) ;
053CE 00205092 (02940) ;
053D0 00205092 (02941) ;
053D2 00205092 (02942) ;
053D4 00205092 (02943) ;
053D6 00205092 (02944) ;
053D8 00205092 (02945) ;
053DA 00205092 (02946) ;
053DC 00205092 (02947) ;
053DE 00205092 (02948) ;
053E0 00205092 (02949) ;
053E2 00205092 (02950) ;
053E4 00205092 (02951) ;
053E6 00205092 (02952) ;
053E8 00205092 (02953) ;
053EA 00205092 (02954) ;
053EC 00205092 (02955) ;
053EE 00205092 (02956) ;
053F0 00205092 (02957) ;
053F2 00205092 (02958) ;
053F4 00205092 (02959) ;
053F6 00205092 (02960) ;
053F8 00205092 (02961) ;
053FA 00205092 (02962) ;
053FC 00205092 (02963) ;
053FE 00205092 (02964) ;
05400 00205092 (02965) ;
05402 00205092 (02966) ;
05404 00205092 (02967) ;
05406 00205092 (02968) ;
05408 00205092 (02969) ;
0540A 00205092 (02970) ;
0540C 00205092 (02971) ;
0540E 00205092 (02972) ;
05410 00205092 (02973) ;
05412 00205092 (02974) ;
05414 00205092 (02975) ;
05416 00205092 (02976) ;
05418 00205092 (02977) ;
0541A 00205092 (02978) ;
0541C 00205092 (02979) ;
0541E 00205092 (02980) ;
05420 00205092 (02981) ;
05422 00205092 (02982) ;
05424 00205092 (02983) ;
05426 00205092 (02984) ;
05428 00205092 (02985) ;
0542A 00205092 (02986) ;
0542C 00205092 (02987) ;
0542E 00205092 (02988) ;
05430 00205092 (02989) ;
05432 00205092 (02990) ;
05434 00205092 (02991) ;
05436 00205092 (02992) ;
05438 00205092 (02993) ;
0543A 00205092 (02994) ;
0543C 00205092 (02995) ;
0543E 00205092 (02996) ;
05440 00205092 (02997) ;
05442 00205092 (02998) ;
05444 00205092 (02999) ;
05446 00205092 (03000) ;
05448 00205092 (03001) ;
0544A 00205092 (03002) ;
0544C 00205092 (03003) ;
0544E 00205092 (03004) ;
05450 00205092 (03005) ;
05452 00205092 (03006) ;
05454 00205092 (03007) ;
05456 00205092 (03008) ;
05458 00205092 (03009) ;
0545A 00205092 (03010) ;
0545C 00205092 (03011) ;
0545E 00205092 (03012) ;
05460 00205092 (03013) ;
05462 00205092 (03014) ;
```

--- JAN. 30, 1980

ADDRESS	INSTR	OP	DATA	COMMENT
00000	182B	SKPL		IF 57 INFO BITS, OUT 6 PARITY BITS
00001	06005DEF	CALL		NEXT BIT TO LSB
00002	3C51	LRS		
00003	0C405B7D	DJP		
00004				
00005	E5005107	INCH		INCR I-INDEX
00006	0000550A	JMP		
00007				
00008				
00009	FC205115	ADDMR		1-CNTR=1-CNTR+13 (0 OF B'S, +VE INDEX)
00010	0D50	CLR		CODE FOR Q=1 IS B, R5=B
00011	E0D0510B	MOVHR		OUTPUT
00012	5097 2661	INCR		INCR H-INDEX
00013	E5005109	INCH		INCR O-INDEX
00014	92600B39	CMPIR		DEL=H-IND -57
00015	509C 182B	SKPL		
00016	06005DEF	CALL		IF 57 INFO BITS, OUTPUT 6 PARITY BITS
00017	0C205B95	DJP		
00018				
00019	00005554	JMP		
00020				
00021	FC205115	ADDMR		1-CNTR=1-CNTR+13 (0 OF B'S, +VE INDEX)
00022	0D50	CLR		Q=1, CODE IS B, R5=B
00023	E0D0510B	MOVHR		OUTPUT
00024	5097 2661	INCR		INCR H-INDEX
00025	E5005109	INCH		INCR O-INDEX
00026	92600B39	CMPIR		DEL=H-IND -57
00027	509C 182B	SKPL		
00028	06005DEF	CALL		IF 57 INFO BITS, PUT OUT 6 PARITY BITS
00029	0C205B95	DJP		
00030				
00031	00005554	JMP		
00032				
00033	FC205115	ADDMR		1-CNTR=1-CNTR+13 (0 OF B'S, +VE INDEX)
00034	0D50	CLR		Q=1, CODE IS B, R5=B
00035	E0D0510B	MOVHR		OUTPUT
00036	5097 2661	INCR		INCR H-INDEX
00037	E5005109	INCH		INCR O-INDEX
00038	92600B39	CMPIR		DEL=H-IND -57
00039	509C 182B	SKPL		
00040	06005DEF	CALL		IF 57 INFO BITS, PUT OUT 6 PARITY BITS
00041	0C205B95	DJP		
00042				
00043	00005554	JMP		
00044				
00045	FC205115	ADDMR		1-CNTR=1-CNTR+13 (0 OF B'S, +VE INDEX)
00046	0D50	CLR		Q=1, CODE IS B, R5=B
00047	E0D0510B	MOVHR		OUTPUT
00048	5097 2661	INCR		INCR H-INDEX
00049	E5005109	INCH		INCR O-INDEX
00050	92600B39	CMPIR		DEL=H-IND -57
00051	509C 182B	SKPL		
00052	06005DEF	CALL		IF 57 INFO BITS, PUT OUT 6 PARITY BITS
00053	0C205B95	DJP		
00054				
00055	00005554	JMP		
00056				
00057	FC205115	ADDMR		1-CNTR=1-CNTR+13 (0 OF B'S, +VE INDEX)
00058	0D50	CLR		Q=1, CODE IS B, R5=B
00059	E0D0510B	MOVHR		OUTPUT
00060	5097 2661	INCR		INCR H-INDEX
00061	E5005109	INCH		INCR O-INDEX
00062	92600B39	CMPIR		DEL=H-IND -57
00063	509C 182B	SKPL		
00064	06005DEF	CALL		IF 57 INFO BITS, PUT OUT 6 PARITY BITS
00065	0C205B95	DJP		
00066				
00067	00005554	JMP		
00068				
00069	FC205115	ADDMR		1-CNTR=1-CNTR+13 (0 OF B'S, +VE INDEX)
00070	0D50	CLR		Q=1, CODE IS B, R5=B
00071	E0D0510B	MOVHR		OUTPUT
00072	5097 2661	INCR		INCR H-INDEX
00073	E5005109	INCH		INCR O-INDEX
00074	92600B39	CMPIR		DEL=H-IND -57
00075	509C 182B	SKPL		
00076	06005DEF			

PAGE 65: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

R500E 1010 (R2573) SKP EQZ
R500F 447C (R2574) XORR R7,R6
R500G E0005100 (R2575) MOVRM R1,0ADSOUTS
R500H 2661 (R2576) INCR R6,1
R500I E5005100 (R2577) INCM ADSOUTS+1
R500J 92000039 (R2578) CMPR R6,S0039
R500K 1020 (R2579) SKPL LEZ
R500L 060050EF (R2580) CALL .PAROUTS
R500M 3C51 (R2581) LRS R5,1
R500N 8C4050BC (R2582) DJP R4,ENS91
R500O 1010 (R2583)
R500P 447C (R2584)
R500Q E0005100 (R2585)
R500R FE405111 (R2586) ENS100
R500S 813050E4 (R2587)
R500T 90005001 (R2588)
R500U 447C (R2589)
R500V E5005100 (R2590)
R500W 92000039 (R2591)
R500X 812050E2 (R2592)
R500Y 060050EF (R2593)
R500Z 2646 (R2594)
R500A 894050D5 (R2595)
R500B 1010 (R2596)
R500C 447C (R2597)
R500D E5005100 (R2598)
R500E 92000039 (R2599)
R500F 812050E2 (R2600)
R500G 060050EF (R2601)
R500H 2646 (R2602)
R500I 894050D5 (R2603)
R500J 1010 (R2604)
R500K 447C (R2605)
R500L E5005100 (R2606)
R500M 92000039 (R2607)
R500N 812050E2 (R2608)
R500O 060050EF (R2609)
R500P 2646 (R2610)
R500Q 894050D5 (R2611)
R500R 1010 (R2612)
R500S 447C (R2613)
R500T E5005100 (R2614)
R500U 92000039 (R2615)
R500V 812050E2 (R2616)
R500W 060050EF (R2617)
R500X 2646 (R2618)
R500Y 894050D5 (R2619)
R500Z 1010 (R2620)
R500A 447C (R2621)
R500B E5005100 (R2622)
R500C 92000039 (R2623)
R500D 812050E2 (R2624)
R500E 060050EF (R2625)
R500F 2646 (R2626)
R500G 894050D5 (R2627)
R500H 1010 (R2628)
R500I 447C (R2629)
R500J E5005100 (R2630)
R500K 92000039 (R2631)
R500L 812050E2 (R2632)
R500M 060050EF (R2633)
R500N 2646 (R2634)
R500O 894050D5 (R2635)
R500P 1010 (R2636)
R500Q 447C (R2637)
R500R E5005100 (R2638)
R500S 92000039 (R2639)
R500T 812050E2 (R2640)
R500U 060050EF (R2641)
R500V 2646 (R2642)
R500W 894050D5 (R2643)
R500X 1010 (R2644)
R500Y 447C (R2645)
R500Z E5005100 (R2646)
R500A 92000039 (R2647)
R500B 812050E2 (R2648)
R500C 060050EF (R2649)
R500D 2646 (R2650)
R500E 894050D5 (R2651)
R500F 1010 (R2652)
R500G 447C (R2653)
R500H E5005100 (R2654)
R500I 92000039 (R2655)
R500J 812050E2 (R2656)
R500K 060050EF (R2657)
R500L 2646 (R2658)
R500M 894050D5 (R2659)
R500N 1010 (R2660)
R500O 447C (R2661)
R500P E5005100 (R2662)
R500Q 92000039 (R2663)
R500R 812050E2 (R2664)
R500S 060050EF (R2665)
R500T 2646 (R2666)
R500U 894050D5 (R2667)
R500V 1010 (R2668)
R500W 447C (R2669)
R500X E5005100 (R2670)
R500Y 92000039 (R2671)
R500Z 812050E2 (R2672)
R500A 060050EF (R2673)
R500B 2646 (R2674)
R500C 894050D5 (R2675)
R500D 1010 (R2676)
R500E 447C (R2677)
R500F E5005100 (R2678)
R500G 92000039 (R2679)
R500H 812050E2 (R2680)
R500I 060050EF (R2681)
R500J 2646 (R2682)
R500K 894050D5 (R2683)
R500L 1010 (R2684)
R500M 447C (R2685)
R500N E5005100 (R2686)
R500O 92000039 (R2687)
R500P 812050E2 (R2688)
R500Q 060050EF (R2689)
R500R 2646 (R2690)
R500S 894050D5 (R2691)
R500T 1010 (R2692)
R500U 447C (R2693)
R500V E5005100 (R2694)
R500W 92000039 (R2695)
R500X 812050E2 (R2696)
R500Y 060050EF (R2697)
R500Z 2646 (R2698)
R500A 894050D5 (R2699)
R500B 1010 (R2700)
R500C 447C (R2701)
R500D E5005100 (R2702)
R500E 92000039 (R2703)
R500F 812050E2 (R2704)
R500G 060050EF (R2705)
R500H 2646 (R2706)
R500I 894050D5 (R2707)
R500J 1010 (R2708)
R500K 447C (R2709)
R500L E5005100 (R2710)
R500M 92000039 (R2711)
R500N 812050E2 (R2712)
R500O 060050EF (R2713)
R500P 2646 (R2714)
R500Q 894050D5 (R2715)
R500R 1010 (R2716)
R500S 447C (R2717)
R500T E5005100 (R2718)
R500U 92000039 (R2719)
R500V 812050E2 (R2720)
R500W 060050EF (R2721)
R500X 2646 (R2722)
R500Y 894050D5 (R2723)
R500Z 1010 (R2724)
R500A 447C (R2725)
R500B E5005100 (R2726)
R500C 92000039 (R2727)
R500D 812050E2 (R2728)
R500E 060050EF (R2729)
R500F 2646 (R2730)
R500G 894050D5 (R2731)
R500H 1010 (R2732)
R500I 447C (R2733)
R500J E5005100 (R2734)
R500K 92000039 (R2735)
R500L 812050E2 (R2736)
R500M 060050EF (R2737)
R500N 2646 (R2738)
R500O 894050D5 (R2739)
R500P 1010 (R2740)
R500Q 447C (R2741)
R500R E5005100 (R2742)
R500S 92000039 (R2743)
R500T 812050E2 (R2744)
R500U 060050EF (R2745)
R500V 2646 (R2746)
R500W 894050D5 (R2747)
R500X 1010 (R2748)
R500Y 447C (R2749)
R500Z E5005100 (R2750)
R500A 92000039 (R2751)
R500B 812050E2 (R2752)
R500C 060050EF (R2753)
R500D 2646 (R2754)
R500E 894050D5 (R2755)
R500F 1010 (R2756)
R500G 447C (R2757)
R500H E5005100 (R2758)
R500I 92000039 (R2759)
R500J 812050E2 (R2760)
R500K 060050EF (R2761)
R500L 2646 (R2762)
R500M 894050D5 (R2763)
R500N 1010 (R2764)
R500O 447C (R2765)
R500P E5005100 (R2766)
R500Q 92000039 (R2767)
R500R 812050E2 (R2768)
R500S 060050EF (R2769)
R500T 2646 (R2770)
R500U 894050D5 (R2771)
R500V 1010 (R2772)
R500W 447C (R2773)
R500X E5005100 (R2774)
R500Y 92000039 (R2775)
R500Z 812050E2 (R2776)
R500A 060050EF (R2777)
R500B 2646 (R2778)
R500C 894050D5 (R2779)
R500D 1010 (R2780)
R500E 447C (R2781)
R500F E5005100 (R2782)
R500G 92000039 (R2783)
R500H 812050E2 (R2784)
R500I 060050EF (R2785)
R500J 2646 (R2786)
R500K 894050D5 (R2787)
R500L 1010 (R2788)
R500M 447C (R2789)
R500N E5005100 (R2790)
R500O 92000039 (R2791)
R500P 812050E2 (R2792)
R500Q 060050EF (R2793)
R500R 2646 (R2794)
R500S 894050D5 (R2795)
R500T 1010 (R2796)
R500U 447C (R2797)
R500V E5005100 (R2798)
R500W 92000039 (R2799)
R500X 812050E2 (R2800)
R500Y 060050EF (R2801)
R500Z 2646 (R2802)
R500A 894050D5 (R2803)
R500B 1010 (R2804)
R500C 447C (R2805)
R500D E5005100 (R2806)
R500E 92000039 (R2807)
R500F 812050E2 (R2808)
R500G 060050EF (R2809)
R500H 2646 (R2810)
R500I 894050D5 (R2811)
R500J 1010 (R2812)
R500K 447C (R2813)
R500L E5005100 (R2814)
R500M 92000039 (R2815)
R500N 812050E2 (R2816)
R500O 060050EF (R2817)
R500P 2646 (R2818)
R500Q 894050D5 (R2819)
R500R 1010 (R2820)
R500S 447C (R2821)
R500T E5005100 (R2822)
R500U 92000039 (R2823)
R500V 812050E2 (R2824)
R500W 060050EF (R2825)
R500X 2646 (R2826)
R500Y 894050D5 (R2827)
R500Z 1010 (R2828)
R500A 447C (R2829)
R500B E5005100 (R2830)
R500C 92000039 (R2831)
R500D 812050E2 (R2832)
R500E 060050EF (R2833)
R500F 2646 (R2834)
R500G 894050D5 (R2835)
R500H 1010 (R2836)
R500I 447C (R2837)
R500J E5005100 (R2838)
R500K 92000039 (R2839)
R500L 812050E2 (R2840)
R500M 060050EF (R2841)
R500N 2646 (R2842)
R500O 894050D5 (R2843)
R500P 1010 (R2844)
R500Q 447C (R2845)
R500R E5005100 (R2846)
R500S 92000039 (R2847)
R500T 812050E2 (R2848)
R500U 060050EF (R2849)
R500V 2646 (R2850)
R500W 894050D5 (R2851)
R500X 1010 (R2852)
R500Y 447C (R2853)
R500Z E5005100 (R2854)
R500A 92000039 (R2855)
R500B 812050E2 (R2856)
R500C 060050EF (R2857)
R500D 2646 (R2858)
R500E 894050D5 (R2859)
R500F 1010 (R2860)
R500G 447C (R2861)
R500H E5005100 (R2862)
R500I 92000039 (R2863)
R500J 812050E2 (R2864)
R500K 060050EF (R2865)
R500L 2646 (R2866)
R500M 894050D5 (R2867)
R500N 1010 (R2868)
R500O 447C (R2869)
R500P E5005100 (R2870)
R500Q 92000039 (R2871)
R500R 812050E2 (R2872)
R500S 060050EF (R2873)
R500T 2646 (R2874)
R500U 894050D5 (R2875)
R500V 1010 (R2876)
R500W 447C (R2877)
R500X E5005100 (R2878)
R500Y 92000039 (R2879)
R500Z 812050E2 (R2880)
R500A 060050EF (R2881)
R500B 2646 (R2882)
R500C 894050D5 (R2883)
R500D 1010 (R2884)
R500E 447C (R2885)
R500F E5005100 (R2886)
R500G 92000039 (R2887)
R500H 812050E2 (R2888)
R500I 060050EF (R2889)
R500J 2646 (R2890)
R500K 894050D5 (R2891)
R500L 1010 (R2892)
R500M 447C (R2893)
R500N E5005100 (R2894)
R500O 92000039 (R2895)
R500P 812050E2 (R2896)
R500Q 060050EF (R2897)
R500R 2646 (R2898)
R500S 894050D5 (R2899)
R500T 1010 (R2900)
R500U 447C (R2901)
R500V E5005100 (R2902)
R500W 92000039 (R2903)
R500X 812050E2 (R2904)
R500Y 060050EF (R2905)
R500Z 2646 (R2906)
R500A 894050D5 (R2907)
R500B 1010 (R2908)
R500C 447C (R2909)
R500D E5005100 (R2910)
R500E 92000039 (R2911)
R500F 812050E2 (R2912)
R500G 060050EF (R2913)
R500H 2646 (R2914)
R500I 894050D5 (R2915)
R500J 1010 (R2916)
R500K 447C (R2917)
R500L E5005100 (R2918)
R500M 92000039 (R2919)
R500N 812050E2 (R2920)
R500O 060050EF (R2921)
R500P 2646 (R2922)
R500Q 894050D5 (R2923)
R500R 1010 (R2924)
R500S 447C (R2925)
R500T E5005100 (R2926)
R500U 92000039 (R2927)
R500V 812050E2 (R2928)
R500W 060050EF (R2929)
R500X 2646 (R2930)
R500Y 894050D5 (R2931)
R500Z 1010 (R2932)
R500A 447C (R2933)
R500B E5005100 (R2934)
R500C 92000039 (R2935)
R500D 812050E2 (R2936)
R500E 060050EF (R2937)
R500F 2646 (R2938)
R500G 894050D5 (R2939)
R500H 1010 (R2940)
R500I 447C (R2941)
R500J E5005100 (R2942)
R500K 92000039 (R2943)
R500L 812050E2 (R2944)
R500M 060050EF (R2945)
R500N 2646 (R2946)
R500O 894050D5 (R2947)
R500P 1010 (R2948)
R500Q 447C (R2949)
R500R E5005100 (R2950)
R500S 92000039 (R2951)
R500T 812050E2 (R2952)
R500U 060050EF (R2953)
R500V 2646 (R2954)
R500W 894050D5 (R2955)
R500X 1010 (R2956)
R500Y 447C (R2957)
R500Z E5005100 (R2958)
R500A 92000039 (R2959)
R500B 812050E2 (R2960)
R500C 060050EF (R2961)
R500D 2646 (R2962)
R500E 894050D5 (R2963)
R500F 1010 (R2964)
R500G 447C (R2965)
R500H E5005100 (R2966)
R500I 92000039 (R2967)
R500J 812050E2 (R2968)
R500K 060050EF (R2969)
R500L 2646 (R2970)
R500M 894050D5 (R2971)
R500N 1010 (R2972)
R500O 447C (R2973)
R500P E5005100 (R2974)
R500Q 92000039 (R2975)
R500R 812050E2 (R2976)
R500S 060050EF (R2977)
R500T 2646 (R2978)
R500U 894050D5 (R2979)
R500V 1010 (R2980)
R500W 447C (R2981)
R500X E5005100 (R2982)
R500Y 92000039 (R2983)
R500Z 812050E2 (R2984)
R500A 060050EF (R2985)
R500B 2646 (R2986)
R500C 894050D5 (R2987)
R500D 1010 (R2988)
R500E 447C (R2989)
R500F E5005100 (R2990)
R500G 92000039 (R2991)
R500H 812050E2 (R2992)
R500I 060050EF (R2993)
R500J 2646 (R2994)
R500K 894050D5 (R2995)
R500L 1010 (R2996)
R500M 447C (R2997)
R500N E5005100 (R2998)
R500O 92000039 (R2999)
R500P 812050E2 (R3000)
R500Q 060050EF (R3001)
R500R 2646 (R3002)
R500S 894050D5 (R3003)
R500T 1010 (R3004)
R500U 447C (R3005)
R500V E5005100 (R3006)
R500W 92000039 (R3007)
R500X 812050E2 (R3008)
R500Y 060050EF (R3009)
R500Z 2646 (R3010)
R500A 894050D5 (R3011)
R500B 1010 (R3012)
R500C 447C (R3013)
R500D E5005100 (R3014)
R500E 92000039 (R3015)
R500F 812050E2 (R3016)
R500G 060050EF (R3017)
R500H 2646 (R3018)
R500I 894050D5 (R3019)
R500J 1010 (R3020)
R500K 447C (R3021)
R500L E5005100 (R3022)
R500M 92000039 (R3023)
R500N 812050E2 (R3024)
R500O 060050EF (R3025)
R500P 2646 (R3026)
R500Q 894050D5 (R3027)
R500R 1010 (R3028)
R500S 447C (R3029)
R500T E5005100 (R3030)
R500U 92000039 (R3031)
R500V 812050E2 (R3032)
R500W 060050EF (R3033)
R500X 2646 (R3034)
R500Y 894050D5 (R3035)
R500Z 1010 (R3036)
R500A 447C (R3037)
R500B E5005100 (R3038)
R500C 92000039 (R3039)
R500D 812050E2 (R3040)
R500E 060050EF (R3041)
R500F 2646 (R3042)
R500G 894050D5 (R3043)
R500H 1010 (R3044)
R500I 447C (R3045)
R500J E5005100 (R3046)
R500K 92000039 (R3047)
R500L 812050E2 (R3048)
R500M 060050EF (R3049)
R500N 2646 (R3050)
R500O 894050D5 (R3051)
R500P 1010 (R3052)
R500Q 447C (R3053)
R500R E5005100 (R3054)
R500S 92000039 (R3055)
R500T 812050E2 (R3056)
R500U 060050EF (R3057)
R500V 2646 (R3058)
R500W 894050D5 (R3059)
R500X 1010 (R3060)
R500Y 447C (R3061)
R500Z E5005100 (R3062)
R500A 92000039 (R3063)
R500B 812050E2 (R3064)
R500C 060050EF (R3065)
R500D 2646 (R3066)
R500E 894050D5 (R3067)
R500F 1010 (R3068)
R500G 447C (R3069)
R500H E5005100 (R3070)
R500I 92000039 (R3071)
R500J 812050E2 (R3072)
R500K 060050EF (R3073)
R500L 2646 (R3074)
R500M 894050D5 (R3075)
R500N 1010 (R3076)
R500O 447C (R3077)
R500P E5005100 (R3078)
R500Q 92000039 (R3079)
R500R 812050E2 (R3080)
R500S 060050EF (R3081)
R500T 2646 (R3082)
R500U 894050D5 (R3083)
R500V 1010 (R3084)
R500W 447C (R3085)
R500X E5005100 (R3086)
R500Y 92000039 (R3087)
R500Z 812050E2 (R3088)
R500A 060050EF (R3089)
R500B 2646 (R3090)
R500C 894050D5 (R3091)
R500D 1010 (R3092)
R500E 447C (R3093)
R500F E5005100 (R3094)
R500G 92000039 (R3095)
R500H 812050E2 (R3096)
R500I 060050EF (R3097)
R500J 2646 (R3098)
R500K 894050D5 (R3099)
R500L 1010 (R3100)
R500M 447C (R3101)
R500N E5005100 (R3102)
R500O 92000039 (R3103)
R500P 812050E2 (R3104)
R500Q 060050EF (R3105)
R500R 2646 (R3106)
R500S 894050D5 (R3107)
R500T 1010 (R3108)
R500U 447C (R3109)
R500V E5005100 (R3110)
R500W 92000039 (R3111)
R500X 812050E2 (R3112)
R500Y 060050EF (R3113)
R500Z 2646 (R3114)
R500A 894050D5 (R3115)
R500B 1010 (R3116)
R500C 447C (R3117)
R500D E5005100 (R3118)
R500E 92000039 (R3119)
R500F 812050E2 (R3120)
R500G 060050EF (R3121)
R500H 2646 (R3122)
R500I 894050D5 (R3123)
R500J 1010 (R3124)
R500K 447C (R3125)
R500L E5005100 (R3126)
R500M 92000039 (R3127)
R500N 812050E2 (R3128)
R500O 060050EF (R3129)
R500P 2646 (R3130)
R500Q 894050D5 (R3131)
R500R 1010 (R3132)
R500S 447C (R3133)
R500T E5005100 (R3134)
R500U 92000039 (R3135)
R500V 812050E2 (R3136)
R500W 060050EF (R3137)
R500X 2646 (R3138)
R500Y 894050D5 (R3139)
R500Z 1010 (R3140)
R500A 447C (R3141)
R500B E5005100 (R3142)
R500C 92000039 (R3143)
R500D 812050E2 (R3144)
R500E 060050EF (R3145)
R500F 2646 (R3146)
R500G 894050D5 (R3147)
R500H 1010 (R3148)
R500I 447C (R3149)
R500J E5005100 (R3150)
R500K 92000039 (R3151)
R500L 812050E2 (R3152)
R500M 060050EF (R3153)
R500N 2646 (R3154)
R500O 894050D5 (R3155)
R500P 1010 (R3156)
R500Q 447C (R3157)
R500R E5005100 (R3158)
R500S 92000039 (R3159)
R500T 812050E2 (R3160)
R500U 060050EF (R3161)
R500V 2646 (R3162)
R500W 894050D5 (R3163)
R500X 1010 (R3164)
R500Y 447C (R3165)
R500Z E5005100 (R3166)
R500A 92000039 (R3167)
R500B 812050E2 (R3168)
R500C 060050EF (R3169)
R500D 2646 (R3170)
R500E 894050D5 (R3171)
R500F 1010 (R3172)
R500G 447C (R3173)
R500H E5005100 (R3174)
R500I 92000039 (R3175)
R500J 812050E2
```

PAGE 66: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(02617) ; - TO OUTPUT 6 PARITY BITS (MSB FIRST)
(02618) ; - INITIALIZE PARITY CHECK WORD (R7) TO 0
(02619) ; - INITIALIZE H-INDEX (R6) TO 1
(02620) ;
(02621) ; PAROUTS MOVIR R3,00005 TO MOVE OUT 6 BITS (+VE INDEX)
(02622) ; PAR51 MOVKR R6,R7,00020 6TH BIT TO R6
(02623) ; LRS R6,5 SHIFT BIT TO LSB
(02624) ; MOVRM R6,0ADSOUBS OUTPUT
(02625) ; INCHM ADSOUBS+1 INCREMENT O-INDEX
(02626) ; LLS R7,1 NEXT BIT TO 6TH POSITION
(02627) ; DJP R3,PAR51
(02628) ;
(02629) ; MOVIR R6,00001 INITIALIZE R6 TO 1
(02630) ; CLR R7 INITIALIZE R7 TO 0
(02631) ;
(02632) ; RETURN
(02633) ;
(02634) ;
(02635) ; --- SUBROUTINE BASES
(02636) ;
(02637) ; - TO GET BASE ADDR AND MEM BUS FOR BUFFER (NOT INDEX REGISTER)
(02638) ; - R6 = BID
(02639) ; - R4,R5 = BASE ADDRESS
(02640) ;
(02641) ; BASES R6,1 INDEX=BID*2
(02642) ; MOVML R4,BCTSBAR6 GET BASE ADDR FROM BCT
(02643) ; ANDIR R4,00006 KEEP ONLY BUS ID
(02644) ; LLS R4,3 MOVE BUS ID TO BITS 4,5
(02645) ; RETURN
(02646) ;
(02647) ;
(02648) ; --- LEAVE SPACE FOR VARIOUS VARIABLES
(02649) ;
(02650) ;
(02651) ; ADOSHAT DATA F'0.0'
(02652) ; ADSOUBS DATA F'0.0'
(02653) ; ADSENCI DATA F'0.0'
(02654) ; ADSBETA DATA F'0.0'
(02655) ; ADSOLDB DATA F'0.0'
(02656) ; ADSOUBT DATA F'0.0'
(02657) ; IDSTB DATA 00000
(02658) ; VSSYN DATA 00001
(02659) ; VSDCL DATA SFFFF
(02660) ; LSRUN DATA 00000

```

PAGE 57: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 35, 1985

```

(02661) :
(02662) :
(02663) : --- UPDATE TOP OF MEMORY
(02664) :
(02665) : TOESNEV=0L
(02666) : END
(02667) : EVEN
(02668) : EJECT

```

PAGE 68: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1968

```
(#2669) ;
(#2670) ;
(#2671) ;
(#2672) ;
(#2673) ;
(#2674) ;
(#2675) ;
(#2676) ;
(#2677) ;
(#2678) ;
(#2679) ;
(#2680) ;
(#2681) ;
(#2682) ;
(#2683) ;
(#2684) ;
(#2685) ;
(#2686) ;
(#2687) ;
(#2688) ;
(#2689) ;
```

```
*****
*
* *** PROGRAM TRBUF ***
*
*****
```

```
PROG:TRSDC          ARVIND S. ARORA
FCB = #112
```

THIS PROGRAM TRANSFERS ELEMENTS FROM A DOUBLE BUFFER (109 LONG EACH) TO A CIRCULAR BUFFER (1024 LONG). THIS IS USEFUL IN TESTING THE PARC ALGORITHM WITHOUT THE USE OF THE IOS2. THIS PROGRAM ESSENTIALLY SIMULATES THE IOS2 IN THE CPU. SINCE THIS PROGRAM ADDS TO THE CPU TIME, WHICH IS ALREADY AT PREMIUM, THE SAMPLING RATE MUST BE LOWERED BY ABOUT 33 PERCENT.

EJECT

PAGE 69: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(B2698) ;
(B2699) ;
(B2700) ;
(B2701) ;
(B2702) ;
(B2703) ;
(B2704) ;
(B2705) ;
(B2706) ;
(B2707) ;
(B2708) ;
(B2709) ;
(B2710) ;
(B2711) ;
(B2712) ;
(B2713) ;
(B2714) ;
(B2715) ;
(B2716) ;
(B2717) ;
(B2718) ;
(B2719) ;
(B2720) ;
(B2721) ;
(B2722) ;
(B2723) ;
(B2724) ;
(B2725) ;
(B2726) ;
(B2727) ;
(B2728) ;
(B2729) ;
(B2730) ;
(B2731) ;
(B2732) ;
(B2733) ;

```

THE FUNCTION CONTROL BLOCK IS AS FOLLOWS:

```

-----
| |
| |
| FCB @ 112 | INIT
|-----
| DBUF2 | DBUF1
|-----
| | CBUF
|-----
| |
|-----
| |
|-----

```

*** NOTE: ENABLE LINES WITH * IN COLUMN 1 TO ASSEMBLE THIS ROUTINE ALONE.

---- SYMBOL DEFINITION TO INTERFACE WITH SNAP-II EXEC. REL 3.05

```

      BCISBA=8582
      FOTSUCB=88C8
      ISVTS=8502
      SYSSFLGS=81FFCE
      TOESOLD=8518F
      TOESPTR=8288

      *OL-TOESPTR
      *      ADDR      TOESNEW(.1)

      ---- DISPATCH TABLE ENTRY

      *OL-FOTSUCB
      *      ADDR      TRSDC

      ---- CSPU MODULE FOR BUFFER TRANSFER, DOUBLE BUFF. --> CIRCULAR BUFF.

      *OL-TOESOLD

      ---- CHECK INIT. SCALAR SELECT

```

PAGE 78: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

05116 707200FF	(02734) :	MOVW	R7,R1,SEF		
05118 2611	(02735) :	MOVLM	\$27,SYSSFLGS		
05119 006E0502	(02736) :	INCR	R1,1		
0511B 00205135	(02737) :	MOVHR	R6,ISVTS(R7)		
0511D 0030512A	(02738) :	JMP	TR\$20,GTZ		
	(02739) :	JMP	TR\$10,LTZ		
	(02740) :				
	(02741) :				
	(02742) :				
0511F 0060	(02743) :	CCR	R6		
	(02744) :				
05120 2611	(02745) :	INCR	R1,1		
05121 0022	(02746) :	MOVHR	R2,R1		
05122 3A21	(02747) :	LLS	R2,1		
05123 C0440502	(02748) :	MOVHRL	R4,BCT\$BA(R2)		
05125 E050514F	(02749) :	MOVHR	R5,ADOUT		
	(02750) :				
05127 CC00515F	(02751) :	MOVZM	IND\$OUT		
05129 2711	(02752) :	DECR	R1,1		
	(02753) :				
	(02754) :				
0512A 0060	(02755) :	NEG	R6		
0512B E06E0502	(02756) :	MOVHR	R6,ISVTS(R7)		
	(02757) :				
0512D 0072	(02758) :	MOVHR	R7,R1		
0512E 3C77	(02759) :	LRS	R7,7		
0512F C04E0502	(02760) :	MOVHRL	R4,BCT\$BA(R7)		
05131 E050514C	(02761) :	MOVHR	R5,ADIN		
05133 0000513F	(02762) :	JMP	TR\$25		
	(02763) :				
	(02764) :				
	(02765) :				
05135 0060	(02766) :	NEG	R6		
05136 E06E0502	(02767) :	MOVHR	R6,ISVTS(R7)		
	(02768) :				
05138 707200FF	(02769) :	MOVW	R7,R1,SEF		
0513A 3A71	(02770) :	LLS	R7,1		
0513B C04E0502	(02771) :	MOVHRL	R4,BCT\$BA(R7)		
0513D E050514C	(02772) :	MOVHR	R5,ADIN		
	(02773) :				
	(02774) :				
0513F F0705160	(02775) :	MOVHR	R7,SIZES		
05141 2771	(02776) :	DECR	R7,1		
	(02777) :				

TIMING INSTRUCTION
 >0,NORMAL OPERATION BUF 1
 <0,NORMAL OPERATION BUF 2
 :=0,INITIALIZATION
 SET SCALAR SELECT TO -1
 GET BASE ADDR FOR CIRC. BUFF
 SAVE ADDR
 SET SID SELECT
 SAVE
 GET BID FOR DOUBLE BUFF 1
 GET BASE ADDR
 SAVE BASE ADDR
 COMPLEMENT SID SELECT
 SAVE
 GET BID FOR DOUBLE BUFF 2
 GET BASE ADDR
 SAVE BASE ADDR
 GET BUFFER SIZE
 SET UP COUNTER

PAGE 72: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 31, 1981

(#2822) 1
(#2823) *
(#2824) *
(#2825)
(#2826)

TOESNEV=0L
END
EVEN
EJECT

08161 0000

PAGE 73: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(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) ;

```

ARVIND S. ARORA

NAME: DCDRS FCB: 111

THIS PROGRAM TAKES FRAMES OF 189 BITS (LSB IN THE 16 BIT WORDS
OUTPUT BY THE IOS2), SYNCHRONIZES ON THE FRAME, AND DECODES THE BITS
INTO A BLOCK OF QUANTIZER LEVELS TO BE USED BY THE RECEIVER OF THE
PARC ALGORITHM.

DEPENDENT ON THE VALUES OF 2 FUNCTION SELECT INDICATORS, IT SELECTS
EITHER TO DECODE, OR SYNCHRONIZE, OR INITIALIZE AND SYNCHRONIZE. IT
ALSO DECIDES WHICH OF THE TWO Q-OUT BUFFERS TO FILL UP.

THE Q-OUT BUFFER HAS THE FOLLOWING FORMAT: PONEY BETA INDICATOR
(-1 MEANS PONEY BETA PRESENT), QUANTIZER LEVELS (1 TO 11, AND A RUN-
LENGTH CODE TO BE DECODED TO 14 LEVEL 1 OUTPUTS), AND AN END-OF-BLOCK
CODE (-VE 12).

EACH TIME THE PROGRAM IS STOPPED, THE MODULE MUST BE RE-LOADED
OR LOCATIONS PSBASE,LSRUN RESET TO ENSURE PROPER INITIALIZATION.

THIS PROGRAM ALSO SIMULATES THE STATE OF THE SHAT BUFFER IN THE
RECEIVER TO ENSURE THE BUFFER DOES NOT UNDER-FLOW OR OVER-FLOW.

CONSTRAINTS:

- 1) THE CONSTRAINTS ON THE SOURCE CODE ARE THE SAME AS THOSE ON THE ENCODER.
- 2) THE OUTPUT OF THE DECODER HAS THE SAME FORM AS THE INPUT OF THE ENCODER.
- 3) THE INPUT OF THIS PROGRAM IS A CIRCULAR BUFFER OF LENGTH 1024.
- 4) ALL THE BUFFERS USED IN THIS PROGRAM MUST RESIDE ON BUS 1.

EJECT

PAGE 74:

MAP MODULES FOR THE P A R C ALGORITHM ---- JAN. 30, 1980

```

(02868) 1
(02869) 1
(02870) 1
(02871) 1
(02872) 1
(02873) 1
(02874) 1
(02875) 1
(02876) 1
(02877) 1
(02878) 1
(02879) 1
(02880) 1
(02881) 1
(02882) 1
(02883) 1
(02884) 1
(02885) 1
(02886) 1
(02887) 1
(02888) 1
(02889) 1
(02890) 1
(02891) 1
(02892) 1
(02893) 1
(02894) 1
(02895) 1
(02896) 1
(02897) 1
(02898) 1
(02899) 1
(02900) 1
(02901) 1
(02902) 1
(02903) 1
(02904) 1
(02905) 1
(02906) 1
(02907) 1
(02908) 1
(02909) 1
(02910) 1
(02911) 1

```

THE FUNCTION CONTROL BLOCK HAS THE FOLLOWING FORMAT:

```

-----
1
-----
1 FCB # 111 1 ISELECT 1
-----
1 0082 1 0081 1
-----
1 ITB2 1 ITB1 1
-----
1 DECT (TABLE) 1 DECB (TABLE) 1
-----
1 ICAP 1 ICB 1
-----

```

*** NOTE: ENABLE ALL LINES WITH * IN COLUMN TO ASSEMBLE THIS PROGRAM ALONE.

--- SYMBOL DEFINITION TO INTERFACE WITH SNAP-II EXEC. REL 3.05

```

8CTS8A-S582      BASE OF SCALAR TABLE
FOTSUFCE-S8C6    FCB # 111
ISVTS-S582       BASE OF INTEGER SCALAR TABLE
NOSMAX-S9        (NO OF WORDS IN SYNC CORR. COMPUTATION)
SYSSFLG-S1FFCE   SYSTEM FLAG LOCATION
TOESOLD-S5288    OLD TOP OF MEMORY
TOESPTR-S288     TOP OF MEMORY POINTER
SVTS-S382        BASE OF SCALAR TABLE

--- TOP OF MEMORY POINTER
*OL=TOESPTR
ADDR TOESNEW(.1)

```

```

PAGE 78: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30. 1988

      (B2912) : --- DISPATCH TABLE ENTRIES
      (B2913) :
      (B2914) : OL=FDTSUFCB
      (B2915) :
      (B2916) : ADDR DCDRS
      (B2917) :
      (B2918) :
      (B2919) : --- CSPU MODULE FOR THE DECODER
      (B2920) :
      (B2921) : OL=TOESOLD
      (B2922) :
      (B2923) : FIRST DECIDE WHICH OPERATIONS ARE TO BE DONE
      (B2924) :
      (B2925) : DCDRS MOVNR R1,BASFCB STACK FCB POINTER
      (B2926) : NOP 2 SAVE SPACE FOR TIMING INSTRUCTION

      (B2927) :
      (B2928) : --- USE FUNCTION SELECT INDICATOR TO DECIDE OPERATION
      (B2929) :
      (B2930) : MOVNR R2,R1,SBFF GET SID OF ISL
      (B2931) : MOVNR R2,SIDSL SAVE SID
      (B2932) : MOVNR R3,ISVTS(R2) GET ISL1
      (B2933) : JMP DECS,GTZ ISL1 > B, DECODER
      (B2934) : JMP SYNC,LTZ ISL1 < B, SYNCHRONIZE
      (B2935) :
      (B2936) : ISL1 = B, INITIALIZE & SYNCHRONIZE
      (B2937) :
      (B2938) :
      (B2939) :
      (B2940) :
      (B2941) : 1) SET ISL1 TO -1
      (B2942) :
      (B2943) : EVEN
      (B2944) : CCR R3 SET R3=-1
      (B2945) : MOVNR R3,ISVTS(R2) SET ISL1 TO -1
      (B2946) :
      (B2947) : 2) SET MAX CORRELATION INDEX WORDS (1B) TO -1
      (B2948) :
      (B2949) : CCR R5 R5=-1
      (B2950) : MOVNR R4,MOSMAX INDEX=9 (FOR 1B WORDS)
      (B2951) : MOVNR R5,PSMAX(R4) MAX CORR WORD = -1
      (B2952) : DJP R4,INS2B
      (B2953) :
      (B2954) : 3) GET BASE ADDRESS OF Q-BUFFER1 & SAVE IN BASQOB

```

```

PAGE 76: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

( #2955 ) ;
( #2956 ) ;
( #2957 ) ;
( #2958 ) ;
( #2959 ) ;
( #2960 ) ;
( #2961 ) ;
( #2962 ) ;
( #2963 ) ;
( #2964 ) ;
( #2965 ) ;
( #2966 ) ;
( #2967 ) ;
( #2968 ) ;
( #2969 ) ;
( #2970 ) ;
( #2971 ) ;
( #2972 ) ;
( #2973 ) ;
( #2974 ) ;
( #2975 ) ;
( #2976 ) ;
( #2977 ) ;
( #2978 ) ;
( #2979 ) ;
( #2980 ) ;
( #2981 ) ;
( #2982 ) ;
( #2983 ) ;
( #2984 ) ;
( #2985 ) ;
( #2986 ) ;
( #2987 ) ;
( #2988 ) ;
( #2989 ) ;
( #2990 ) ;
( #2991 ) ;
( #2992 ) ;
( #2993 ) ;
( #2994 ) ;
( #2995 ) ;
( #2996 ) ;
( #2997 ) ;
( #2998 ) ;

( TO BE USED FOR STORING SYNC CORRELATION VALUES )
( INDEXED BY R2 )

INCR R1,1
MOVW R2,R1,SBFF BID OF OOB1
LLS R2,1 BID*2
MOVRL R3,BCTSSA(R2) GET BASE ADDR WORD FOR OOB1
MOVIR R3,S14 BUS 1, IND REG R2
MOVRML R3,BASOQB SAVE BASE ADDR

--- UPDATE PROGRAM LOCATIONS REFERRING TO OOB
MOVW R4,OOB10
MOVW R4,OOB11
MOVW R4,OOB12
MOVW R4,OOB13
MOVW R4,OOB14
MOVW R4,OOB15

4) INITIALIZE OOB BUFFER TO 0
MOVIR R2,S18F R2=399
CLR R5
OQB10=0L+1
INS4# MOVW R5,BAS4(R2) INIT OOB TO 0
DJP R2,INS4#

5) GET BASE ADDR OF INPUT CIRCULAR BUFFER, IND REG R3, SIZE=1024
INCR R1,3
MOVW R2,R1,SBFF BID OF ICB
LLS R2,1 BID*2
MOVRL R3,BCTSSA(R2) GET BASE ADDR WORDS FOR ICB
MOVIR R3,S16 BUS 1, IND REG R3
MOVRML R3,BAS1CB SAVE BASE ADDR

--- MODIFY PROGRAM LOCATIONS REFERRING TO ICB
MOVW R4,ICB10
MOVW R4,ICB11
MOVW R4,ICB12
MOVW R4,ICB13
MOVW R4,ICB14

```


PAGE 70: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

```

05108 9A100001 (03043) ANDIR R1,00001 MASK LSB
05109 9A100001 (03044) READY TO MOVE IN BIT 1
0510A 9A700001 (03045) SCS20 MOVIR R7,01
0510B 9A700001 (03046) ICB10-OL+1
0510C 9A700001 (03047) ANDMR R7,BAS1CB(R3) MOVE LSB OF INPUT WORD
0510D 4472 (03048) KORRR R7,R1 RESULT OF CORR. OF SYNC WITH INFO BIT
0510E 002001EE (03049) JMP SCS22,GTZ ;SAME = 0, DIFFERENT = 1
0510F 002001EE (03050) JUMP FOR DIFFERENT
0510G 002001EE (03051) ;
0510H 002001EE (03052) ; --- FOR MATCH WITH SYNC BIT
0510I 002001EE (03053) ;
0510J 002001EE (03054) OOB11-OL+1
0510K 002001EE (03055) MOVMR R6,BAS00B(R2) R6=SYNC COR VAL FOR CURRENT BIT
0510L 002001EE (03056) INCR R6,1 ADD 1 TO IT
0510M 002001EE (03057) CMPRR R6,R5 CMPR WITH MAX SO FAR (R6-R5)
0510N 002001EE (03058) JMP SCS21,LTZ THIS VALUE < MAX SO FAR
0510O 002001EE (03059) MOVRR R5,R6 IF MAX, UPDATE MAX VALUE
0510P 002001EE (03060) MOVRR R4,R2 UPDATE MAX INDEX
0510Q 002001EE (03061) OOB12-OL+1
0510R 002001EE (03062) SCS21 MOVRR R6,BAS00B(R2) SAVE NEW CORR VALUE
0510S 002001EE (03063) DECR R2,1 DECR HIST INDEX
0510T 002001EE (03064) JMP SCS24
0510U 002001EE (03065) ;
0510V 002001EE (03066) ; --- FOR NO MATCH WITH SYNC BIT
0510W 002001EE (03067) ;
0510X 002001EE (03068) SCS22 DECR R2,1 DECR HIST INDEX
0510Y 002001EE (03069) OOB13-OL+1
0510Z 002001EE (03070) MOVMR R6,BAS00B(R2) SYNC CORR VAL TO R6
0510A 002001EE (03071) INCR R6,1
0510B 002001EE (03072) CMPRR R6,R5 (R6-R5)
0510C 002001EE (03073) JMP SCS23,LTZ THIS VAL NOT MAX
0510D 002001EE (03074) MOVRR R5,R6 IF MAX, UPDATE MAX VAL
0510E 002001EE (03075) MOVRR R4,R2 UPDATE MAX INDEX
0510F 002001EE (03076) OOB14-OL+1
0510G 002001EE (03077) SCS23 MOVRR R6,BAS00B(R2) SAVE NEW CORR VAL
0510H 002001EE (03078) ;
0510I 002001EE (03079) ; --- MERGE HERE BOTH CASES, MATCH OR NO-MATCH
0510J 002001EE (03080) ;
0510K 002001EE (03081) SCS24 DECR R3,1 INP IND
0510L 002001EE (03082) ANDIR R3,003FF MOD 1/24
0510M 002001EE (03083) DJP R2,SCS20 DEC HIST IND & LOOP
0510N 002001EE (03084) ;
0510O 002001EE (03085) ; --- IF NOT SYNC'D IN 10 SECONDS, REINITIALIZE
0510P 002001EE (03086) ;

```


PAGE 88: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(03131) ;
05226 3C41      LRS      R4.1      R4/2+BASE IS BIT-1 POINTER
05226 4C46      ADDR      R4.R3      MOD 1024
05227 9A403FF  ANDIR      R4.03FF
05229 E0405489 MOVNR      R4.P8BIT1

(03136) ;
(03137) ; ---- FUNCTION LIST SELECTOR UPDATE
(03138) ;
05228 F0205486 MOVNR      R2.SIDSL
05229 90700001 MOVIR      R7.S1
05229 90700001 MOVNR      R7.ISVTS(R2)
05229 90700001 INCR      R2.1
05231 2621      MOVNR      R6.ISVTS(R2)
05232 F0640502 NEG        R6
05234 0060      MOVNR      R6.ISVTS(R2)
05235 E0640502

(03146) ;
(03147) ; 5) INITIALIZE BETA, T, AND Q-OUT FOR NEXT RECEIVER OPERATION
(03148) ;
(03149) ; ---- INITIALIZE T-B I.E.2B, AND BETA=48 I.E. 0.0
(03150) ;
05237 F0105484 MOVNR      R1.BASFCB
05239 2612      INCR      R1.2
0523A 6022      MOVNR      R2.R1
0523B 0260      TEST      R6
0523C 1030      SKP       LTZ
0523D 3C20      LRS       R2.8
0523E 9A2000FF ANDIR      R2.SFF

(03158) ;
05240 0D10      CLR        R1
05241 3A21      LLS        R2.1
05242 E0140382 MOVNR      R1.SVTS(R2)

(03162) ;
05244 90100030 MOVIR      R1.S30
05246 2622      INCR      R2.2
05247 E0140382 MOVNR      R1.SVTS(R2)

(03166) ;
(03167) ; ---- INITIALIZE NEXT Q-OUT BUFF [1 (NO PH BETA), 126 B'S, -12]
(03168) ;
05249 F0105484 MOVNR      R1.BASFCB
05248 2611      INCR      R1.1
0524C 6022      MOVNR      R2.R1
0524D 0260      TEST      R6
0524E 1030      SKP       LTZ
0524F 3C20      LRS       R2.8

```

ISL1 = 1, FOR DECODER
SET ISL1 TO +1
SID OF ISL2
VAL OF ISL2
ISL2 = -ISL2

GET FCB POINTER
POINT TO ITB IN FCB
GET VALS OF ITB1 AND ITB2
CHECK FUNCTION LIST FOR NEXT OP.
ISL2 = -1, FUNC. LIST 1
ISL2 = 1, FUNC. LIST 2
KEEP APPROP. ITB

OUT T=0 I.E. 20
OUT BETA=48 I.E. 0.0

BID WORD
ISL2 = -1 FUNC LIST 1
ISL2 = 1 FUNC LIST 2

```

PAGE 91:      MAP MODULES FOR THE P A R C ALGORITHM      --- JAN. 35, 1985

#5258 9A2#FFF (#3175)      ANDIR  R2,FFF
#5252 3A21  (#3176)      LLS      R2,1
#5253 C#44#582  (#3177)      MOVHRL R4,BCT$BA(R2)      ADDR OF NEXT Q-OUT BUFF
#5255 9#4#514  (#3179)      MOVIR  R4,$14      BUS 1, IND REG R2
#5257 844#54AE (#3181)      MOVHRL R4,BASOQB      SAVE
#5259 E#5#5262 (#3184)      MOVHRL R5,OQB#1
#5258 E#5#526A (#3185)      MOVHRL R5,OQB#2
#525D 9#2#57E  (#3188)      MOVIR  R2,$7E      Q-LEVEL 1*2
#525F 9#5#5262 (#3189)      MOVIR  R6,$2
#5261 E#6454AE (#3191)      MOVHRL R5,BASOQB(R2)      126 1'S TO OQB
#5263 8C2#5261 (#3192)      SC$52      R2,SC$52
#5265 9#2#57F  (#3194)      MOVIR  R2,$7F
#5267 9#6#FFF4 (#3195)      MOVIR  R6,$FFF4      END CODE= -12
#5269 E#6454AE (#3197)      MOVHRL R6,BASOQB(R2)
#5268 F#6#5406 (#32#1)      MOVHRL R6,SZ$SHAT
#526D 266F  (#32#2)      INCR    R6,16
#526E E#6#5487 (#32#3)      MOVHRL R6,P$SHAT
#5270 F#1#5484 (#32#5)      MOVHRL R1,BASFCB
#5272 2613  (#32#6)      INCR    R1,3
#5273 7#22#FFF (#32#9)      MOVHRL R2,R1,$$FFF
#5276 3A21  (#321#)      LLS      R2,1
#5278 C#44#582 (#3211)      MOVHRL R4,BCT$BA(R2)      R4,R5=BASE ADDR WORDS
#5279 9#4#51E  (#3212)      MOVIR  R4,$$1E      BUS1, IND REG R7
#527A 844#5482 (#3213)      MOVHRL R4,BASDEC$
#527C E#5#526E (#3217)      MOVHRL R5,DEC$1$
#527D 844#5482 (#3218)      MOVHRL R5,BASDEC$

```


ADDRESS	INSTR	OPERANDS	REMARKS
0631F 2611	INCR	R1,1	BID FOR QOB8
06320 70220FFF	MOVW	R2,R1,\$0FFF	
06322 3A21	LLS	R2,1	
06323 C0440502	MOVWRL	R4,BCTSBA(R2)	R4,R5=BASE ADDR
06325 9040012	MOVIR	R4,\$0012	BUS1, IND REG R1
06327 044054AE	MOVWRL	R4,BASOQB	SAVE
06329 2611	INCR	R1,1	
0632A 70220FFF	MOVW	R2,R1,\$0FFF	
0632C E02054B5	MOVWRL	R2,\$105TB	SAVE SID FOR T,BETA
0632E 0000033E	JMP	DC\$21	
06330 2611	INCR	R1,1	ISL=-1 BUFF#2
06331 0022	MOVW	R2,R1	
06332 3C27	LRS	R2,7	
06333 C0440502	MOVWRL	R4,BCTSBA(R2)	R4,R5=BASE ADDR
06335 9040012	MOVIR	R4,\$0012	BUS 1, IND REG R1
06337 044054AE	MOVWRL	R4,BASOQB	
06339 2611	INCR	R1,1	
0633A 0022	MOVW	R2,R1	
0633B 3C20	LRS	R2,8	
0633C E02054B5	MOVWRL	R2,\$105TB	SAVE SID FOR T,BETA
0633E 2000	NOP		R5 CONTAINS QOB ADDR
0633F E060547C	MOVWRL	R5,QOB26	
06341 E0605407	MOVWRL	R5,QOB27	
06342 1			
06342B 1			
063426 1			
063427 1			
06343 90200002	MOVIR	R2,\$0002	COUNTER FOR 3 BLOCKS
06340 F0605409	MOVWRL	R5,\$0011	SAVE BIT1 POINTER IN R5
06347 9C50003F	ADDIR	R5,\$003F	POINT TO NEXT FRAME I.E.+63
06349 9A5003FF	ANDIR	R5,\$03FF	
06340 90300039	MOVIR	R3,\$0039	POINT TO PARITY BIT
0634D 4C3A	ADDIR	R3,R5	
0634E 9A3003FF	ANDIR	R3,\$03FF	

PAGE 87: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

#5350 8D70 (M3438) CLR R7
#5351 9B400005 (M3439) MOVIR R4,000005
#5353 3A71 (M3440) DCS31 R7,1
#5355 00000000 (M3441) ICB22=OL+1
#5354 FB684AC (M3442) MOVHR R6,BASICB(R3)
#5356 567C0001 (M3443) IORRR R7,R6,000001
#5358 2631 (M3444) ;
#5359 9A30003FF (M3445) INCR R3,1
#5358 8C405353 (M3446) ANDIR R3,003FF
#5358 8C405353 (M3447) ;
#5358 8C405353 (M3448) DJP R4,DCS31
#5358 8C405353 (M3449) ;
#5358 8C405353 (M3450) ; --- PARITY CHECK ON 57 BITS
#5358 8C405353 (M3451) ;
#5358 8C405353 (M3452) MOVRR R3,R5
#5358 8C405353 (M3453) MOVIR R4,000038
#5358 8C405353 (M3454) CLR R6
#5358 8C405353 (M3455) ;
#5358 8C405353 (M3456) DCS32 INCR R6,1
#5358 8C405353 (M3457) ICB21=OL+1
#5358 8C405353 (M3458) MOVHR R1,BASICB(R3)
#5358 8C405353 (M3459) SRBC R1
#5358 8C405353 (M3460) XORRR R7,R6
#5358 8C405353 (M3461) ;
#5358 8C405353 (M3462) INCR R3,1
#5358 8C405353 (M3463) ANDIR R3,003FF
#5358 8C405353 (M3464) ;
#5358 8C405353 (M3465) DJP R4,DCS32
#5358 8C405353 (M3466) ;
#5358 8C405353 (M3467) ; --- CORRECT ANY ERROR
#5358 8C405353 (M3468) ;
#5358 8C405353 (M3469) TEST R7
#5358 8C405353 (M3470) JMP DCS33,EQZ
#5358 8C405353 (M3471) DECR R7,1
#5358 8C405353 (M3472) MOVRR R3,R5
#5358 8C405353 (M3473) ADDR R3,R7
#5358 8C405353 (M3474) ANDIR R3,003FF
#5358 8C405353 (M3475) MOVIR R1,000001
#5358 8C405353 (M3476) ICB22=OL+1
#5358 8C405353 (M3477) XORRR R1,BASICB(R3)
#5358 8C405353 (M3478) ;
#5358 8C405353 (M3479) DCS33 DJP R2,DCS30
#5358 8C405353 (M3480) ;
#5358 8C405353 (M3481) ; 4) UPDATE FOR SYNC MONITOR

```

COUNTER FOR 6 BITS

NEXT PARITY BIT
LOAD TO LSB OF PARITY WORD

INP INDEX

SET UP INP INDEX
COUNTER 56 - 0

INFO WORD
MASK BIT 1
PARITY UPDATE FOR 1

INP IND

SET UP ERROR IND

LOOP FOR 3 H-FRAMES

```

PAGE 88:      MAP MODULES FOR THE P A R C ALGORITHM      --- JAN. 30, 1980

      (M3482) ;
      (M3483) ;
      (M3484) ;
      (M3485) ;
      (M3486) ;
      (M3487) ;
      (M3488) ;
      (M3489) ;
      (M3490) ;
      (M3491) ;
      (M3492) ;
      (M3493) ;
      (M3494) ;
      (M3495) ;
      (M3496) ;
      (M3497) ;
      (M3498) ;
      (M3499) ;
      (M3500) ;
      (M3501) ;
      (M3502) ;
      (M3503) ;
      (M3504) ;
      (M3505) ;
      (M3506) ;
      (M3507) ;
      (M3508) ;
      (M3509) ;
      (M3510) ;
      (M3511) ;
      (M3512) ;
      (M3513) ;
      (M3514) ;
      (M3515) ;
      (M3516) ;
      (M3517) ;
      (M3518) ;
      (M3519) ;
      (M3520) ;
      (M3521) ;
      (M3522) ;
      (M3523) ;
      (M3524) ;
      (M3525) ;

      (M3379 4B3A)
      (M337A F87B54D3)
      (M337C 9B1B8B81)
      (M337E F41B54CF)
      (M338B 9A1B8B81)
      (M338C 8B8B5383)
      (M3382 F41B64AC)
      (M3384 2A82)
      (M3386 3B71)
      (M3388 1B8B)
      (M3387 3A72)
      (M3388 E87B54D3)
      (M338A 2B3E)
      (M338B 8B8B53A2)
      (M338D F87B5488)
      (M338F 9C7B8B8D)
      (M3391 9A7B83FF)
      (M3393 E87B5488)
      (M3395 F87B5486)
      (M3397 B06B)
      (M3398 E86E85B2)
      (M339A 2671)
      (M339B F86E85B2)
      (M339D B86B)
      (M339E E86E85B2)
      (M33A8 8B8B54A6)

      V(I+1)=RIGHT SHIFT 1 [V(I)], IF CORRELATED
      V(I+1)=LEFT SHIFT 2 [V(I)], NOT CORRELATED
      BIT 11 CLEAR ==> SYNC LOST
      MOVRR R3,R5 POINT TO NEXT SYNC BIT
      MOVRR R7,VSPAR GET SYNC VARIABLE
      MOVIR R1,8B8B81 CHECK CORR. IN THIS FRAME
      XORRR R1,VSSYNC
      ANDIR R1,8B8B81
      ICB23=OL+1
      XORRR R1,BASICB(R3) 1 - MATCH , 0 - NO MATCH
      SRBCL R,R1
      ARS R7,1 CORRELATED
      SKP R,1
      LLS R7,2 UNCORRELATED
      MOVRR R7,VSPAR SAVE NEW VALUE
      SRBCL 11,R7 IF BIT 11 CLEAR, SYNC LOST
      JMP DC55B SYNC RETAINED, GO TO DECODE
      ---- IF SYNC LOST, SET UP FOR SYNC ACQUISITION NEXT TIME
      MOVRR R7,PSBASE
      ADDIR R7,8BD UPDATE BASE OF FRAME POINTER
      ANDIR R7,8B3FF
      MOVRR R7,PSBASE
      ---- UPDATE FUNCTION SELECT FLAGS AND RETURN
      ISL1=B
      ISL2=-ISL2
      MOVRR R7,SIDSISL
      CLR R6
      MOVRR R6,ISVTS(R7)
      INCR R7,1
      MOVRR R6,ISVTS(R7)
      NEG R6
      MOVRR R6,ISVTS(R7)
      ---- GO FOR XMTR UPDATE
      JMP DC59B JUMP FOR XMTR UPD

```

PAGE 89: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 31, 1981

```

(03526) ;
(03527) ;
(03528) ; 5) MOV T,BETA,PH BETA TO THEIR LOCATIONS
(03529) ;
(03530) ; ---- MOVE T TO INTEGER SCALAR TABLE
(03531) ;
(03532) ; DC 5H MOVHR R7,SIDSTB
(03533) ; LLS R7,1
(03534) ; MOVHR R6,VST
(03535) ; MOVHR R6,SVTS(R7)
(03536) ;
(03537) ; ---- MOVE BETA TO ITS LOCATION IN INTEGER SCALAR TABLE
(03538) ;
(03539) ; INCR R7,2
(03540) ; MOVHR R6,VSBETA
(03541) ; MOVHR R6,SVTS(R7)
(03542) ;
(03543) ; ---- NOW MOVE PHONEY BETA TO LOCATION IN OUTPUT BUFFER
(03544) ;
(03545) ; MOVHR R1,BASOQB+1 INITIALIZE OUTPUT POINTER
(03546) ; MOVHR R6,VSPHB GET PHONEY BETA VALUE
(03547) ;
(03548) ; ---- UPDATE SHAT BUFFER POINTER IF PHONEY BETA
(03549) ;
(03550) ; JMP DC51,CTZ NO PH. BETA. JUMP AND PROCEED
(03551) ; MOVHR R7,VSGAP GET GAP SIZE
(03552) ; ADDR R7,PSHAT UPDATE SHAT POINTER
(03553) ;
(03554) ; DC51 PUSHX1 R1,R6 OUTPUT PH. BETA VALUE
(03555) ;
(03556) ; 6) DECODE NEXT T,BETA, PH BETA (R3,R5 STILL HAV LOC FOR NEXT FRAME)
(03557) ;
(03558) ; ---- FIRST T
(03559) ;
(03560) ; INCR R3,1
(03561) ; ANDIR R3,SB3FF POINT TO 1ST BIT (MSB) OF T
(03562) ;
(03563) ; MOVHR R4,RS
(03564) ; CLR R7
(03565) ;
(03566) ; DC56H LLS R7,1
(03567) ; IS24=0L+1
(03568) ; MOVHR R6,BASICB(R3)
(03569) ; IORR R7,R6,SB5B1

```

PAGE 98: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

```

(03578) ;
033C4 2631      INCR      R3,1      INP IND
033C5 9A3003FF  ANDIR      R3,03FF
(03579) ;
033C7 0C40030F  DJP        R4,DC060
(0357A) ;
033C9 E0700400  MOVHR     R7,VST      SAVENEW T
(0357B) ;
(0357C) ;
(0357D) ;
(0357E) ;
(0357F) ;
(03580) ;
033C0 90700001  MOVIR     R7,S1      TO INITIALIZE PHONEV BETA
033C1 E0700402  MOVHR     R7,VSPHB   INITIALIZE PHONEV BETA
033C2 90200001  MOVIR     R2,S1
(03583) ;
033D1 90400006  MOVIR     R4,S6
033D3 0D70      CLR        R7
(03586) ;
033D4 3A71      ILS        R7,1
(03587) ;
033D5 00000306  ICB20-OL+1
033D6 F06004AC  MOVHR     R6,BASICB(R3)
033D7 067C0001  IONKR
(03591) ;
033D9 2631      INCR      R3,1
033DA 9A3003FF  ANDIR      R3,03FF
(03594) ;
033DC 0C4003D4  DJP        R4,DC062
(03596) ;
033DE 0000030F  DECB20-OL+1
033DF F0600402  MOVHR     R6,0A0DECB(R7)      DECODE VAL OF BETA
(03599) ;
033E0 01300300  JMP        DC063,GEZ
(035A1) ;
033E2 E0600402  MOVHR     R6,VSPHB
(035A2) ;
033E4 0C2003D1  DJP        R2,DC061
(035A3) ;
033E6 906000030  MOVIR     R6,0A030
(035A7) ;
033E8 E06004D1  MOVHR     R6,V0BETA
(035A9) ;
(035AB) ;
(035AC) ;
(035AD) ;
(035AE) ;
(035AF) ;
(035B0) ;
(035B1) ;
(035B2) ;
(035B3) ;
(035B4) ;
(035B5) ;
(035B6) ;
(035B7) ;
(035B8) ;
(035B9) ;
(035BA) ;
(035BB) ;
(035BC) ;
(035BD) ;
(035BE) ;
(035BF) ;
(035C0) ;
(035C1) ;
(035C2) ;
(035C3) ;
(035C4) ;
(035C5) ;
(035C6) ;
(035C7) ;
(035C8) ;
(035C9) ;
(035CA) ;
(035CB) ;
(035CC) ;
(035CD) ;
(035CE) ;
(035CF) ;
(035D0) ;
(035D1) ;
(035D2) ;
(035D3) ;
(035D4) ;
(035D5) ;
(035D6) ;
(035D7) ;
(035D8) ;
(035D9) ;
(035DA) ;
(035DB) ;
(035DC) ;
(035DD) ;
(035DE) ;
(035DF) ;
(035E0) ;
(035E1) ;
(035E2) ;
(035E3) ;
(035E4) ;
(035E5) ;
(035E6) ;
(035E7) ;
(035E8) ;
(035E9) ;
(035EA) ;
(035EB) ;
(035EC) ;
(035ED) ;
(035EE) ;
(035EF) ;
(035F0) ;
(035F1) ;
(035F2) ;
(035F3) ;
(035F4) ;
(035F5) ;
(035F6) ;
(035F7) ;
(035F8) ;
(035F9) ;
(035FA) ;
(035FB) ;
(035FC) ;
(035FD) ;
(035FE) ;
(035FF) ;

```

THIS POSITION MEANS ERROR IN BETA
; FORCE BETA TO 0.0

--- HANG ONTO 1ST INFO BIT IN NEXT FRAME (IN R5)
MOVHR R5,R3

7) NOW TO DECODE QUANTIZER OUTPUTS

PAGE 92: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

0041A 0020000 (R3650) MOVHR R6,B(R1)
0041C 0070 (R3659) ;
0041D FC100110 (R3660) CLR R7
0041F 2012 (R3661) ADDHR R1,LSRUN
(R3662) INCR R1,2
00420 004003FE (R3663) ;
(R3664) DJP R4,DCS71
00422 00000040 (R3665) ;
(R3666) JMP DCS75
(R3667) ;
(R3668) ;---- CODE WORD
(R3669) ;
00424 341C (R3670) DCS73 PUSHX1 R1,R6
00426 0070 (R3671) CLR R7
(R3672) ;
00428 004003FE (R3673) DCS74 DJP R4,DCS71
(R3674) ;
(R3675) ;---- AT THE END OF EACH H-FRAME
(R3676) ;
00420 2636 (R3677) DCS75 INCR R3,6
00429 9A3003FF (R3678) ANDIR R3,S03FF
(R3679) ;
0042B 002003FC (R3680) DCS76 DJP R2,DCS70
(R3681) ;
(R3682) ;---- END OF 3 H-FRAMES I.E. ONE INFO FRAME OF 189 BITS
(R3683) ;
0042D 403A (R3684) MOVHR R3,R5
(R3685) ;
(R3686) ;
0042E 0260 (R3687) TEST R6
0042F 00200464 (R3688) JMP DCS84,GTZ IF FRAME COVERED EXACTLY QUIT
(R3689) ;
00431 90400000A (R3690) MOVHR R4,SA LAST LEVEL, 11 MORE BITS
(R3691) ;
00433 3A71 (R3692) DCS77 LLS R7,1
00435 000000430 (R3693) IC827=0L+1
00436 F00004AC (R3694) MOVHR R6,BASICB(R3)
00438 00700001 (R3695) IORHR R7,R6,S0001
(R3696) ;
00430 2631 (R3697) INCR R3,1
00430 9A3003FF (R3698) ANDIR R3,S03FF
(R3699) ;
00430 0000043C (R3700) DECT21=0L+1
00430 F0000400 (R3701) MOVHR R6,BASDECT(R7)

```

FOR NEXT CODE WORD
SET UP OUT POINTER AFTER
RPT INSTR HAS DECR IT BY LRUN+2

LOOP FOR NEXT INP

END OF H-FRAME PROCESSING

LOOP FOR 3 H-FRAMES

AT THE END OF 189 FRAME
UPDATE TO NEXT INFO BIT

PAGE 93: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 38, 1988

```

08430 0010844E (03702)      JMP      DC879, EQZ      NOT YET CODE
0843F 00208462 (03703)      JMP      DC808, GTZ      CODE WORD
                                !LTZ, NULL OR RUN LENGTH
                                !---- NULL CODE
08441 F2608404 (03706)      CMPMR    R6, V8THR      THRESHOLD--16
08443 00208464 (03707)      JMP      DC804, GTZ      >THR, NULL CODE
                                !<THR, RUN LENGTH
                                !---- RUN LENGTH CODE, OUTPUT 14 1'S
08445 F0708110 (03711)      MOVMR    R7, LSRUN      COUNTER FOR 14 1'S
08447 00608002 (03712)      MOVIR    R6, S2      LEVEL 1*2
08449 341C      DC870      PUSHX1   R1, R6      OUTPUT TO BUFFER
0844A 0C708449 (03717)      DJP
                                !
0844C 00008464 (03719)      JMP      DC884
0844E 0C408433 (03721)      DJP      R4, DC877
08450 00008464 (03722)      JMP      DC884
                                !
                                !---- LAST CODE WORD
08452 341C      DC880      PUSHX1   R1, R6
08453 00008464 (03729)      JMP      DC884
08455 403A      DC881      MOVRR    R3, R5
                                !
                                !---- AFTER A NULL CODE, PUT IN 5 EXTRA LEVEL 1'S
                                ! ONLY IF 0 SAMPLES OUT = 126
08456 0060807F (03736)      MOVIR    R6, S7F      126*1
08458 4E62      SUBRR    R6, R1      126-(ADDR+N)
08459 FC6084AF (03737)      ADDMR    R6, BASOQB+1      126-(BASE ADDR+N)+BASE ADDR
0845B 01108464 (03738)      JMP      DC884, NEZ      N .NE. 126, NO EXTRA SAMPLES
                                !
0845D 00708004 (03741)      MOVIR    R7, S4      COUNTER FOR 5 1'S
0845F 00608002 (03742)      MOVIR    R6, S2      LEVEL 1*2
08461 341C      DC883      PUSHX1   R1, R6      OUTPUT LEVEL 1'S
08462 0C708461 (03745)      DJP

```

```

PAGE 94: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

(03746) ;
(03747) ;
(03748) ; --- CHECK RECEIVER BUFFER FULL
(03749) ;
(03750) DCS84
(03751) MOVNR R7, SZSSHAT
(03752) MOVNR R6, PSSSHAT
(03753) SUBIR R6, 57F
(03754) SUBNR R1, BASOQB+1
(03755) ADDR R6, R1
(03756) CHPR R6, R7
(03757) JMP DCS86, LEZ
(03758) SUBNR R6, R7
(03759) SUBNR R1, R6
(03760) MOVNR R6, R7
(03761) JMP DCS89
(03762) ;
(03763) ; --- CHECK RECEIVER BUFFER EMPTY CONDITION
(03764) DCS86
(03765) CHPR R6, 58
(03766) JMP DCS89, GEZ
(03767) ;
(03768) MOVNR R7, 52
(03769) QOB26=OL+1
(03770) DCS87
(03771) MOVNR R7, BASOQB(R1)
(03772) INCR R1, 1
(03773) IJN R6, DCS87
(03774) ;
(03775) CLR R6
(03776) DCS89
(03777) MOVNR R6, PSSSHAT
(03778) ; --- NULL CODE
(03779) ;
(03780) MOVNR R6, 5C
(03781) NEG R6
(03782) QOB27=OL+1
(03783) MOVNR R6, BASOQB(R1)
(03784) ;
(03785) ; 5, UPDATE FOR NEXT OPERATION OF DECODER
(03786) ;
(03787) ; --- NEXT INFO BIT POINTER
(03788) MOVNR R3, PSINFO
(03789) ;

GET REC BUFFER SIZE
REC BUFFER POINTER
SAMPLES REMOVED FROM BUFF. 126+1
COMPUTE # SAMPLES OUTPUT
SAMPLES ADDED TO SHAT BUFF.
(BUFF. POINTER - BUFF. SIZE)
NOT FULL. JMP AND PROCEED
NO. OF EXTRA SAMPLES
DELETE THE EXTRA SAMPLES
SHAT POINTER = 1024
GO FOR END CODE

(BUFF POINTER - 0)
NOT EMPTY, PROCEED
IF EMPTY, FILL WITH LEVEL 1'S
LEVEL 1*2

OUTPUT LEVEL 1
OUT IND

SHAT POINTER = 0
SAVE NEW SHAT POINTER

NULL CODE = -12

```

PAGE 95: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

(03798) ;--- NEXT BIT1 POINTER
(03791) ;
0540A F05054B9      MOVMR R5,PSBIT1
0540C 9C500000      ADDIR R5,SBD
0540E 9A5003FF      ANDIR R5,S03FF
05490 E05054B9      MOVMR R5,PSBIT1
(03796) ;
(03797) ;--- NEXT BASE OF FRAME
(03798)      MOVMR R3,PSBASE
05494 9C300000      ADDIR R3,SBD
05496 9A3003FF      ANDIR R3,S03FF
05498 E03054B8      MOVMR R3,PSBASE
(03802) ;
(03803) ;--- NEXT SYNC VALUE
(03804)      MOVIR R7,S0001
0549A 90700001      XORRM R7,VSSVNC
0549C F07054CF      ;
(03806) ;
(03807) ;--- FUNCTION LIST SELECTOR ISL2
(03808)      MOVMR R2,SIDSISL
0549E F02054B6      INCR R2,1
054A0 2621          MOVMR R3,ISVTS(R2)
054A1 F0340502      NEG R3
054A3 0030        MOVMR R3,ISVTS(R2)
054A4 E0340502      ;
(03813) ;
(03814) ;--- ALL DONE. SO RETURN
(03815) ;
(03816) ;
(03817) ;--- UPDATE XMTR BEFORE RETURNING
(03818) ;
054A6 0E70      DCS90      RETURN
(03819)      MOVMR R1,SIDSISL
054A7 2612      INCR R1,2
054A8 9C100502  ADDIR R1,ISVTS
054AA 00004036  JN0      TX0
SID FOR FUNC SEL.
POINT TO IFSEL3
COMPUT ADDR FOR IFSEL3
GO TO XMTR UPDATE MODULE

(03824) ;
(03825) ;--- LEAVE SPACE FOR VARIOUS VARIABLES
(03826) ;
(03827) ;
054AC 00000000  BASICB DATA F'0.0'
054AE 00000000  BASOQB DATA F'0.0'
054B0 00000000  BASDECT DATA F'0.0'
054B2 00000000  BASDECB DATA 00
054B4 0000      BASFCB DATA 00
054B6 0000      SIDSTB DATA 00

```

```

PAGE 96:      MAP MODULES FOR THE P A R C ALGORITHM      --- JAN. 35, 1988

85486 8888      (83834) SIDISL      SS
85487 8888      (83835) PSSHAT     SS
85488 8888      (83836) PSBASE     SS
85489 8888      (83837) PSBIT1     SS
8548A 8888      (83838) PSINFO     SS
8548B 8888      (83839) PSMAX      10F'.B.'

854CF 8881      (83840) VSSYNC      S1
854D0 8888      (83841) VST        SS
854D1 8888      (83842) VSBETA     SS
854D2 8888      (83843) VSPH8      SS
854D3 4888      (83844) VSPAR      $4888
854D4 FFE8      (83845) VSTHR      DATA $FFE8
854D5 8888      (83846) VSCAP      SS
854D6 F388      (83847) SZSSHAT     DATA $388
854D7 8888      (83848) TSLIM      DATA $58
                        (83849) *LSRUN  DATA SD
                        (83850) ;
                        (83851) ;
                        (83852) ;--- UPDATE TOP OF MEMORY
                        (83853) ;
                        888854D8      TOESNEW=0L
                        88888288      #L=TOESPRT
                        88288 881854D8 ADDR TOESNEW(.BUS18)
                        8828A      END

;SIZE=944

```


PAGE 98: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

DC821:	MS33E (MS484) (MS421)	
DC828:	MS347 (MS431) (MS479)	
DC831:	MS353 (MS448) (MS448)	
DC832:	MS361 (MS456) (MS465)	
DC833:	MS377 (MS478) (MS478)	
DC838:	MS3A2 (MS503) (MS532)	
DC851:	MS388 (MS558) (MS554)	
DC860:	MS38F (MS566) (MS574)	
DC861:	MS3D1 (MS584) (MS584)	
DC862:	MS3D4 (MS587) (MS595)	
DC863:	MS3E8 (MS608) (MS608)	
DC878:	MS3FC (MS629) (MS688)	
DC871:	MS3FE (MS627) (MS631)	(MS644) (MS664) (MS673)
DC872:	MS418 (MS641) (MS658)	
DC873:	MS424 (MS642) (MS678)	
DC874:	MS426 (MS673)	
DC875:	MS428 (MS646) (MS666)	(MS677)
DC876:	MS428 (MS688)	
DC877:	MS433 (MS692) (MS722)	
DC878:	MS448 (MS716) (MS717)	
DC879:	MS44E (MS722) (MS722)	
DC888:	MS482 (MS753) (MS727)	
DC881:	MS488 (MS681) (MS731)	
DC882:	MS486 (MS736)	
DC883:	MS481 (MS744) (MS745)	
DC884:	MS484 (MS688) (MS728)	(MS728) (MS723) (MS729) (MS739) (MS758)
DC886:	MS478 (MS756) (MS764)	
DC887:	MS478 (MS769) (MS771)	
DC889:	MS481 (MS768) (MS765)	(MS775)
DC890:	MS486 (MS348) (MS723)	(MS725) (MS725)
DC893:	MS182 (MS777) (MS728)	
DEC818:	MS2E6 (MS217) (MS316)	
DEC828:	MS3DF (MS219) (MS397)	
DEC88:	MS314 (MS293) (MS383)	
DEC728:	MS487 (MS232) (MS363)	
DEC721:	MS43C (MS233) (MS378)	
DMY8:	MS794 (MS436) (MS918)	(MS911) (MS913) (MS914)
EN8188:	MS0CD (MS286)	
EN8181:	MS0D5 (MS2592)	(MS2688)
EN8182:	MS0E4 (MS258)	(MS2588) (MS2684)
EN8111:	MS0E2 (MS2597)	(MS2688)
EN838:	MS0F6 (MS2412)	(MS2416)
EN848:	MS085 (MS2429)	(MS2437)
EN858:	MS018 (MS2446)	(MS2448)

--- JAN. 30, 1988

MAP MODULES FOR THE P A R C ALGORITHM

PAGE 99:

```

EN001: 0001F (02442) (02451)
EN002: 00020 (02451) (02459)
EN003: 00030 (02476) (02483)
EN004: 0004A (02478) (02486) (02505) (02535)
EN005: 00054 (02480) (02491) (02549)
EN006: 00066 (02492) (02502)
EN007: 00068 (02489) (02509)
EN008: 0006E (02511) (02516)
EN009: 0007D (02522) (02532)
EN010: 00092 (02515) (02539)
EN011: 00095 (02541) (02547)
EN012: 000A3 (02513) (02553)
EN013: 000A6 (02555) (02562)
EN014: 000B2 (02498) (02566)
EN015: 000BC (02572) (02582)
EN016: 000C2 (00076) (02363)
EN017: 0007B (02126) (02228)
EN018: 00081 (00063)
EN019: 0006E (00074) (00078)
EN020: 000FB (00462) (00463)
EN021: 000F2 (01040) (01041)
EN022: 000F3 (01734) (01735)
EN023: 000C4 (00058) (00075)
EN024: 000A2 (01381) (01486) (01488)
EN025: 0002A (00047)
EN026: 0006F (00098) (00953)
EN027: 00028 (01831) (01833)
EN028: 000BC (01118) (01112)
EN029: 0010D (02993) (03046)
EN030: 00297 (02994) (03245)
EN031: 002A9 (02995) (03289)
EN032: 002C8 (02996) (03287)
EN033: 002D0 (02997) (03387)
EN034: 00355 (02999) (03441)
EN035: 00363 (03000) (03457)
EN036: 00376 (03001) (03476)
EN037: 00383 (03002) (03494)
EN038: 003C1 (03003) (03567)
EN039: 00306 (03004) (03588)
EN040: 00408 (03005) (03632)
EN041: 00435 (03006) (03693)
EN042: 00112 (02364) (02424) (02455) (02657)
EN043: 00176 (02951) (02952)
EN044: 00193 (02979) (02988)

```

PAGE 100: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1968

```

INCSZ:  #4A0F (#0277) (#0070)
INCSZ:  #4A0F (#0279) (#0095)
INFSB:  #4A0B (#0275) (#0081)
INFTM:  #4A07 (#0273) (#0034)
INCSZ:  #4AF1 (#0281) (#0085)
INCSZ:  #4A4F (#0271) (#0029)
INCSZ:  #4F1B (#0284) (#2107)
INRC:  #4CB1 (#0282) (#1500)
INTX:  #004A (#0900) (#0900)
IN1:  #0047 (#0094) (#0054)
IN2:  #0043 (#0090) (#0092)
IN4:  #615E (#2017)
IN8:  #615F (#2751) (#2709)
IN8OUT: #486C (#0081) (#0269)
INIT8:  #5170 (#2944)
ISVT8:  #0002 (#0059) (#0064) (#0065) (#0066) (#0067) (#0068) (#0130) (#0203) (#0270) (#0347)
ISVT18: #0003 (#0064) (#0136) (#0276) (#2784) (#2932) (#2945) (#3141) (#3143) (#3145) (#3347)
ISVT28: #0004 (#0065) (#0138) (#0367) (#0369) (#0386) (#3517) (#3519) (#3521) (#3810) (#3812) (#3822)
ISVT38: #0005 (#0066) (#0147) (#0276)
ISVT48: #0006 (#0067) (#0152) (#0278)
ISVT58: #0007 (#0068) (#0280)
LSRUN:  #6115 (#0295) (#2509) (#2539) (#2553) (#2660) (#3653) (#3661) (#3713)
LOOP8:  #0008 (#0506) (#0619) (#0634)
LOOP1:  #001F (#0544) (#0583)
LOOP2:  #0006 (#0490) (#0511)
LOOP4:  #000C (#0720) (#0754)
LOOP6:  #0009 (#0730) (#0751)
LOOP8:  #0034 (#0591) (#0600)
LOOP7:  #004E (#0612) (#0621)
LOOP8:  #0050 (#0625) (#0633)
LOOP9:  #0050 (#0655) (#0678)
N00:  #0000 (#0435) (#0707) (#0788) (#0930) (#1407) (#1408) (#1557) (#2044) (#2045) (#2059)
N00FF: #00FF (#0062) (#0201)
N0RC:  #0466 (#0134) (#0355) (#0456) (#0457)
N0TX:  #0430 (#0202) (#0206) (#0455)
NEGA:  #0007 (#1206) (#1433)
NEXT1: #0027 (#0036) (#0038)
N0MAX: #0009 (#2097) (#2950) (#3023) (#3025) (#3108)
N0FTR: #0030 (#0039) (#0067) (#0075)
NULL1: #000E (#1404)
O0001: #5262 (#3104) (#3190)

```

PAGE 181: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 35, 1985

```

OQB2: 0526A (03105) (03196)
OQB3: 0519A (02967) (02978)
OQB11: 051E2 (02968) (03054)
OQB12: 051EA (02969) (03061)
OQB13: 051FB (02970) (03069)
OQB14: 051F8 (02971) (03076)
OQB15: 05206 (02972) (03098)
OQB26: 0547C (03422) (03768)
OQB27: 05487 (03423) (03781)
PBASE: 05488 (02299) (03036) (03125) (03128) (03358) (03361) (03587) (03518) (03798) (03881)
      (03836)
PBB17: 05489 (03135) (03238) (03255) (03272) (03288) (03429) (03618) (03792) (03795) (03837)
PBIPO: 0548A (03331) (03617) (03624) (03788) (03838)
PSMAX: 0548B (02951) (03826) (03827) (03894) (03183) (03189) (03839)
PSMAT: 05487 (03283) (03852) (03751) (03775) (03835)
PAR81: 050F1 (02622) (02627)
PAROUT8: 050EF (02588) (02638) (02546) (02561) (02588) (02598) (02621)
PCRC8: 04086 (01737) (01751)
PCRC8SA: 05088 (01748) (01764) (02818)
PCRC8SZ: 0509A (01749) (02818)
PCTX8: 0508B (01054) (01057) (01444)
PCTX8SA: 0508C (01055) (01444)
PCTX8SZ: 0508D (00476) (00493) (00788)
PERIOD: 05029 (00538) (00566)
PICH38: 040A6 (00485) (00479) (00788)
PICH38SA: 0508B (00476) (00493) (00788)
PICH38SZ: 0508B (00477) (00788)
PRE1: 0502C (01849) (01981)
PRE2: 0503F (01071) (01077)
PRE3: 05011 (01105) (01113) (01136) (01382)
OCKP: 0501D (01147) (01149)
QUAN: 05046 (01092) (01093) (01097)
QUAN1: 05052 (01272) (01273) (01268)
QUAN2: 0506D (01283) (01442)
QUAN3: 05065 (01294) (01439)
QUAN4: 05075 (01381) (01311)
QUAN5: 05078 (01318) (01314) (01317)
QUAN6: 0508F (01278) (01441)
RC8: 04088 (00779) (00126)
RCIN18: 04098 (00982) (00346)
RCPI1: 050AF (01197) (01198) (01422)
RECORD: 05068 (00691) (00942)
RESET: 050A9 (00782) (00788)

```

PAGE 182: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```

SSTX:      04A1D (00213) (00222) (00289) (00789)
SC81B:     051C8 (03026) (03029)
SC82B:     051DA (03048) (03083)
SC821:     051E9 (03088) (03082)
SC822:     051EE (03088) (03088)
SC823:     051F7 (03073) (03077)
SC824:     051F9 (03064) (03081)
SC831:     0520B (03092) (03096) (03101) (03103)
SC841:     0520F (03109) (03111)
SC851:     0521C (03118) (03121)
SC852:     05261 (03191) (03192)
SC853:     05298 (03244) (03250)
SC854:     052AB (03257) (03266)
SC8541:    052BE (03270) (03280)
SC855:     052C6 (03296) (03293)
SC856:     052D8 (03303) (03321)
SC857:     052D8 (03306) (03314)
SC858:     052EF (03318) (03327)
SC859:     052F8 (03345)
SC87B:     052FD (03110) (03353)
SH8RC:     04F0F (00141) (00151) (00349) (02092)
SHIFT:     00051 (01007) (01090)
SI081SL:   054B6 (02931) (03139) (03345) (03365) (03384) (03515) (03808) (03834)
SI08TB:    054B5 (03403) (03417) (03532) (03023)
SIGNAL:     00090 (01029) (02007)
SIZES:      05160 (02775) (02019)
START:      00000 (01497)
SVT8:       00302 (00060) (00120) (00442) (02426) (03161) (03165) (03535) (03541)
SVT1118:   00464 (00454) (00486) (02231)
SVT1119:   00468 (00457) (00458) (02038)
SVT1178:   0046C (00488) (00459) (00784)
SVT1109:   0046E (00459) (01405) (00783)
SVT509:     003E6 (00442) (00443) (00047)
SVT518:    003E8 (00443) (00444) (00047)
SVT549:     003EE (00444) (00445) (00076)
SVT568:     003F2 (00445) (00446) (00924)
SVT639:     00404 (00446) (00447) (01400) (01643)
SVT689:     0040E (00447) (00448) (01506)
SVT709:     00412 (00448) (00449) (01529)
SVT749:     0041A (00449) (00450) (01558)
SVT798:     00426 (00450) (00451) (01603)
SVT839:     0042E (00451) (00452) (01642)
SVT988:     00446 (00452) (00453) (02040)
SVT998:     0044A (00453) (00454) (02233)

```

PAGE 183: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

SYNCS:	SB1CB (B2934) (B3823)		
SV38FLGS:	1FFCE (B8861) (B8143)	(B8146) (B8216) (B8218) (B8363)	
SZ8SHAT:	SB4DB (B3281) (B3758)	(B3847)	
TSLIM:	SB4DB (B3848)		
TSRC:	SB4F13 (B8132) (B2895)		
TOESNEW:	SB4DB (B3854) (B3856)		
TOESPRAT:	SB288 (B8438) (B3855)		
TR81B:	SB12A (B2748) (B2755)		
TR82B:	SB13B (B2739) (B2766)		
TR82B:	SB13F (B2762) (B2775)		
TR83B:	SB14B (B2795) (B2884)		
TR8DC:	SB116 (B8878) (B2735)		
TRAN:	SB888 (B1772) (B1783)		
TX8:	SB436 (B8888) (B8281)	(B3823)	
VB8ETA:	SB4D1 (B3327) (B3548)	(B3688) (B3842)	
VBDEL:	SB114 (B8297) (B2474)	(B2659)	
VBGAP:	SB4DB (B3812) (B3551)	(B3846)	
VBPAR:	SB4D3 (B3336) (B3489)	(B3581) (B3844)	
VBPHB:	SB4D2 (B3388) (B3319)	(B3546) (B3581)	(B3682) (B3843)
VBSYN:	SB113 (B2411) (B2418)	(B2658)	
VSSYN:	SB4CF (B3842) (B3128)	(B3354) (B3492) (B3848)	
VST:	SB4DB (B3295) (B3534)	(B3576) (B3841)	
VSTHR:	SB4D4 (B3655) (B3787)	(B3845)	
VADAP:	SB858 (B2192)		
VCHECK:	SB872 (B1585)	(B1521) (B1633) (B1635)	
VCONT1:	SB846 (B1867) (B1568)	(B1578)	
VENB:	SB876 (B1634) (B1641)		
VGAP:	SB81A (B1813) (B1825)		
VH8RC:	SB4F2D (B8133) (B8148)	(B8348) (B2124)	
VH8TX:	SB4A25 (B8289) (B8212)	(B8288) (B8795)	
VLOOP:	SB823 (B1829) (B1633)		
VLOOP1:	SB836 (B8868) (B8866)		
VLOOP2:	SB88E (B8799) (B8811)		
VLOOP3:	SB81A (B8798) (B8811)		
VLOOP4:	SB81C (B8813) (B8816)		
VLOP1:	SB82D (B2189) (B2118)		
VLOP2:	SB871 (B2221) (B2223)		
VNEGA:	SB888 (B1591) (B1593)	(B1594)	
VPCRC8:	SB4EC2 (B1738) (B2819)	(B2826) (B2252)	
VPCRC8A:	SB4F86 (B2822) (B2242)		
VPCRC8I:	SB4F8E (B2818) (B2258)		
VPCRC8S:	SB813 (B2519) (B2559)		
VPCRC8Z:	SB18B (B2821) (B2252)		
VPCTX8:	SB4C84 (B1844) (B1453)	(B1459) (B1662)	

PAGE 184: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1985

VPCTH8A: B4C92 (B1486) (B1682)
 VPCTH8I: B4B02 (B1482) (B1668)
 VPCTH8S: B0030 (B1483) (B1687)
 VPCTH8Z: B0109 (B1488) (B1682) (B0773) (B0969)
 VPICH38: B4A8E (B0486) (B0767)
 VPICH38A: B4A16 (B0778) (B0968)
 VPICH38I: B4AF6 (B0766) (B0967)
 VPICH38S: B0060 (B0767) (B0930)
 VPICH38Z: B00EA (B0769) (B0969)
 VPRE1: B0037 (B2134) (B2222)
 VQ1: B004C (B2166) (B2166) (B2169)
 VQ3: B0061 (B2165) (B2178) (B1584)
 VQUARI: B0046 (B1579) (B1581)
 VQUAH2: B004C (B1587) (B1588)
 VQUAH3: B004F (B1579) (B1588)
 VTRAR: B000D (B2047) (B2052)
 WAIT: B0086 (B0630) (B0631)
 WRD4: B0003 (B0009) (B0126)
 WRD6: B0004 (B0078) (B0129)

LINE WITH ERRORS: B (MAP VERSION B00101.1B) E- B

PAGE 11

```

(00001)
(00002)
(00003)
(00004)
(00005)
(00006)
(00007)
(00008)
(00009)
(00010)
(00011)
(00012)
(00013)
(00014)
(00015)
(00016)
(00017)
(00018)
(00019)
(00020)
(00021)
(00022)
(00023)
(00024)
(00025)
(00026)
(00027)
(00028)
(00029)
(00030)
(00031)
(00032)
(00033)
(00034)
(00035)
(00036)
(00037)
(00038)
(00039)
(00040)
(00041)
(00042)
(00043)
(00044)

      *** IOSDC ***
      ARVIND S. ARORA
      FEB 11, 1988

      IOS-2 PROGRAM TO INTERFACE THE BITSTREAM FROM THE PARC ALGORITHM
      TO THE MODEN. THE SOURCE CODER AT THE TRANSMITTER STORES THE OUTPUT
      IN A DOUBLE BUFFER, AND THE DECODER REQUIRES ITS INPUT TO BE IN A
      CIRCULAR BUFFER. THIS PROGRAM OUTPUTS THE CONTENTS OF THE DOUBLE
      BUFFERS THROUGH THE IOS2 TO THE DIGITAL INTERFACE, AND INPUTS THE
      RECEIVED BITSTREAM OF THE DIGITAL INTERFACE TO THE CIRCULAR BUFFER.
      THE PROGRAM OPERATES IN FULL DUPLEX FOR COMPATIBILITY WITH THE
      RS-423 MODEN USED IN THE PRESENT SYSTEM.

      REGISTER & FLAG USAGE:
      RB: XMTR BUFFER SWITCH, RB=ADSTXA TO INDICATE BUFFER 1
      R1: XMTR BUFFER ADDRESS POINTER
      R2: RCVR BUFFER ADDRESS POINTER
      F1: FLAG TO INDICATE MODE OF OPERATION,
          F1-CLEAR TO INDICATE INITIALIZE
          F1-SET TO INDICATE NORMAL OPERATION
      P1: FLAG SET INDICATES RECEIVER DATUM AVAILABLE
      P2: FLAG SET INDICATES XMTR READY FOR NEXT DATUM

      ----- IOS-2 MODULE
      ---
      SYMBOL DEFINITIONS
      ADSTXA=D'26700'
      ADSTXB=D'27700'
      SZSTX =D'100'
      ENDSTXA=ADSTXA+SZSTX-1
      ENDSTXB=ADSTXB+SZSTX-1
      AD8RCG=D'29000'
      SZ8RC=D'1024'
      END8RC=AD8RCG+SZ8RC-1
      CLCRATE=8ECC4

```


PAGE 4:

```

ADDRCC: 07140 (00040) (00042) (00061) (00100) (00092)
ADSTXA: 06464 (00034) (00037) (00060) (00091) (00082)
ADSTXB: 06C34 (00035) (00038) (00073) (00081) (00082)
CLCRATE: 06C04 (00044) (00063)
ENDSRCC: 07547 (00042) (00099)
ENDSTXA: 06520 (00037) (00070)
ENDSTXB: 06C00 (00038) (00080)
IOSINIT: 00004 (00060) (00063) (00080) (00093) (00099) (00101)
IOSLOOP: 00007 (00066) (00070) (00083) (00088) (00099) (00101)
IOSSTOP: 00020 (00103)
IOSDCS: 00010 (00067) (00097)
RCCS: 00400 (00041) (00042) (00030)
SZSRC: 00000 (00036) (00037) (00072)
TXS: 00000 (00066) (00072)
TXAS: 00000 (00077)
TXBS: 00017 (00073) (00087)

```

LINES WITH ERRORS: 0 (MAP VERSION 000101.10) E- 0