

AD-A101 428

RENSSELAER POLYTECHNIC INST TROY NY DEPT OF ELECTRIC--ETC F/6 17/2
DESIGN AND IMPLEMENTATION OF THE INTERACTIVE COMMUNICATIONS SIM--ETC(U)
APR 81 J W MODESTINO, K R MATIS, K Y JUNG F30602-78-C-0083

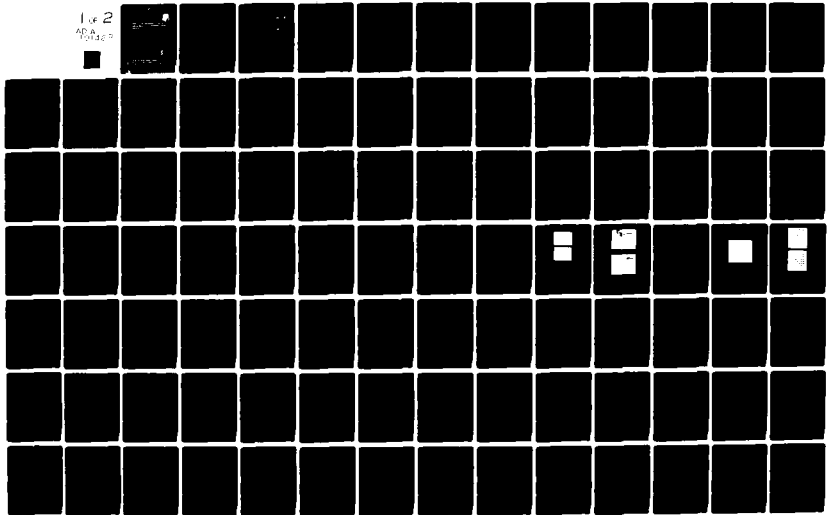
UNCLASSIFIED

TR-80-1

RADC-TR-81-37

NL

1 of 2
AD-A101 428



LEVEL

##

12

RADC-TR-81-37
Final Technical Report
April 1981



AD A101428

DESIGN AND IMPLEMENTATION OF THE INTERACTIVE COMMUNICATIONS SIMULATOR (ICS)

Rensselaer Polytechnic Institute

**James W. Modestino
Kirt R. Matis**

**K. Y. Jung
Acie L. Vickers**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

S DTIC ELECTE D
JUL 16 1981
H

**ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441**

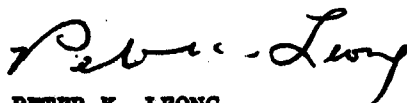
FILE COPY

81 7 15 004

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

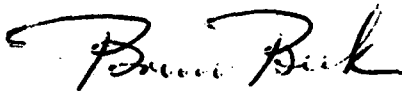
RADC-TR-81-37 has been reviewed and is approved for publication.

APPROVED:



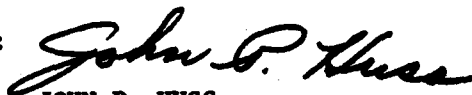
PETER K. LEONG
Project Engineer

APPROVED:



BRUNO BEEK
Acting Technical Director
Communications and Control Division

FOR THE COMMANDER:



JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (DCLF) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 18) RADC TR-81-37	2. GOVT ACCESSION NO. A707428	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6) DESIGN AND IMPLEMENTATION OF THE INTERACTIVE COMMUNICATIONS SIMULATOR (ICS)		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Apr 1978 - Apr 1981 PERFORMING ORG. REPORT NUMBER TR80-1
7. AUTHOR(s) 12) James W. Modestino K. Y. Jung Kirt R. Matis Acie L. Vickers		8. CONTRACT OR GRANT NUMBER(s) 15) F30602-78-C-0083
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rensselaer Polytechnic Institute Troy NY 12181 D-111E-7SE		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 16) 451920P7 1720
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (DCLF) Griffiss AFB NY 13441		12. REPORT DATE 11) Apr 1981
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		13. NUMBER OF PAGES 91
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same		
18. SUPPLEMENTARY NOTES RADC Project Engineer: Peter K. Leong (DCLF)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Digital Communications Source Encoding/Decoding Modeling/Simulation Channel Encoding/Decoding Interactive Graphics Display Modulation/Demodulation Transceiver Functions Propagation Channel Model		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the development and capabilities of a fairly comprehensive system for the digital simulation of a wide variety of point-to-point digital communication systems. This system is called the Interactive Communications Simulator (ICS). The ICS is a flexible, graphics oriented, high speed, and highly interactive hardware/software system consisting of a PDP 11-40 minicomputer acting as host to a fast peripheral array processor, the Floating Point Systems AP-120B. Its		

402653

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

modeling structure is the classical breakout of the various generic signal processing functions in any communication link, from source to sink. The signal processing functions are predominantly modeled in terms of their complex envelope representations for uniformity, ease, and accuracy of analyses. It is fully supported by an extensive graphics support package and many powerful analysis subroutines, to facilitate user-interactions, analyses, and output displays. The modeling and simulation tasks are optimally partitioned between the PDP 11/40 host and the AP-120B peripheral array processor to ensure ease-of-use and highly efficient manipulations. The ICS also features "realistic" channel models, in addition to the analytically expedient Additive White Gaussian Noise (AWGN) channel, so that the performance and behavior of all modeled transceiver functions can be more exactly assessed and specified. All simulation modules are written in the AP Assembly Language, and the system software, graphics support package and analysis subroutines are written in DEC FORTRAN IV.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Table of Contents

	Page no.
1.0 INTRODUCTION	1
2.0 GENERAL BACKGROUND	1
3.0 FUNCTIONAL DESCRIPTION OF THE ICS	3
3.1 SYSTEMS MODEL	3
Channel Encoder	6
Modulator/Transmitter	7
Channel	16
Demodulator/Receiver	27
Channel Decoder	32
3.2 OPERATION OF THE ICS	34
3.3 TYPICAL GRAPHICAL OUTPUT	38
4.0 SOFTWARE DESCRIPTION	47
4.1 FORTRAN DRIVER PROGRAM	48
4.2 DATA STRUCTURES	48
4.3 SYSTEM SOFTWARE CONNECTION	64
4.4 AP-120B PROGRAM REQUIREMENTS	68
5.0 GRAPHICS SUPPORT PACKAGE	69
5.1 PLOTTING ROUTINES	69
The Command Mode	75
5.2 GRAPHICS LIBRARY	76
6.0 FUTURE ICS ENHANCEMENTS	85
6.1 HARDWARE IMPROVEMENTS	89
REFERENCES	90

List of Figures

Figure #		Page no.
Figure 1	Hardware Configuration Supporting Interactive Communications Simulator (ICS)	4
Figure 2	Block Diagram of General Communication System	5
Figure 3	Illustration of $K=3$, $R=1/2$ Convolutional Encoder	8
Figure 4	General Narrowband Filtering Operation.	15
Figure 5	Overall Block Diagram of Channel Model.	17
Figure 6	Tapped Delay Line Model of Diffuse Multipath Generator	19
Figure 7	Illustration of Impulsive Channel Noise Model	21
Figure 8	Simple Pulse Waveform Model for Low-Density Shot Noise	25
Figure 9	Predetection Portion of Receiver.	28
Figure 10	Model for Zero-Memory Nonlinearities of the Limiting Variety	30
Figure 11	Basic Decision Feedback Equalizer (DFE) Structure	33
Figure 12	Software Organization of Interactive Communications Simulator	39
Figure 13	Typical Baseband Signaling Waveforms for BPSK Modulation on AWGN Channel Employing $K=6$, $R=1/2$ Convolutional Code.	42
Figure 14	Typical Channel Signaling Waveforms	43
Figure 15	Typical Phase and Bit Synchronization Errors for Coherent BPSK System.	45
Figure 16	Plots of Symbol and Bit Error Probability for BPSK System Operating on AWGN Channel and Employing $K=6$, $R=1/2$ Convolutional Code with Viterbi Decoding	46
Figure 17	Data Flows in Operation of the ICS	49
Figure 18.1 thru 18.11	Overall Logic Flow of Driver Program	50- 60

List of Figures cont'd.

Figure 19	Channel Data Block Diagram	62
Figure 20	Logic for Sliding Data Stream Structure.	63
Figure 21	Dynamic Addressing Scheme in ICS	65
Figure 22.1 and 22.2	ICS Software Connection.	66 67
Figure 23.1 and 23.2	Flowchart for Plotting Routines.	72 73

List of Tables

		Page no.
Table 1	Description of Some Typical Modulation Capabilities Available in Interactive Communications Simulator	12
Table 2	Pulse Waveforms Available in Impulse Noise Generator	26
Table 3	Time-Domain Plotting Programs Used in VALIDATION Mode	70
Table 4	Breakdown of Different Plotting Schemes	71

Acknowledgement

It is a pleasure to acknowledge the significant contributions of the many students in the Electrical and Systems Engineering Department at RPI who have participated in this effort. This includes D. Brown, R. Chang, E. Godreau, V. Eyuboglu, D. Hayward, T. Murakami, A. Ningo, R. Palmer, and P. Sher.

1.0 INTRODUCTION:

During the past two years an extensive effort has been underway within the Electrical and Systems Engineering Department at Rensselaer Polytechnic Institute (RPI) to develop a fairly comprehensive system for the digital simulation of a wide variety of point-to-point digital communication systems. This effort has resulted in the Interactive Communications Simulator (ICS), a flexible, graphics oriented, and highly interactive hardware/software system consisting of a typical minicomputer acting as host to a fast peripheral array processor.

The ICS is resident at RPI while a similar version has been installed in a compatible hardware system at the Digital Communications Experimental Facility (DICEF) at RADC. This system is being employed both to evaluate existing modem performance and to explore new modulation/coding concepts appropriate for military, commercial, and space applications.

The purpose of the present report is to document the development and present state of this system while indicating possible directions for future enhancements. In Section 2 we provide some general background on the ICS. This is followed by a detailed functional description in Section 3. The overall software structure is described in Section 4 with additional details on the various interactive graphics routines provided in Section 5. Finally, a brief description of future enhancement possibilities is provided in Section 6.

2.0 GENERAL BACKGROUND:

Digital simulation provides a useful and effective adjunct to direct analytical evaluation of communication system performance. Indeed, there are situations where explicit performance evaluation defies analysis and results can be obtained only through either actual hardware evaluation or

digital simulation. The speed and flexibility associated with digital simulation generally provides a compelling reason for adopting this approach.

Unfortunately, an accurate simulation of most communication systems of even moderate complexity can be terribly time consuming on present-day general purpose machines. This is due to the large number of repetitive signal processing operations that must be performed in order to obtain a statistically valid measure of system performance. Recently a class of fast floating-point array processors has become available which greatly facilitates these repetitive signal processing operations. The use of the word "array" in this context is intended to indicate a processor optimized for handling large data arrays in distinction to the large machines organized as parallel arrays of independent processing elements. These array processors are intended to be employed as peripheral floating-point processors in conjunction with a general-purpose host computer which provides overall system control. This hardware configuration provides a unique opportunity for accurate and statistically meaningful digital simulation of existing and future communication systems.

The RPI developed ICS has exploited the potential of this hardware configuration. This has resulted in an extensive hardware/software system for the digital simulation of arbitrary point-to-point communication systems. While the system can be extended to include analog communications, we will be primarily interested in digital communication systems. This system makes extensive use of interactive computer graphics and includes a Digital Equipment Corporation (DEC) PDP-11/40 acting as host to a Floating Point Systems, Inc., AP-120B floating-point array processor. A block diagram of the hardware configuration supporting the ICS software is illustrated in Fig. 1.

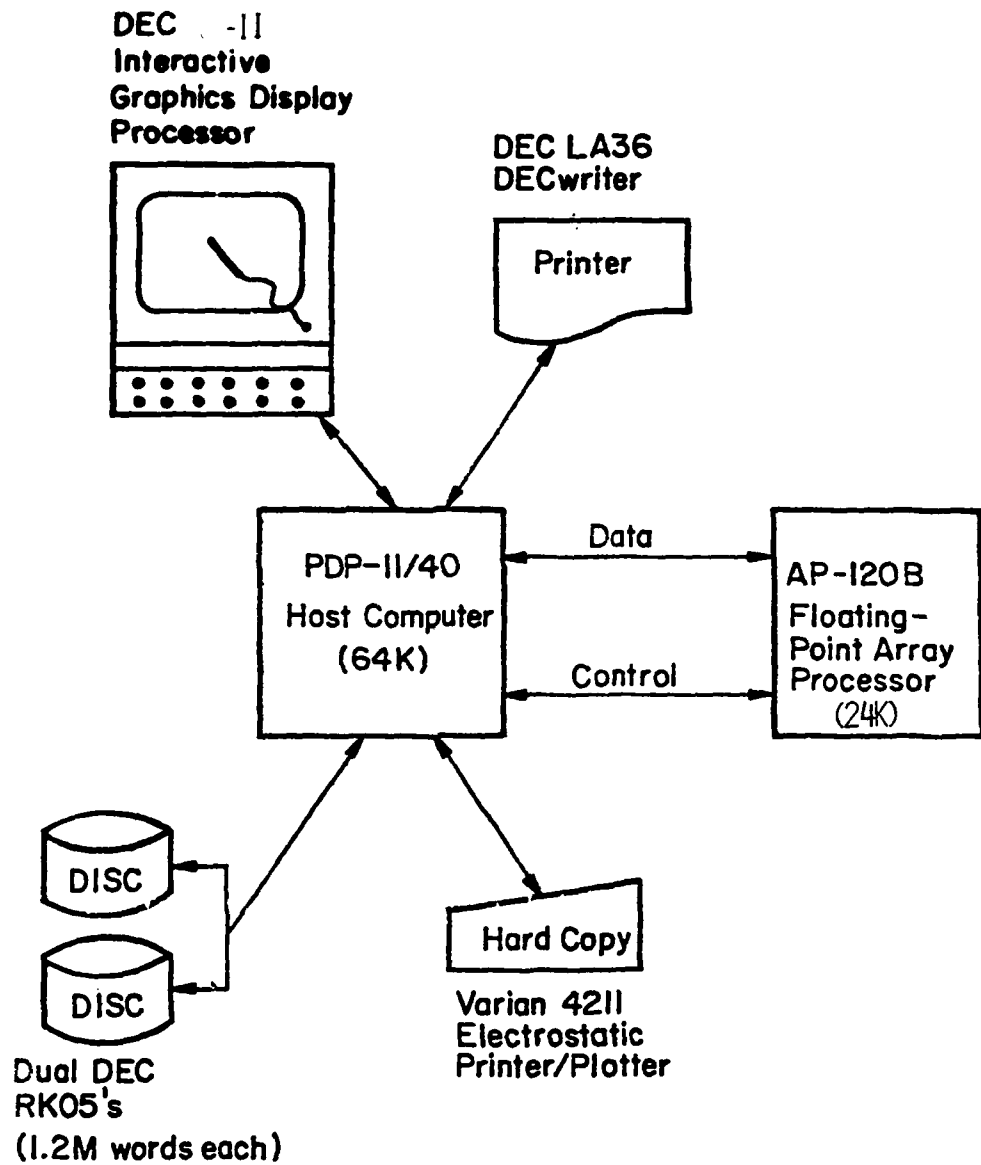


Figure 1

Hardware Configuration Supporting Interactive Communications Simulator (ICS)

The user can configure a wide variety of digital communication systems from basic modules provided as system facilities. These facilities include: channel coders/decoders; modulators/demodulators (modems) for a number of modulation strategies; receiver front-end filtering; and various bit synchronization and phase tracking algorithms. In addition to additive white Gaussian noise (AWGN) channels, the simulator includes the ability to model impulsive noise channels as well as fading-dispersive channels typical of say HF or Tropospheric scatter channels.

3.0 FUNCTIONAL DESCRIPTION OF THE ICS:

In this section we provide a complete functional description of the ICS. We begin with a discussion of the general systems model and proceed to a description of the various usage modes of the ICS. Finally, we discuss some typical graphical output from the ICS.

3.1 SYSTEMS MODEL:

A Block diagram of a generic communication system which provides a model for simulation software development is illustrated in Fig. 2. The information source generally produces a discrete sequence or analog waveform which is to be encoded, transmitted over a specified channel, reconstructed and delivered to a remote destination or information sink. In Fig. 2, the purpose of the source encoder is to encode the source output, presumably in an efficient fashion, into a binary sequence $\{a_i\}$ for subsequent encoding and transmission. The source decoder at the remote destination provides the inverse function. That is, it employs the binary sequence $\{\hat{a}_i\}$ to reconstruct an approximation to the source output. The binary sequence $\{\hat{a}_i\}$ may differ from the actual transmitted sequence $\{a_i\}$ due to channel errors. This has an obvious effect on the accuracy with which the source decoder can reconstruct the source output.

While source coding/decoding schemes could easily be incorporated into the

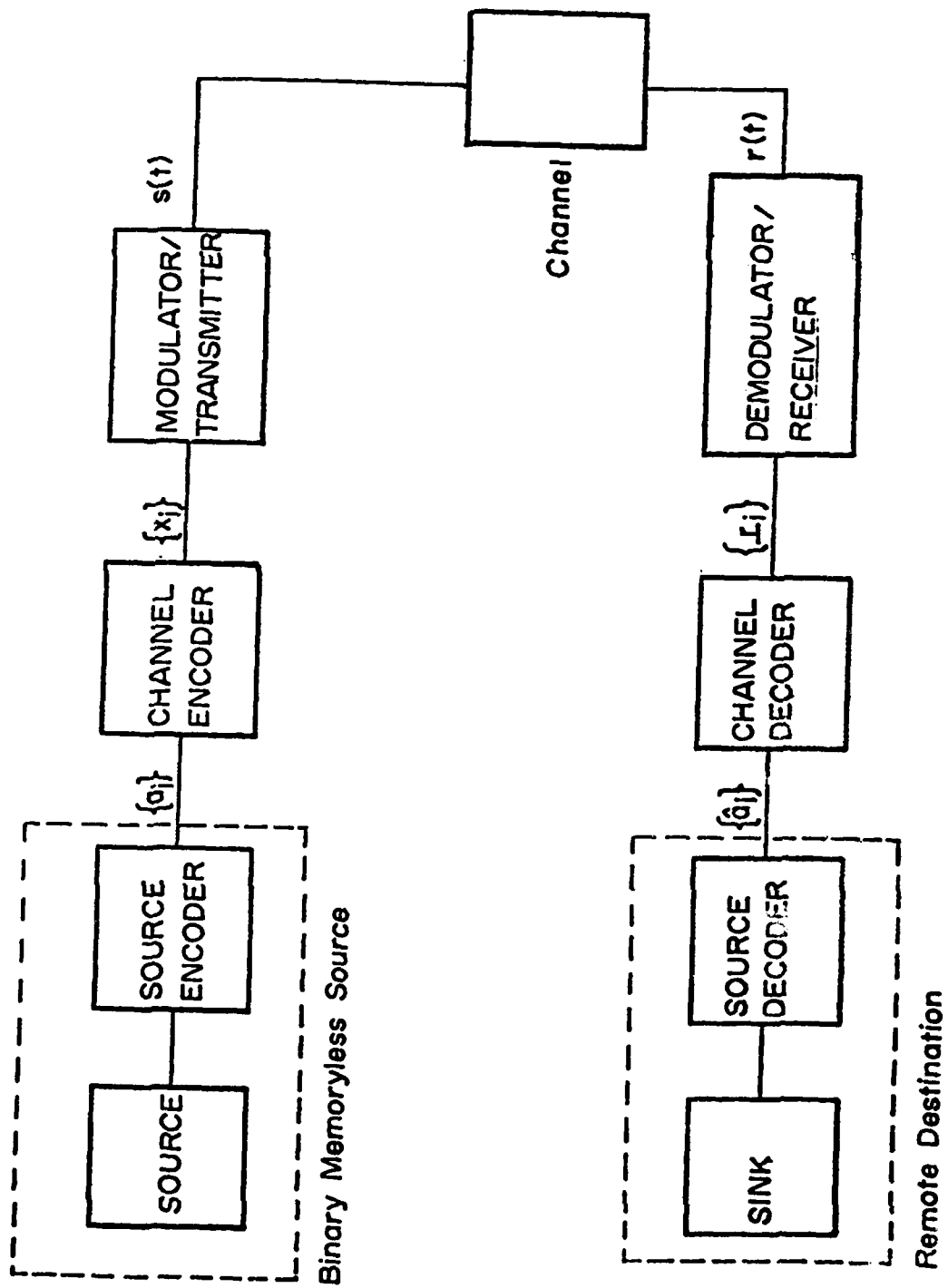


Figure 2
Block Diagram of General Communication System

simulator, we have chosen not to do so for a variety of reasons. As a result, we assume that the combination of source and source encoder can be modeled as a binary memoryless source as indicated in Fig. 2. The major criterion of system performance is then the bit error probability P_b , i.e., the probability that $\hat{a}_i \neq a_i$. This quantity can be evaluated by straightforward Monte-Carlo simulation. We will see, however, that there are other modes of usage of this system. Each of the remaining elements in Fig. 2 will be described in detail.

Channel Encoder

The purpose of the channel encoder is to accept the binary sequence $\{a_i\}$ at its input and through the insertion of controlled redundancy produce the M-ary sequence $\{x_i\}$ at its output. Generally, the encoded output sequence is such that $x_i \in \{0, 1, \dots, M-1\}$, although in the binary case it will be convenient to assume that $x_i = \pm 1$, $i = 1, 2, \dots$. Both block and convolutional channel encoding capabilities are provided in the simulator. In the former, the binary sequence $\{a_i\}$ is segmented into blocks of size k to which $n-k$ redundant bits are added to produce a rate $R=k/n$ code measured in units of information bits per transmitted channel symbol. There is complete independence between blocks. Convolutional codes, on the other hand, do not impose a block structure on the input information sequence and produce a sliding dependency over a span of input symbols. The encoder can be implemented as a shift register into which information bits are shifted b at a time. After each shift n modulo two sums are computed from the contents of selected shift register stages. These n modulo two sums then constitute the encoder output sequence associated with a particular branch of b input information bits. The normalized code rate in this case is $R=b/n$. Note that each input information bit affects[†] $[k/b]n$ encoder output

[†] The notation $[x]$ means the largest integer not exceeding x .

symbols where K is the length of the shift register. The quantity K is called the constraint length of the code. A typical $K=3$, $R=1/2$ encoder is shown in Fig. 3. Tabulations of good convolutional code constructions are available for various K and R (cf. [1], [2], [19]).

The existing block and/or convolutional codes are generally effective in correcting random errors and are seriously deficient in their ability to correct burst errors. In several important applications, the channel environments are known to introduce burst errors due to time-correlated fading, multipath, co-channel interference, jamming, etc. There are few constructive coding approaches for channels with memory. Indeed, the few existing approaches are extremely sensitive to channel modeling assumptions which are often gross estimates. In the vast majority of existing systems the approach is to employ appropriate interleaving schemes in an effort to make the channel appear memoryless. In particular, the serial channel symbol sequence $\{x_i\}$ is interleaved or scrambled in such a fashion that successive channel symbols are transmitted separated by many channel signaling intervals. This interleaving is generally quite effective in reducing the effect of burst errors and allows use of the well-developed classes of random error-correcting codes. At this time an interleaving capability has not been provided in the ICS. This is one of the capabilities which should be included in any future ICS enhancements as described in Section 6.

Modulator/Transmitter

The purpose of the modulator/transmitter is to map the channel encoder output sequence $\{x_i\}$ (possibly interleaved) into a waveform $s(t)$ suitable for transmission over the channel. Encoder outputs are presented to the modulator/transmitter every T_s seconds producing a time-limited channel signaling waveform which vanishes outside an interval of total duration T_s seconds. The quantity T_s is called the baud interval, or equivalently $f_s=1/T_s$ is the

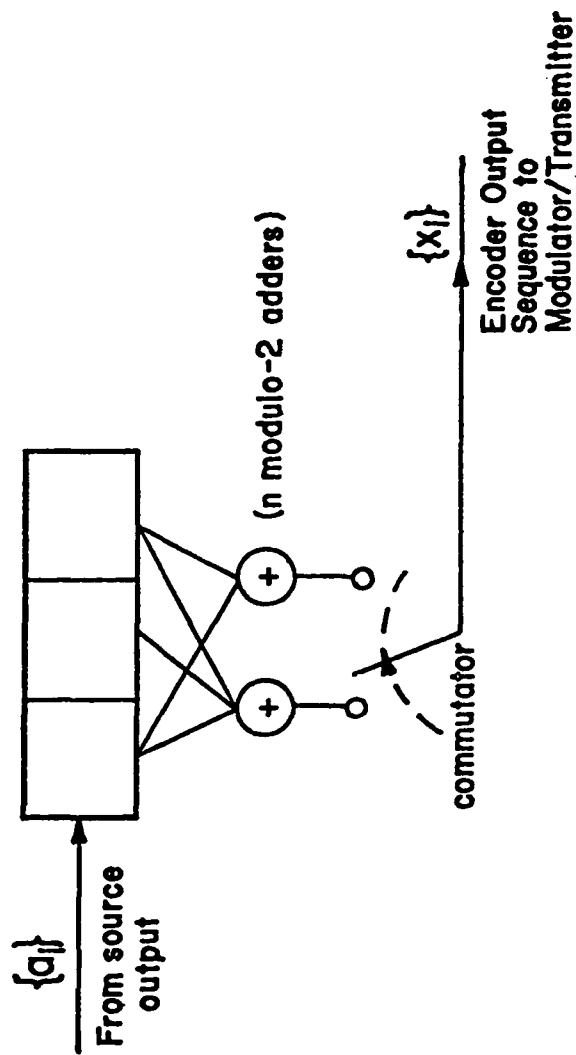


Figure 3
 Illustration of $K=3$, $R=1/2$
 Convolutional Encoder

baud rate. We have found it convenient to normalize all frequency (time) domain quantities to the baud rate (interval). For example, the bandwidths of frequency selective channel filtering elements are all normalized to f_g . This allows the user the ability to configure and execute a simulation program independent of the actual baud rate. This facilitates user interaction with the communications simulator and provides simulation results in a convenient parametric form.

In simulating the channel signaling waveform $s(t)$ we make extensive use of complex notation. This allows simulation of a bandpass signal in terms of its lowpass complex envelope and eliminates the need for simulating a high-frequency carrier component. Specifically, we assume that $s(t)$ can be expressed as

$$s(t) = \sqrt{2} \operatorname{Re}\{\tilde{s}(t)e^{j2\pi f_c t}\}, \quad (1)$$

where f_c is an assumed known carrier component and $\tilde{s}(t)$ represents the complex envelope which can be represented in the form

$$\tilde{s}(t) = s_c(t) - js_s(t), \quad (2)$$

where $s_c(t)$ and $s_s(t)$ are real quantities representing the inphase and quadrature (I/Q) components respectively. By generating only the I/Q components, the simulator need not be concerned with the explicit details of the RF portions of the system. In particular, discrete samples of the I/Q components are generated at equally spaced intervals.

In the construction of the ICS it has proven sufficient to consider that $\tilde{s}(t)$ can be expressed as

$$\tilde{s}(t) = \sqrt{\frac{E_s}{T_s}} \sum [C_{1,i} u_c(t - iT_s - \tau) + jC_{2,i} u_s(t - iT_s - \tau)] e^{j\theta}, \quad (3)$$

where τ and θ represent random timing epoch and carrier phase respectively. The receiver will be required to estimate and track these initially unknown quantities. Here the sequences $\{C_{1,i}\}$ and $\{C_{2,i}\}$ are determined from the channel encoder output sequence $\{x_i\}$, while $u_c(t)$ and $u_s(t)$ are elementary baseband signaling waveforms. By appropriate choice of the mappings $\{C_{1,i}\}$ and $\{C_{2,i}\}$, together with the baseband signaling waveforms $u_c(t)$ and $u_s(t)$, the expression for $\tilde{s}(t)$ is sufficiently general to include most known modulation formats. This includes: coherent binary phase-shift keying (BPSK), differentially coherent phase-shift keying (DPSK), quadrature phase-shift keying (QPSK), noncoherent frequency-shift keying (FSK), and quadrature amplitude-shift keying (QASK). In the case of the offset or staggered quadrature phase-shift keying (OQPSK), and the constant envelope formats such as continuous phase frequency shift keying (CPFSK) and the special case of minimum shift keying (MSK) (cf. [3], [4]) this formulation must be modified slightly. Specifically, in this case

$$\tilde{s}(t) = \sqrt{\frac{E_s}{T_s}} \sum [C_{1,i} u_c(t - 2iT_s - \tau) + jC_{2,i} u_s(t - 2iT_s - \tau)] e^{j\theta}, \quad (4)$$

where now the elementary baseband waveforms are translated by two baud intervals at a time.

In the case of coherent BPSK, for example, we have $C_{2,i} = 0, u_s(t) = 0$ while $C_{1,i} = x_i$ (assume $x_i = \pm 1$) and $u_c(t)$ is equal to the pulse-like waveform[†]

$$\begin{aligned} u_0(t) &= 1 & ; & & 0 \leq t \leq T_s \\ &= 0 & ; & & \text{elsewhere.} \end{aligned} \quad (5)$$

Similarly, in the case of noncoherent binary frequency-shift keyed (BFSK) modulation it is easily shown that $C_{1,i} = \cos\theta_i + j\sin\theta_i$ while $C_{2,i} = x_i C_{1,i}$. Here $\{\theta_i\}$ represents a sequence of independent and identically distributed (i.i.d.) phases uniformly distributed on $[-\pi, \pi]$. Without this latter assumption, the phase could be estimated at the receiver on the basis of past transmissions thereby violating our assumption of noncoherent reception. The corresponding baseband signaling waveforms are given by

$$u_c(t) = \cos(\Delta\omega t/2) u_0(t) \quad , \quad (6a)$$

and

$$u_s(t) = \sin(\Delta\omega t/2) u_0(t) \quad , \quad (6b)$$

where $\Delta\omega$ is some integer multiple of $2\pi/T_s$ in order to insure orthogonality between transmitted tones. Additional choices of $\{C_{k,i}\}$ $k=1,2$, and associated baseband pulseshapes $u_c(t)$ and $u_s(t)$ are provided in Table 1.

It should be noted that in constructing Table 1 we have imposed the constraint

[†] The pulse waveform $u_0(t)$ in Table 1 has value unity over the interval $[-T_s, T_s]$ and zero elsewhere.

Table 1
Description of Some Typical Modulation Capabilities
Available in Interactive Communications Simulator

Modulation	$C_{1,i}$	$C_{2,i}$	$u_c(t)$	$u_s(t)$	Remarks
BPSK	x_i	0	$u_0(t)$	0	$x_i = \pm 1$
DPSK	$x_i C_{1,i-1}$	0	$u_0(t)$	0	$x_i = \pm 1$ assumed $C_{1,0}$ known
BFSK	$e^{j\theta_i}$	$x_i e^{j\theta_i}$	$\cos(\Delta\omega t/2)u_0(t)$	$\sin(\Delta\omega t/2)u_0(t)$	$x_i = \pm 1$ $\Delta\omega = 2\pi k/T$ θ_i - i.i.d. uniform
QPSK	x_{2i}	x_{2i+1}	$(1/\sqrt{2})u_0(t)$	$(1/\sqrt{2})u_0(t)$	$x_i = \pm 1$
MSK	x_{2i}	x_{2i+1}	$\cos(\pi t/2T_B)u_0'(t)$	$\sin(\pi t/2T_S)u_0'(t-T_S)$	$x_i = \pm 1$ I/Q waveforms repeated every $2T_B$ sec.

Note: $u_0(t)$ and $u_0'(t)$ defined in text.

$$\int_{(i-1)T_s}^{iT_s} |\tilde{s}(t)|^2 dt = E_s; \quad i=1,2,\dots \quad (7)$$

so that E_s represents the constant signal energy per channel signaling element or baud. Indeed, since the simulation is to be performed digitally we must replace (7) by

$$\sum_{k=N(i-1)}^{Ni-1} |\tilde{s}(k\Delta T_s)|^2 \Delta T_s = E_s; \quad i=1,2,\dots \quad (8)$$

where N is the number of samples per baud and is under user control. Typical choices are $N=8, 16, \text{ or } 32$. The quantity $\Delta T_s = T_s/N$ is, of course, the sampling period. Observe that (7) is indeed satisfied for each of the entries in Table 1.

The communications simulator then generates the lowpass I/Q samples $\{s_c(k\Delta T_s)\}$ and $\{s_s(k\Delta T_s)\}$. For example, in the case of coherent BPSK described above we have

$$s_c(k\Delta T_s) = x_i \sqrt{2E_s/T_s} \cos\theta; \quad (i-1)N \leq k < iN \quad (9a)$$

while

$$s_s(k\Delta T_s) = x_i \sqrt{2E_s/T_s} \sin\theta; \quad (i-1)N \leq k < iN \quad (9b)$$

corresponding to the i 'th successive baud interval, $i=1,2,\dots$

Finally, we should mention one last function provided in the modulator/transmitter module. In particular, any narrowband filtering or nonlinear amplifier distortion effects will be simulated as part of the modulator/transmitter. For example, if $h(t)$ represents the impulse response of a narrowband or bandpass filter, then it can be expressed in the form

$$h(t) = 2\text{Re}\{\tilde{h}(t)e^{j2\pi f_c t}\} \quad (10)$$

where $\tilde{h}(t) = h_c(t) - j h_s(t)$ represents the baseband equivalent impulse response. Here $h_c(t)$ and $h_s(t)$ are real lowpass functions representing inphase and quadrature filtering effects respectively. Let $\tilde{s}_0(t)$ represent the complex envelope at the output of this filter excited by $\tilde{s}(t)$ at its input. That is, $\tilde{s}_0(t) = \tilde{s}(t) \otimes \tilde{h}(t)$ where \otimes represents convolution. Expressing $\tilde{s}_0(t)$ in terms of its inphase and quadrature components according to

$$\tilde{s}_0(t) = s_{0c}(t) - j s_{0s}(t) \quad (11)$$

we have

$$s_{0c}(t) = s_c(t) \otimes h_c(t) - s_s(t) \otimes h_s(t) \quad (12a)$$

and

$$s_{0s}(t) = s_c(t) \otimes h_s(t) + s_s(t) \otimes h_c(t) \quad (12b)$$

which is illustrated more conveniently in Fig. 4. A variety of filtering options are available to the user in the ICS. These filters are implemented as digital filters and include such options as linear-phase finite impulse response (FIR) filters and a variety of infinite impulse response (IIR) filters such as maximally-flat Butterworth response and Chebyshev equiripple response filters. Filter parameters such as cutoff frequency (normalized to the baud rate), roll-off characteristics, passband phase and amplitude characteristics are chosen by the user from an appropriately formatted menu list. While it would be possible to provide various nonlinear saturation effects associated with the RF power amplifier in the modulator/transmitter module, such a capability is not present in the ICS at this time.

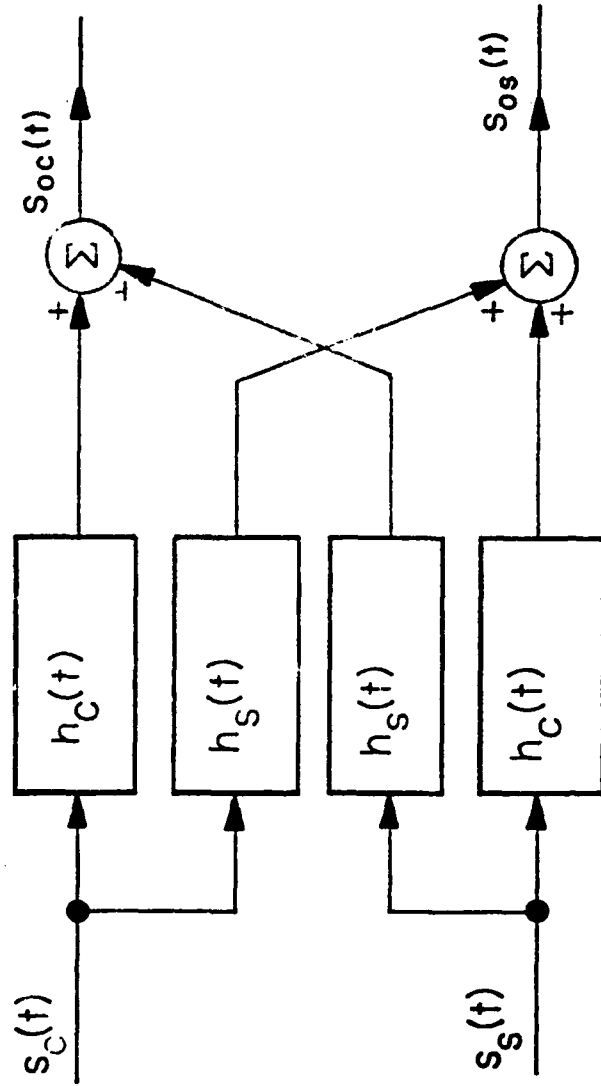


Figure 4
General Narrowband Filtering Operation

Channel

In choosing a channel model for the ICS it was important to provide a fairly general approach which encompasses the entire range of potential applications. A useful channel model which we feel satisfies this requirement consists of a three component fading multipath channel plus additive noise. In particular, the fading multipath channel is assumed to consist of a single direct path, a specular multipath component, and a diffuse multipath component as illustrated in Fig. 5. Here $\tilde{s}(t)$ represents the complex envelope of the transmitted signal component which we assume can be expressed in terms of its inphase and quadrature components according to $\tilde{s}(t) = s_c(t) - js_s(t)$. The quantities ζ_s and ζ_d in Fig. 5 represent respectively the specular and diffuse signal energies normalized to the energy in the direct path. These parameters are under direct user control. In addition, the user is allowed to adjust the relative phase ϕ of the specular multipath component relative to the direct path. Other parameter choices include the nominal differential delay τ_0 and the differential doppler f_0 associated with the two multipath components. The delay and doppler spreads about these nominal values will be determined by choice of the channel scattering function $\sigma(\tau, f)$ associated with the diffuse multipath component. In this context, the quantity $\sigma(\tau, f) d\tau df$ represents the energy associated with delays in the range $(\tau, \tau+d\tau)$ and doppler shifts in the range $(f, f+df)$.

The diffuse multipath component has been implemented by means of the tapped delay line model illustrated in Fig. 6. This is a fairly general model for fading dispersive channels under the so called wide-sense stationary uncorrelated scattering (WSSUS) assumption (cf. [5], [6]). The delay T in Fig. 6 is assumed equal to the Nyquist rate $T=1/W$ where W is the transmitted signal bandwidth. In practice, a sampling rate somewhat higher than this may be desirable in some applications.

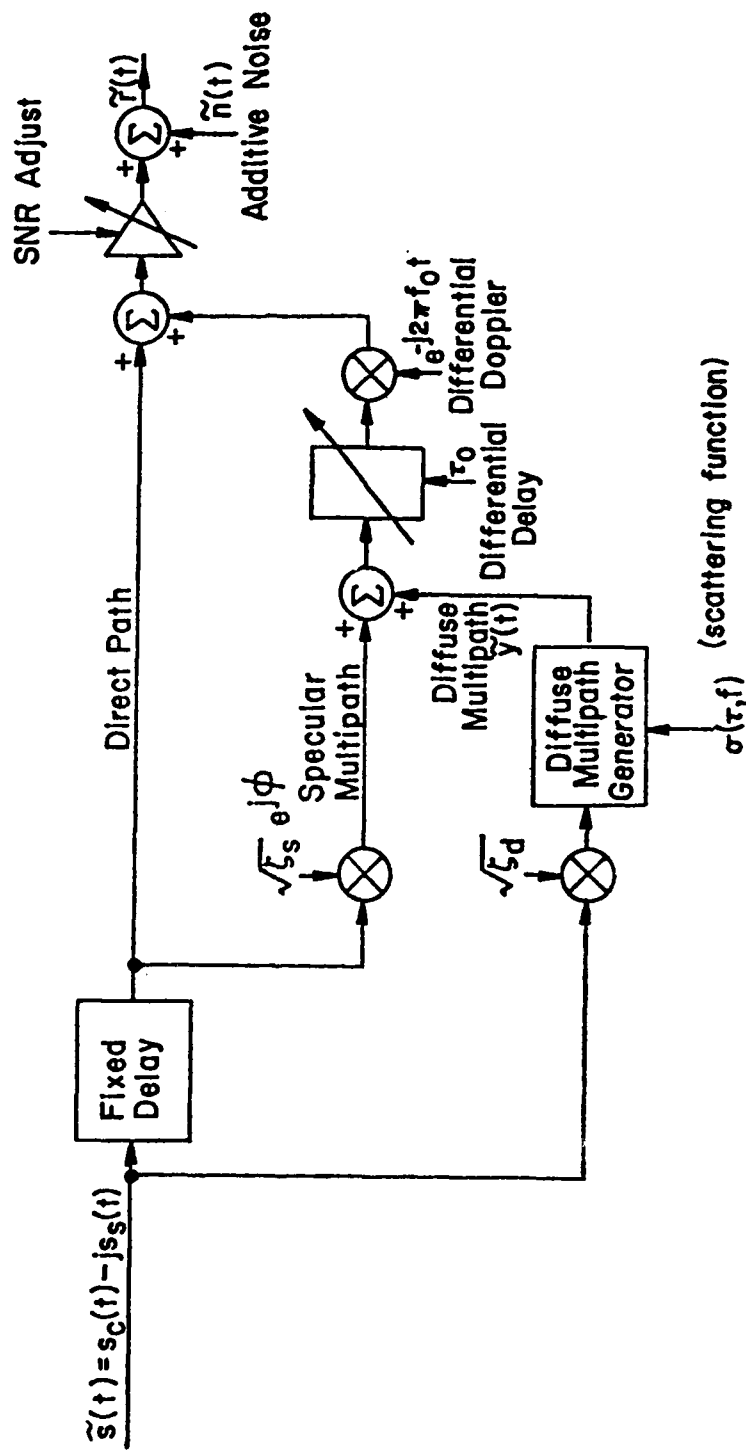


Figure 5

Overall Block Diagram of Channel Model

The quantities $\tilde{g}_i(t)$, $i=0, \pm 1, \dots, \pm N$ in Fig. 6 represent tap gain functions which are modeled as mutually independent complex zero-mean wide-sense stationary Gaussian processes. In the communications simulator we have restricted all the tap gain functions to possess second-order Butterworth power spectral densities of the form

$$S_i(f) = \frac{2}{\pi} \frac{B_i^3}{B_i^4 + f^4}; \quad i=0, \pm 1, \dots, \pm N \quad , \quad (13)$$

where B_i represents the 3 dB bandwidth in Hz. Specification of the 3 dB bandwidths B_i , $i=0, \pm 1, \pm 2, \dots, \pm N$ then effectively determines the doppler as a function of delay profile as illustrated in Fig. 6. Similarly, specification of the multiplying factors σ_i , $i=0, \pm 1, \dots, \pm N$, determines the delay profile. The parameters N , B_i , and σ_i , $i=0, \pm 1, \dots, \pm N$, are specified by the user. The tap gain functions are then obtained as the output of appropriately defined digital filters excited by white Gaussian noise inputs. While other more general doppler-as-a-function-of-delay profiles are easily incorporated into the ICS, we feel that the approach described here will prove adequate for the intended applications.

Note that the channel model as illustrated in Fig. 6 includes provision for an additive noise component. An adequate model for a wide range of applications, including ELF/VLF and HF, can be represented as the linear combination of a low-density shot noise process and additive white Gaussian noise (AWGN). That is, the complex envelope $\tilde{n}(t)$ of the additive bandpass noise can be expressed as

$$\tilde{n}(t) = \tilde{y}(t) + \tilde{w}(t) \quad . \quad (14)$$

Here $\tilde{y}(t)$ represents a complex low-density shot noise process while $\tilde{w}(t)$

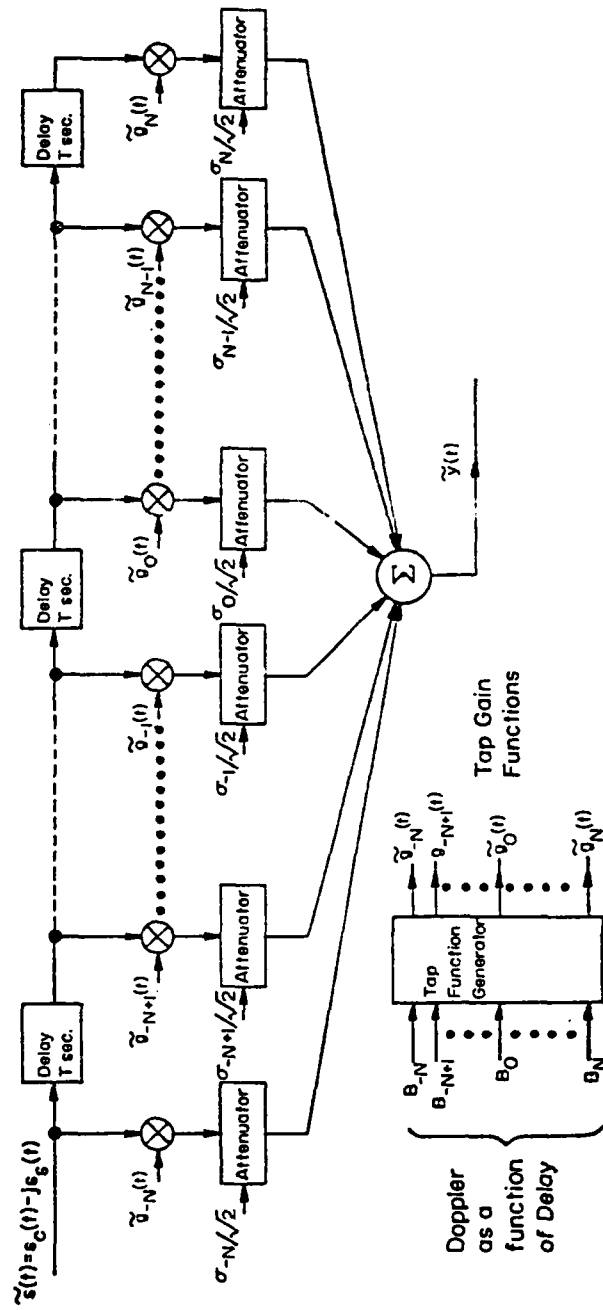


Figure 6
Tapped Delay Line Model of Diffuse Multipath Generator

represents complex WGN with double-sided noise spectral density $N_0/2$ watts/Hz. By low-density we mean that the interarrival times of the impulses are relatively long compared to typical impulse durations. The low-density shot noise component $\tilde{y}(t)$ accounts for the relatively infrequent high-level and time-resolved impulse noise hits due to atmospheric discharge phenomenon or various contributions due to man-made noise. The Gaussian noise component $\tilde{w}(t)$, on the other hand, is due to a combination of front-end noise and the large number of low-level and overlapping impulse hits. The latter contribution is expected to exhibit Gaussian behavior from central limit theorem considerations.

The low-density shot noise component can be modeled as the output of a linear time-invariant filter excited by an amplitude modulated impulse train or point process at its input. The filter generating the complex low-density shot process $\tilde{y}(t)$ is assumed to possess impulse response $\tilde{h}(t)$, or equivalently system transfer function $\tilde{H}(s)$. A block diagram of the channel noise model is illustrated in Fig. 7. The impulse response $\tilde{h}(t)$ will in general be complex possessing an inphase component $h_c(t)$ and a quadrature component $h_s(t)$. As indicated in Fig. 7, the input to the linear filter generating the low-density shot noise component $\tilde{y}(t)$ is $\tilde{u}(t)$, assumed to be of the form

$$\tilde{u}(t) = \sum_{i=0}^{N(t)} \tilde{u}_i \delta(t-t_i) . \quad (15)$$

Here $\{\tilde{u}_i\}$ is a sequence of complex independent and identically distributed (i.i.d.) random variables which can be expressed in terms of I/Q components according to

$$\tilde{u}_i = u_{ci} - ju_{si} \quad ; \quad i=0,1,\dots \quad (16)$$

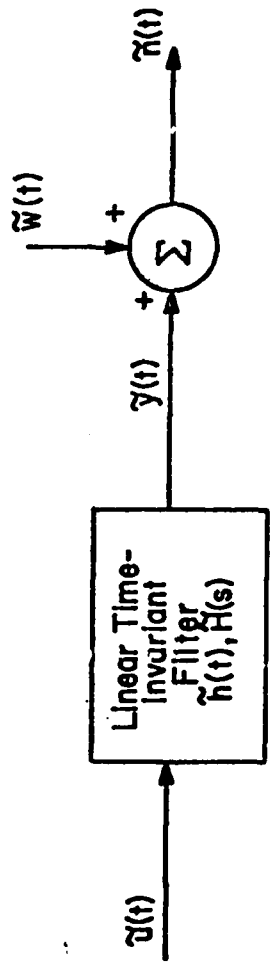


Figure 7
Illustration of Impulsive Channel Noise Model

The process $\{N(t), t \geq 0\}$ in (15) is a point process, in particular a counting process, whose event times are described by the sequence $\{t_i\}$. More specifically, $\{N(t), t \geq 0\}$ undergoes a unit jump at each of the event times $\{t_i\}$. The Poisson process is a good example of a counting process. The communications simulator possesses the ability to simulate stationary renewal counting processes possessing Gamma distributed interarrival times with p.d.f.

$$f(x) = \frac{x^{\nu-1} \exp\{-x/\beta\}}{\Gamma(\nu)\beta^\nu} \quad ; \quad x \geq 0 \quad , \quad (17)$$

where $\nu \geq 1$ and the parameter β is defined according to $\beta = 1/\nu\lambda$ with $\lambda \geq 0$ fixed. The quantity λ represents the average number of events per unit time. This quantity is specified by the user in units normalized to the baud duration T_s seconds. Specification of the two parameters ν and λ , then, completely defines the point process $\{N(t), t \geq 0\}$.

For example, if $\nu=1$, then

$$f(x) = \lambda e^{-\lambda x} \quad ; \quad x \geq 0 \quad , \quad (18)$$

This is the exponential distribution. In this case, $\{N(t), t \geq 0\}$ is Poisson with λ the average number of hits per unit time. Another special case is obtained by letting $\nu \rightarrow \infty$ while holding λ fixed. The result is

$$f(x) = \delta(x-1/\lambda) \quad (19)$$

where $\delta(\cdot)$ is the delta function. In this case, the impulses are periodic with rate λ per unit time. By introducing the class of driving point processes described here, we have a parameterized impulsive noise model which includes these two extremes as special cases. The class of point processes $\{N(t), t \geq 0\}$, and hence $\bar{u}(t)$, is easily generated on a digital machine.

We have assumed that the I/Q components $\{u_{ci}\}$ and $\{u_{si}\}$ can be described by

$$u_{ci} = R_i \cos \theta_i, \quad (20a)$$

and

$$u_{si} = R_i \sin \theta_i, \quad (20b)$$

with $\{R_i\}$ an i.i.d. sequence possessing common probability density function (p.d.f.) $f_r(\cdot)$ while $\{\theta_i\}$ is likewise an i.i.d. sequence uniformly distributed over $[-\pi, \pi]$. Related work [7]-[8] on modeling impulse noise phenomenon indicate that an appropriate choice of p.d.f. $f_r(\cdot)$ is the lognormal distribution

$$f_r(R) = \frac{1}{\sqrt{2\pi\sigma R}} \exp\left\{-\frac{(\ln R - \mu)^2}{2\sigma^2}\right\}; \quad R \geq 0, \quad (21)$$

where μ and σ^2 are the mean and variance, respectively, of a Gaussian variate g for which $r = e^g$. Another useful choice is the power-Rayleigh distribution defined by

$$f_r(R) = \frac{\alpha}{R_0^\alpha} R^{\alpha-1} \exp\{-(R/R_0)^\alpha\}; \quad R \geq 0, \quad (22)$$

where the characteristic exponent α lies in the range $0 < \alpha \leq 2$ and R_0 is a scale parameter to be specified. The sequence $\{\bar{u}_i\}$ is thus easily generated digitally. Some guidance on choosing parameter values to match observed impulse noise characteristics is provided in [7]-[8].

The low-density shot noise process $\tilde{y}(t)$ appearing at the output of the linear filter in response to $\tilde{u}(t)$ at its input can be expressed as

$$\tilde{y}(t) = y_c(t) - j y_s(t) \quad , \quad (23)$$

where the I/Q components $y_c(t)$ and $y_s(t)$ can be generated according to

$$y_c(t) = \sum_{i=0}^{N(t)} R_i [h_c(t-t_i) \cos \theta_i - h_s(t-t_i) \sin \theta_i] \quad , \quad (24a)$$

and

$$y_s(t) = \sum_{i=0}^{N(t)} R_i [h_s(t-t_i) \cos \theta_i + h_c(t-t_i) \sin \theta_i] \quad . \quad (24b)$$

Here $h_c(t)$ and $h_s(t)$ are, as described previously, the I/Q components of the complex impulse response function $\tilde{h}(t)$. We will assume that $h_c(t)$ and $h_s(t)$ are both equal to the same impulse response function $h_0(t)$. Several choices for $h_0(t)$ are possible. For example, the simplest choice is the pulse waveform of duration t_c seconds normalized to the baud duration as illustrated in Fig. 8. The corresponding impulse response is

$$h_0(t) = u_{-1}(t) - u_{-1}(t-t_c) \quad (25)$$

where $u_{-1}(t)$ is the unit step function

$$u_{-1}(t) = \begin{cases} 1 & ; \quad t \geq 0 \\ 0 & ; \quad t < 0 \end{cases} \quad (26)$$

The duration t_c is specified by the user such that $t_c \ll 1/\lambda$ so that the impulse process is truly low-density. Other choices of pulse waveform available are illustrated in Table 2.

Finally, the WGN component which is added to the low-density shot noise component to complete the channel noise model is described in terms of its I/Q components according to

$$\tilde{w}(t) = w_c(t) - j w_s(t) \quad (27)$$

The components $w_c(t)$ and $w_s(t)$ will be modeled as mutually independent zero-mean

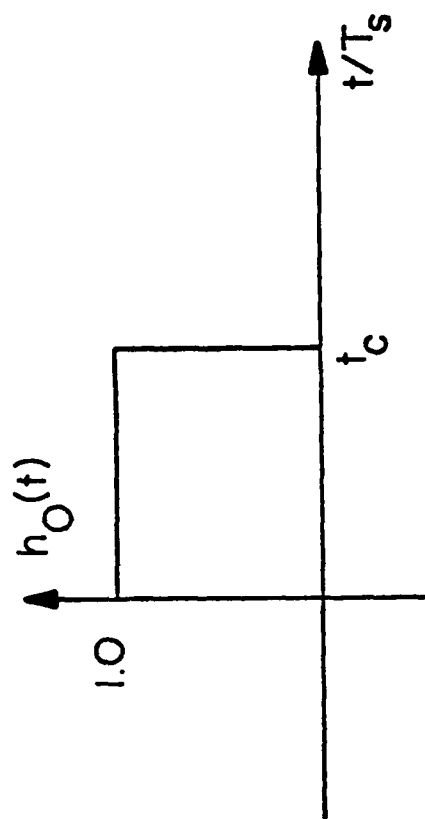


Figure 8
Simple Pulse Waveform Model for Low-Density Shot Noise

Table 2

Pulse Waveforms Available in Impulse Noise Generator

Pulse Type	Impulse Response $h_0(t)$	System Transfer Function $H_0(s)$
Unipolar Pulse	$u_{-1}(t) - u_{-1}(t - t_c)$	$\frac{1 - e^{-st_c}}{s}$
Exponential Pulse	$e^{-at} u_{-1}(t)$	$\frac{1}{s+a}$
Decaying Sine	$\frac{1}{\omega_c} e^{-at} \sin \omega_c t u_{-1}(t)$	$\frac{1}{(s+a)^2 + \omega_c^2}$
Decaying Cosine	$e^{-at} \cos \omega_c t u_{-1}(t)$	$\frac{s+a}{(s+a)^2 + \omega_c^2}$

white Gaussian noise processes, each possessing double-sided noise spectral density $N_0/2$ watts/Hz. These components are easily simulated.

Demodulator/Receiver

The main purpose of the demodulator/receiver function module is to perform data demodulation of the channel output $r(t)$. That is, to produce the sequence $\{r_i\}$ indicated in Fig. 2 which is applied as input to the channel decoder. The demodulator/receiver produces an output once per baud interval. We have purposefully allowed $\{r_i\}$ to be a vector sequence to accommodate various soft decision quantization or decoding schemes. For example, in an M-ary system the receiver might well produce an ordered list of the l largest outputs ($l < M$) during successive channel signaling intervals. This is the so-called list-of- l quantization scheme [9] which provides a reasonably efficient and computationally effective means for implementing soft-decision decoding in a variety of M-ary signaling situations.

The demodulator/receiver module contains a predetection front-end which precedes the data demodulator and/or other portions of this module. This predetection section is illustrated in Fig. 9 and consists of a zero-memory nonlinear (ZMNL) device sandwiched between two narrowband filters. The filters are implemented as described previously for the output filtering provided in the modulator/transmitter module. A ZMNL processing capability has been provided in the ICS to allow simulation of various unintentional RF saturation efforts, such as in a satellite transponder, and intentional nonlinear processing, such as might be used to mitigate against the effects of impulse noise.

The various ZMNL processing schemes are generally of the limiting variety and have been implemented in the ICS as illustrated in Fig. 10. Here the envelope $R_0(t) = \sqrt{s_{0c}^2(t) + s_{0s}^2(t)}$ of the complex envelope $\tilde{s}_0(t)$ is formed and passed

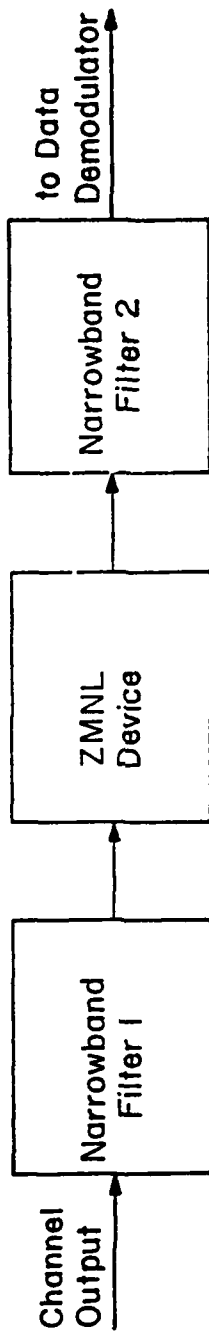


Figure 9
Predetection Portion of Receiver

through an instantaneous or zero-memory nonlinear (ZMNL) device with input/output characteristic $g[R_0(t)]$ depending only upon the instantaneous envelope $R_0(t)$.

Forming the product

$$\tilde{s}'_0(t) = g[R_0(t)]\tilde{s}_0(t) \quad , \quad (28)$$

we can synthesize a variety of nonlinearities of the limiting or saturation type by appropriate choice of the function $g(R)$. These nonlinearities are all characterized by the fact that they depend upon the envelope and not the inphase and quadrature components individually. For example, the choice $g(R)=1/R$ can easily be shown to correspond to an ideal bandpass limiter. The choice

$$g(R) = \begin{cases} 1 & ; \quad 0 < R < 1 \\ 1/R & ; \quad R > 1 \end{cases} \quad , \quad (29)$$

on the other hand, represents a soft limiter or clipper characteristic. A wide variety of nonlinear characteristics are provided under user control in an attempt to provide a realistic and yet computationally efficient model for demodulator/receiver nonlinear effects.

It should be noted that in order to perform its basic data demodulation function, the demodulator/receiver module must perform a variety of ancillary functions. These ancillary functions include symbol synchronization, phase and/or frequency tracking, and adaptive symbol equalization. In addition, data symbol de-interleaving is performed within the demodulator/receiver module. It is these ancillary functions which seriously complicate the design of this function module. The ICS also has the capability of patching symbol synchronization and carrier phase and/or frequency from transmitter to receiver. This facility is useful, for example, when one would like to isolate and

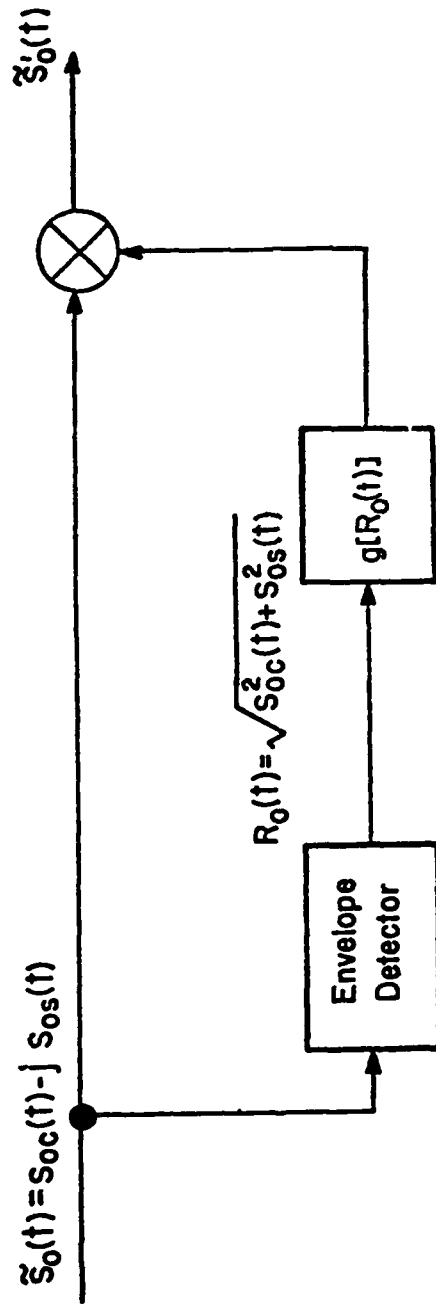


Figure 10
Model for Zero-Memory Nonlinearities of the Limiting Variety

eliminate the various synchronization loop tracking dynamics from consideration.

It should be noted that the various synchronization schemes are intimately related to the explicit modulation strategies employed. The various menu trees take this into account so that, for example, the user is not allowed the absurd choice of a carrier phase tracking loop when a noncoherent modulation strategy has been chosen. Furthermore, even for coherent systems the carrier tracking loop depends explicitly upon the nature of the modulation alphabet. For example, the typical squaring operation which precedes a carrier tracking or Costas loop in BPSK systems effectively eliminates the bi-phase modulation. It should be replaced by a M^{th} order nonlinearity in MPSK systems with appropriate provision for resolving the resulting phase ambiguity.

In addition to data demodulators corresponding to each of the modulation strategies included in the communications simulator, appropriate adaptive equalizer sub-modules are provided. More specifically, for the channel model described previously, the implicit diversity inherent in the multipath spread can be exploited only through adaptive equalization. That is, by resolution of the various multipath components it is possible to completely remove this source of intersymbol interference (ISI) and, more importantly, realize a performance which is an improvement over that obtained in the absence of multipath. The current trend in high-speed modem design for fading dispersive channel environments is to exploit this implicit diversity. As a result, an initial equalizer capability has been provided in the communications simulator.

At present there are three competing approaches to equalization of fading dispersive channels. The first makes use of the Viterbi algorithm (VA) to implement maximum likelihood sequence estimation on the basis of an assumed knowledge of the channel state. This scheme has been described by Forney [9] and Proakis [10] among others. The second approach is based upon decision

feedback equalization (DFE) and has been studied extensively beginning with the work by Monsen [11]-[13]. The basic DFE structure is illustrated in Fig. 11. The function of the transversal filter in the feedback path is to eliminate the ISI due to previously decided symbols or "postcursors" of the channels impulse response. Similarly, the function of the linear filter in the forward path is to eliminate ISI from future or previously undecided symbols, i.e., the "precursors" of the channels impulse response. The third, and final approach makes use of a tapped delay-line (TDL) equalizer of fixed length to equalize the effects of ISI.

The present version of the ICS includes options for choosing either of the three types of equalizer (VA, DFE, or TDL) together with the appropriate user protocol for choosing the corresponding equalizer parameters. At present, we assume knowledge of the channel filtering characteristics through a "cheater line" which informs the equalizer of the overall impulse response of the channel, including any narrowband filtering used in either the modulator/transmitter or the demodulator/receiver. A possibility for future enhancements is to provide an adaptive capability for determining and tracking the instantaneous state.

Channel Decoder

The channel decoder will accept the sequence $\{r_i\}$ appearing at the output of the demodulator/receiver and through appropriate decoding algorithms will produce the output sequence $\{\hat{a}_i\}$ delivered to the remote destination. Here we have provided several decoding options depending upon whether block or convolutional codes are used. In the latter case we employ the VA (cf. [14], [15]) for decoding arbitrary convolutional codes with constraint lengths in the range $3 \leq K \leq 10$. For block codes we have implemented several hard-decision decoders using essentially table look-up techniques. This has included the Hamming (8,4) and the Golay (24,12) codes which are typical choices in a wide variety of

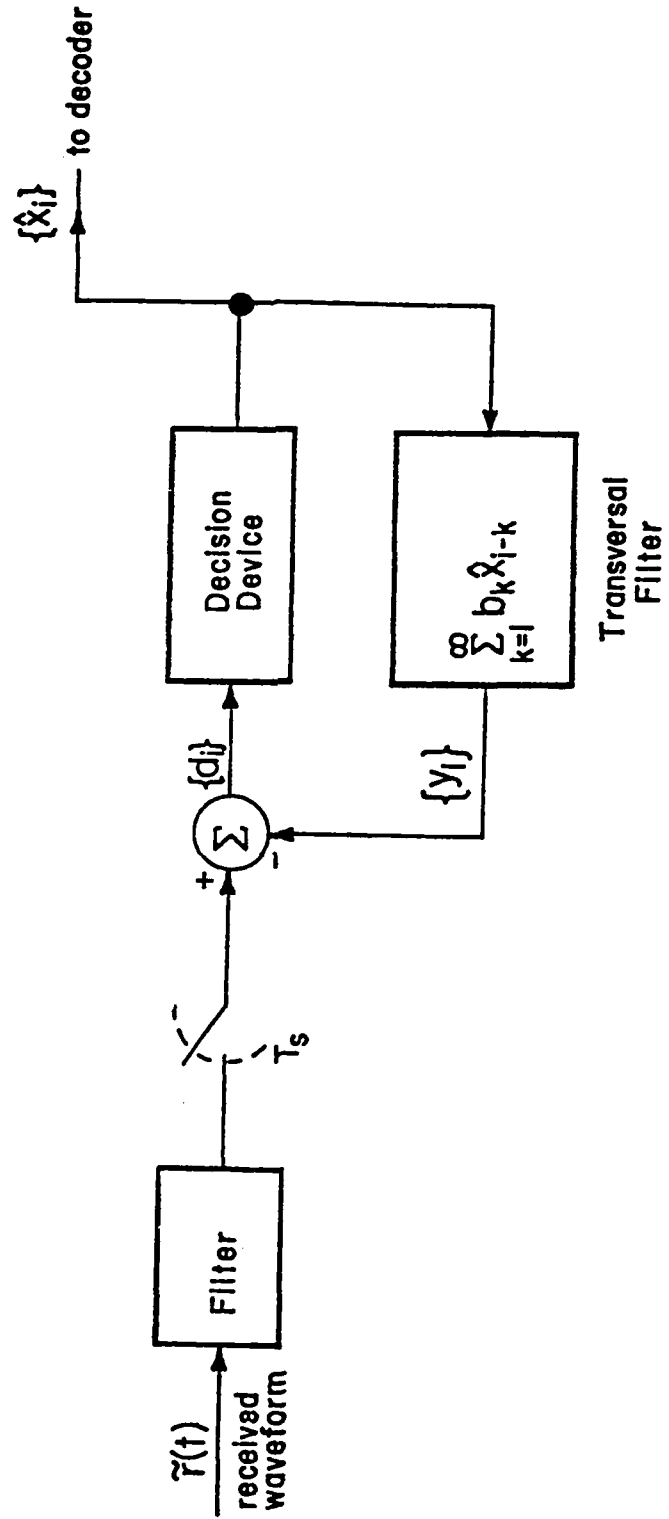


Figure 11
 Basic Decision Feedback Equalizer (DFE) Structure

applications. In addition, we have provided a fairly general soft-decision decoding capability for arbitrary block codes using the Wolf algorithm [16], [20] which resembles in many ways the VA. Use of this decoding capability requires explicit knowledge of the code parity check matrix. For selected codes this information is stored in tables on the host PDP-11/40. The user then specifies a code (e.g., the Golay (24, 12) code) and the appropriate parity check matrix is loaded with the decoder program.

3.2 OPERATION OF THE ICS:

The overall technical design of the ICS has been described in some detail in the preceding section. In this section we describe the interactive use of this system. Careful consideration has been given to this issue to allow flexible and efficient usage. Our general philosophy is that simulation should be used as an adjunct and not as a substitute for analysis. The communications simulator has been developed consistent with this philosophy. While the basic approach is Monte-Carlo simulation, every effort has been made to incorporate analytical and graphical techniques where and when appropriate to enhance and extend the capabilities of this system.

The ICS has been designed to be used in four distinct modes of operation. In the first, which we call the DESIGN mode, the user assembles a simulation model in building-block fashion from basic function modules provided as part of the system. Appropriate user protocols have been provided to guide even the casual user through the DESIGN stage. Specifically, the block diagram of a general communications system illustrated in Fig. 2 is displayed on the graphics CRT. Using the light pen, the user selects successive blocks and a menu list appears describing each of the available options for that functional element. The user chooses from this list by entering the appropriate keyboard response eventually creating a DESIGN file representing an executable simulation model of a complete end-to-end communications system. This DESIGN file

is suitably named and entered into the system library. At this point the user can exit from the DESIGN mode or continue creating alternate executable DESIGN files. This is the most straightforward of the four usage modes and requires no further elaboration.

The second usage mode will be called the VALIDATION mode. Here the user will be allowed to interactively exercise a simulation model which has previously been developed in a DESIGN session. The purpose of the VALIDATION mode will be to allow the user to verify a simulation model before a commitment to time-consuming Monte-Carlo simulation. More specifically, the user will be allowed to view time waveforms and/or frequency domain quantities such as power spectral densities, system transfer functions, etc., at critical points in the system to be simulated. This should enable the user to verify the end-to-end behavior of the simulation model for the communication system under study or the input/output behavior of any specific sub-module. The user, situated at the graphics terminal, can by either light pen or keyboard entry view time or frequency domain quantities as if a scopeprobe was connected to the corresponding point in an actual communications system. We refer to this ability as the SCOPEPROBE concept. This capability of the communication simulator should not be considered as merely a procedural verification stage preceding an actual simulation. Rather, we feel strongly that the VALIDATION mode can itself be employed as a highly flexible and useful tool in conceptual communication system studies. That is, by employing the VALIDATION mode a useful and realistic appreciation of actual system behavior can be developed. Such an appreciation can often serve as a useful adjunct to detailed analysis. For example, the effects of various filtering operations can be observed directly, degradations due to intersymbol interference or multipath can be assessed visually, etc. Hard-copy output of graphics can be obtained using the Varian

4211 printer/plotter. Some typical graphical outputs are provided in the next section.

In order to modify or otherwise change a simulation model, the user must leave the VALIDATION stage and re-enter the DESIGN mode. Appropriate editing capabilities have been provided to allow flexible modification of previously developed simulation models. At any rate, the user is at some point ready to commit a simulation model to extensive Monte-Carlo simulation. It is at this point that the user enters the third stage which we call the SIMULATION mode. In this mode, an executable DESIGN file corresponding to a simulation model must be identified together with appropriate simulation parameters. Primary interest is in the evaluation of bit error probability P_b as a function of the quantity E_b/N_0 . Here E_b represents the signal energy per bit while $N_0/2$ represents the double-sided noise spectral density in watts/Hz. As a result, the important parameters to be specified are both an initial and final value of E_b/N_0 for which simulation results are desired as well as an increment in E_b/N_0 . In addition, parameters such as the number of errors to be collected for each value of E_b/N_0 and the maximum number of iterations must be specified. These latter quantities are necessary in order to insure statistical validity of the results. The communications simulator will then loop through the specified range of E_b/N_0 . This is the only parameter for which looped operation is provided in the SIMULATION mode. For other parameters it is required that an appropriate DESIGN file representing a simulation model with specific parameter choices has been identified for subsequent execution.

In the SIMULATION mode, appropriate data logging and bookkeeping software is provided to allow tabulation of bit error histories and evaluation of bit error probabilities as a function of E_b/N_0 . This software is considered an important part of the communications simulator. In addition to straightforward

evaluation of cumulative bit error counts, other basic statistical outputs are provided. For example, histograms of the interarrival times between bit errors, the average duration of error bursts, etc. We feel that many of these other statistics derived from bit error histories are useful in some applications for providing a more complete picture of overall communication system performance. Finally, statistical files are created for generating histograms of other parameters during a SIMULATION session. This includes histograms of the additive channel noise (both amplitude and phase), and both the phase and bit synchronization errors.

The last and final mode of operation is called the ANALYSIS mode. This mode provides two basic functions. The primary function is to provide the graphical display interpretation of results obtained during a typical SIMULATION session. Appropriate display drivers and associated software have been provided to allow results to be displayed in a format to allow easy interpretation by communication systems engineers. For example, a basic requirement is to display curves representing simulated bit error probability P_b as a function of E_b/N_0 . In those cases where closed form analytical expressions or bounds on P_b vs. E_b/N_0 are available, these are included as user selected option. The user protocol in the ANALYSIS mode will provide guidance to the user concerning any such facilities available for the system configuration and/or parameter choices under study. For example, fairly tight upper bounds are available [14], [17] on P_b vs. E_b/N_0 for convolutional codes in conjunction with Viterbi decoding and employing a variety of modulation formats on the AWGN channel. Similar bounds have been derived [18] for soft-decision decoding of block codes employing the Wolf algorithm. These bounds, for the most part, neglect the effects of narrowband filtering, imperfect phase and/or symbol synchronization,

intersymbol interference, etc. Nevertheless, the readily computed upper bounds are useful in providing a context or perspective in which to assess the degradations due to these effects. For this reason, the ability to compute and display these idealized performance bounds together with results obtained by simulation have been incorporated as a useful feature of the ICS.

In the course of developing the ICS, we have found it useful to access the computational and display capabilities associated with the ANALYSIS mode on a stand-alone basis. For example, the ability to compute and plot upper bounds, and in some cases, actual theoretical performance of selected communication systems, has proven useful in a number of related system studies. As a result, the ANALYSIS mode has been designed to allow dual entry thus providing a secondary function as a stand-alone computational and display resource. The library associated with this mode can be easily updated to provide an expanding analysis tool quite independent of the simulation capabilities associated with the overall system.

A block diagram illustrating the overall software organization of the communications simulator is illustrated in Fig. 12. An optional introduction to the system is provided in the user protocol associated with the MAIN DRIVER program.

3.3 TYPICAL GRAPHICAL OUTPUT:

In this section we describe some of the hard-copy graphical output obtained with the ICS. We consider first the time-domain waveforms obtainable in the VALIDATION mode. Limited frequency-domain graphical display and/or output capabilities are available and will be described below. Virtually any time waveform can be observed individually or in combination with other related waveforms. The user must first specify a time or viewing window consisting of a fixed number of baud. A segment of a waveform is then displayed within the viewing window and the contents can be indexed through a

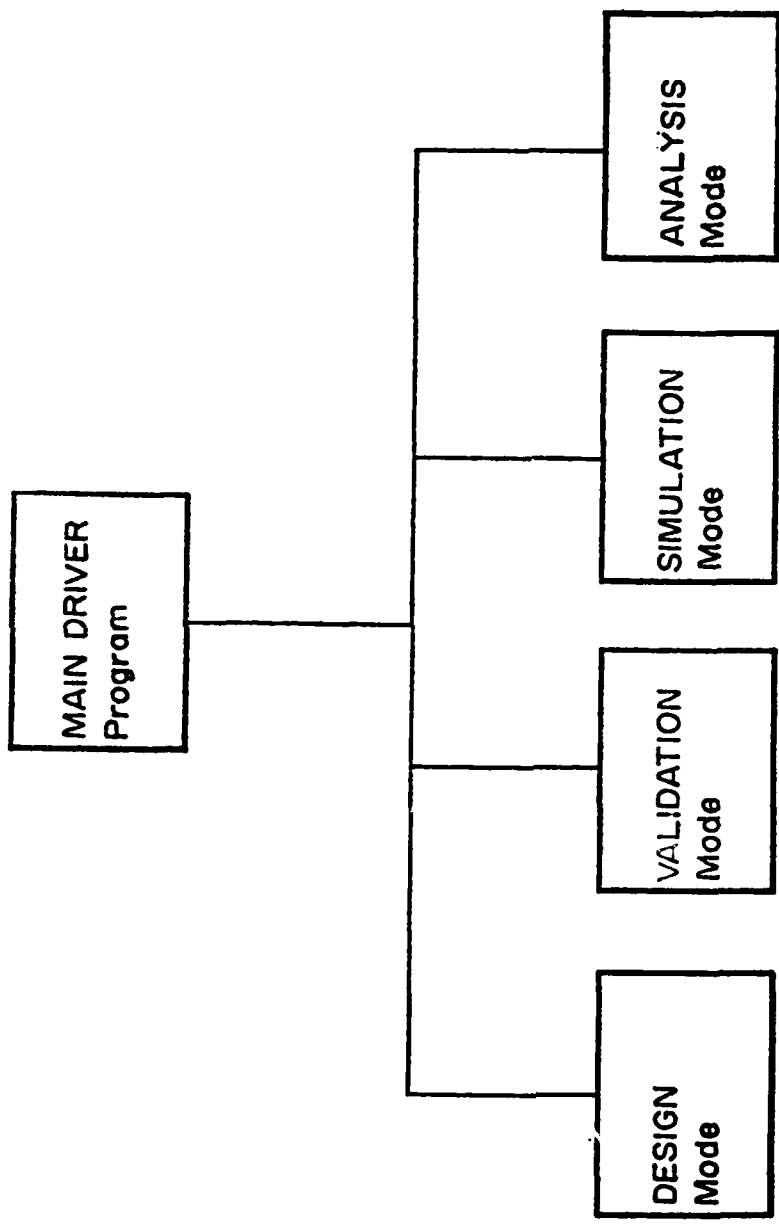


Figure 12
Software Organization of Interactive
Communications Simulator

waveform file in either fixed increments or continuously scrolled through a large file. The scrolling can be stopped and restarted under keyboard control.

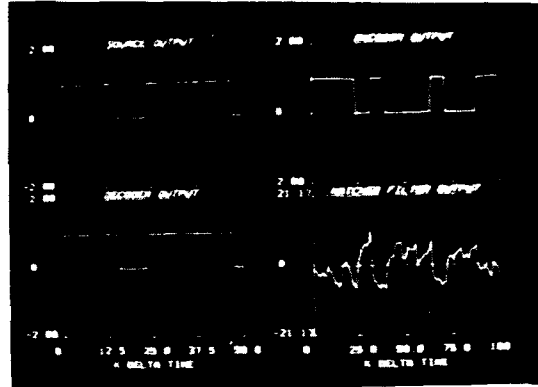
We have found it convenient to classify time-domain waveforms into three broad categories:

1. BASEBAND Waveforms
2. CHANNEL Waveforms
3. SYNCHRONIZATION Waveforms

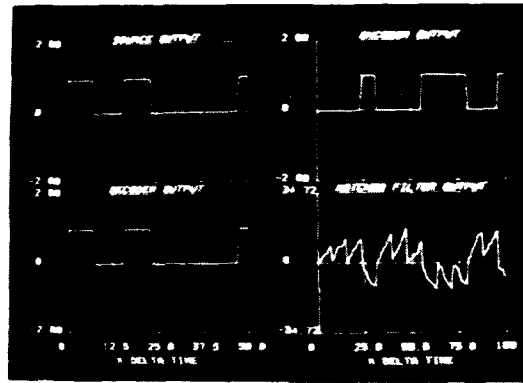
The BASEBAND Waveforms include all time-domain quantities which allow lowpass signal representations and are in some sense synchronous with the baud rate. This would include source outputs, source/channel encoder and decoder outputs, and matched filter outputs sampled at the baud rate. By comparison, CHANNEL waveforms include all time-domain waveforms which require complex representations, either in polar coordinates or in terms of I/Q components. Actually, the user protocol allows the user to specify either representation. Frequency-domain plotting and display capabilities are available for any CHANNEL waveform. The frequency-domain representation is computed for the observed waveforms appearing within the viewing window and using a wide variety of possible window functions. Detailed information on the various window options is provided in the user protocol. Again, the frequency-domain representation can be displayed in either polar or rectangular (i.e. I/Q) format under user control. Finally, the last category of SYNCHRONIZATION waveforms include all lowpass waveforms, generally not synchronous with the baud rate, which can be used to describe the various adaptive tracking loops. This would include instantaneous phase and/or bit timing errors, as well as typical error signals appearing in adaptive equalizer loops. The graphics driver programs and logic are decidedly different for each of these three categories of waveforms.

Some typical BASEBAND signaling waveforms are illustrated in Figs. 13a and 13b for a $K=6$, $R=1/2$ convolutionally encoded BPSK system on the AWGN channel with $E_b/N_0=3\text{dB}$ and 10dB respectively. In this case the observation window consists of approximately 12 baud and we have made use of a MULTIPLE WAVEFORM plotting option. More specifically, while the user can look at any individual BASEBAND waveform in the SINGLE WAVEFORM plotting and display mode, the ICS provides the capability of plotting certain logical combinations in the MULTIPLE WAVEFORM plotting option. These figures illustrate the source output and Viterbi decoder output in time alignment in the left-hand column while the encoder output and matched filter output are in time alignment along the righthand column. This is a logical arrangement of multiple BASEBAND waveforms and is provided to the user as a display option. Notice the raw channel symbol errors that are made at low E_b/N_0 in making hard decisions on the polarity of the noisy matched filter output. The substantial error control coding in this case is impervious to these errors.

Typical CHANNEL signaling waveforms are illustrated in Fig. 14. The waveforms in Fig. 14a correspond to a coherent BPSK system operating on the AWGN channel at the relatively large value $E_b/N_0 = 20\text{dB}$, while Fig. 14b represents a noncoherent BPSK system operating under identical channel conditions. Here we again make use of a MULTIPLE WAVEFORM plotting capability with the left-hand column associated with the modulator/transmitter output $s(t)$ while the right-hand column is associated with the channel output $r(t)$. Furthermore, the CHANNEL waveforms in this case are plotted in polar coordinates in terms of envelope and phase. Similar plots in rectangular coordinates using I/Q components could have been chosen under user control. Likewise, individual CHANNEL waveforms could have been selected under the SINGLE WAVEFORM plotting and display option. In this case, frequency-domain representations could be computed and displayed.



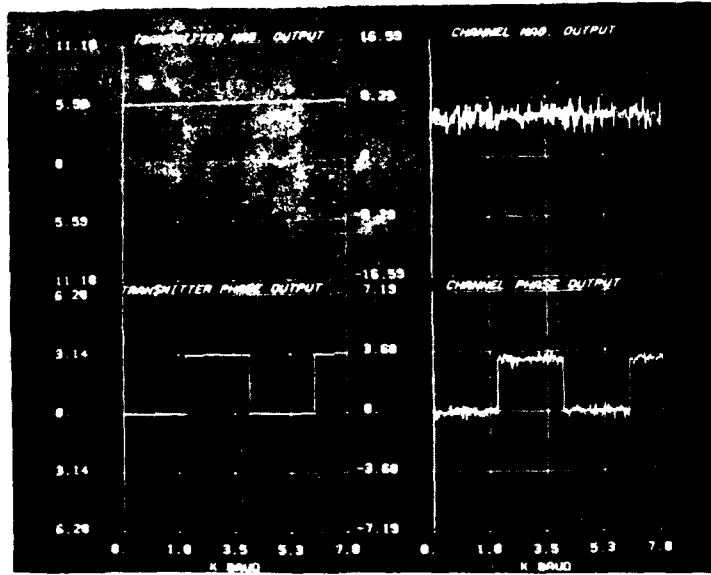
a.) $E_b/N_0 = 3dB$



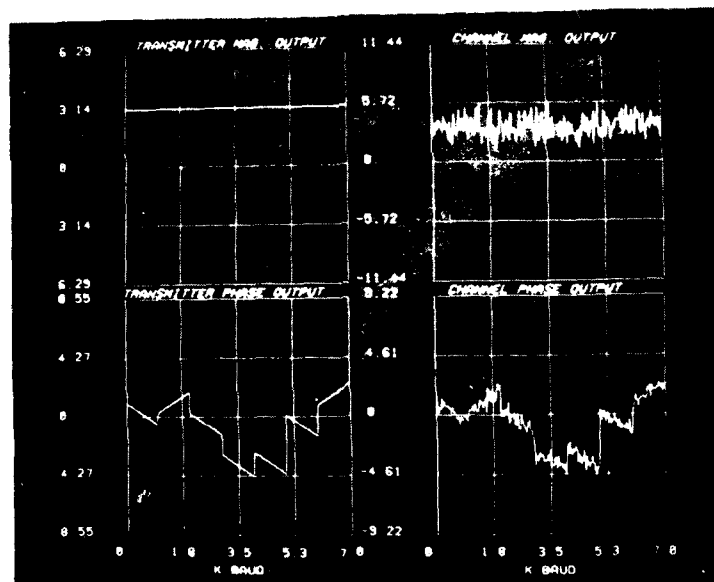
b.) $E_b/N_0 = 10dB$

Figure 13

Typical Baseband Signaling Waveforms
for BPSK Modulation on
AWGN Channel Employing
K=6, R=1/2 Convolutional Code



a.) Coherent BPSK, $E_b/N_0 = 20\text{dB}$



b.) Noncoherent BPSK, $E_b/N_0 = 20\text{dB}$

Figure 14
Typical Channel Signaling Waveforms

Finally, as described above, the last category of time-domain signals are SYNCHRONIZATION waveforms. Typical phase and bit timing errors are illustrated in Fig. 15 for a coherent BPSK system. While these waveforms are plotted jointly in time alignment, the user has the option of plotting either waveform individually under the SINGLE WAVEFORM plotting option. Similarly, plots of the error signal in the tracking loop of an adaptive equalizer can be displayed in this category of time-domain waveforms.

A number of additional graphical plotting and display capabilities exist in the ANALYSIS mode. For the most part, these consist of accessing, formatting, and displaying error statistics or noise amplitude probability distributions (APD's) generated in executing an appropriate DESIGN file in the SIMULATION mode. For example, a DESIGN file, representing a coherent BPSK system operating on the AWGN channel and employing a $K=6$, $R=1/2$ convolutional code, was executed in the SIMULATION mode and both raw and decoded error statistics collected. In Fig. 16 we illustrate the measured symbol error probability P_e and the decoded bit error probability P_b . The decoded bit error probability is plotted as a function of E_b/N_0 while the symbol error probability is plotted as a function of the quantity E_s/N_0 . Here, E_s is the signal energy on a per baud basis and is related to E_b according to $E_s = RE_b$ where R is the code rate (in this case $R=1/2$). Also included in the respective plots in Fig. 16 is the theoretical symbol error probability P_e^\dagger and the computed upper bound on P_b^\dagger . The simulation results compare favorably to these computed results. These results provide a complete picture of the performance of the coded coherent BPSK system on the AWGN channel. The total elapsed

† Simulation results are plotted with crossed while computed results employ open circles. The numbers along the left-hand margin of these plots represents powers of 10.

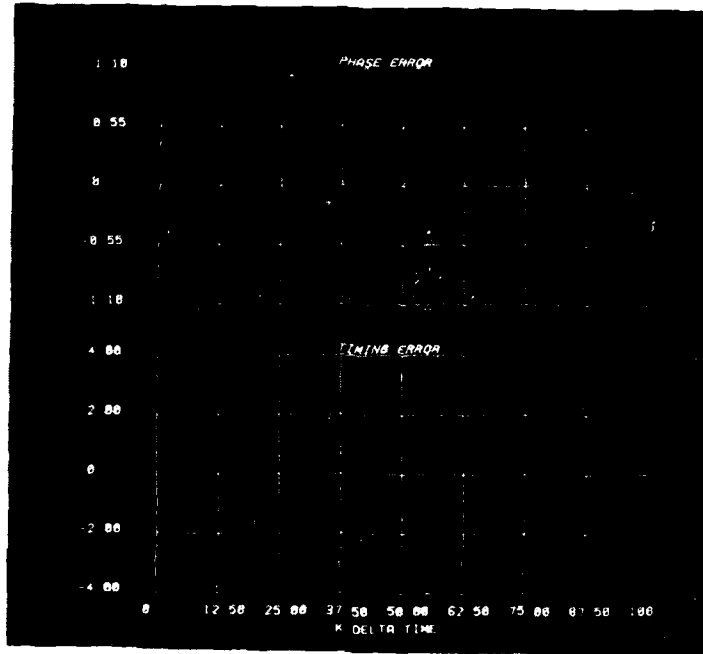
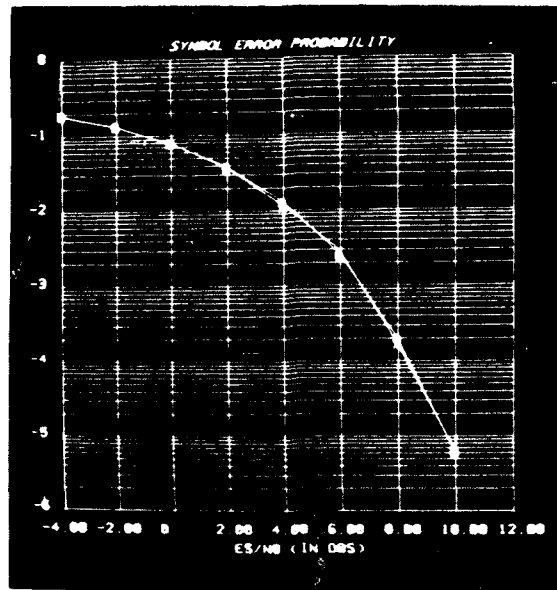
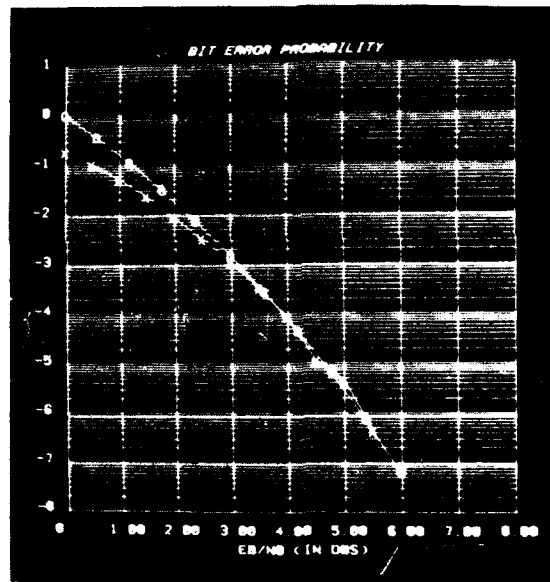


Figure 15

Phase and Bit Synchronization Errors
for Coherent BPSK System, $E_b/N_0 = 20\text{dB}$



a.) Symbol Error Probability P_e



b.) Bit Error Probability P_b

Figure 16

Plots of Symbol and Bit Error Probability
for BPSK System Operating on AWGN
Channel and Employing K=6, R=1/2
Convolutional Code with Viterbi Decoding

time to execute the SIMULATION mode in this case over the range $0\text{dB} < \underline{E}_b / \underline{N}_0 < 6\text{dB}$ in steps of 0.5dB was 2 hrs. This is significant when one observes that error events as rare as one in 10^7 have been accurately measured. Comparable simulation capabilities would be impossible in existing general-purpose machines. Indeed, it is questionable whether or not such an extensive simulation would be undertaken in the first place.

Other graphical output available in this case would include: histograms of the additive channel noise amplitude and phase, two-dimensional histograms of the phase and time synchronization errors, and various statistics derived from bit error histories as described earlier. In the interests of space, these will not be described here.

4.0 SOFTWARE DESCRIPTION:

The purpose of this section is to describe the basic structure of the software implementing the SIMULATION and VALIDATION modes of the ICS. It is in these modes that data is generated, processed, and results tabulated.

In the VALIDATION mode, the ICS performs a relatively short simulation of a point-to-point communication system which has been previously configured in the DESIGN mode. The number of baud processed in this case is typically in the range 100-500. Waveform data is extracted from the AP-120B and stored on the PDP-11/40 disc for later display and/or hard-copy output in the ANALYSIS mode.

In the SIMULATION mode, the ICS performs an extended Monte-Carlo simulation of the configured system represented by an executable DESIGN file. Extensive error statistics are tabulated during this stage for later access in the ANALYSIS mode.

4.1 FORTTRAN DRIVER PROGRAM:

Each of the communication system blocks is implemented by an AP-120B assembly language module. All data generation and processing is performed in the AP-120B. Once a complete system has been designed by the user, it is the job of the driver program to select, load, and run the various AP program modules corresponding to the various system components. The same driver is used for both the SIMULATION and VALIDATION modes. The only difference is the amount and type of data which is collected.

Upon entering the VALIDATION mode, the user specifies the length of the simulation and at which points in the system he would like to collect waveform data. When the VALIDATION run actually begins, the driver program will access the DESIGN file to determine which program modules to load along with their associated simulation parameters. After this, the entire design file is transferred to the AP-120B so that any of the DESIGN file information may be accessed by an executing AP program. After the AP has been loaded, the program modules are exercised sequentially with waveform data being stored on the PDP-11/40 disc.

Operation of the driver during the SIMULATION stage is identical except for the fact that the system now may perform a number of simulations for different values of signal-to-noise ratio. During this mode, error statistics rather than waveform data are stored on disc. The data flows during the operation of the simulator are diagrammed in Fig. 17. The overall logic flow of the driver program is shown in Fig. 18.

4.2 DATA STRUCTURES:

It is difficult to simulate a continuous data stream inside a processor with limited memory, since only a small block of data may be processed at one time. There are basically two schemes for facilitating this.

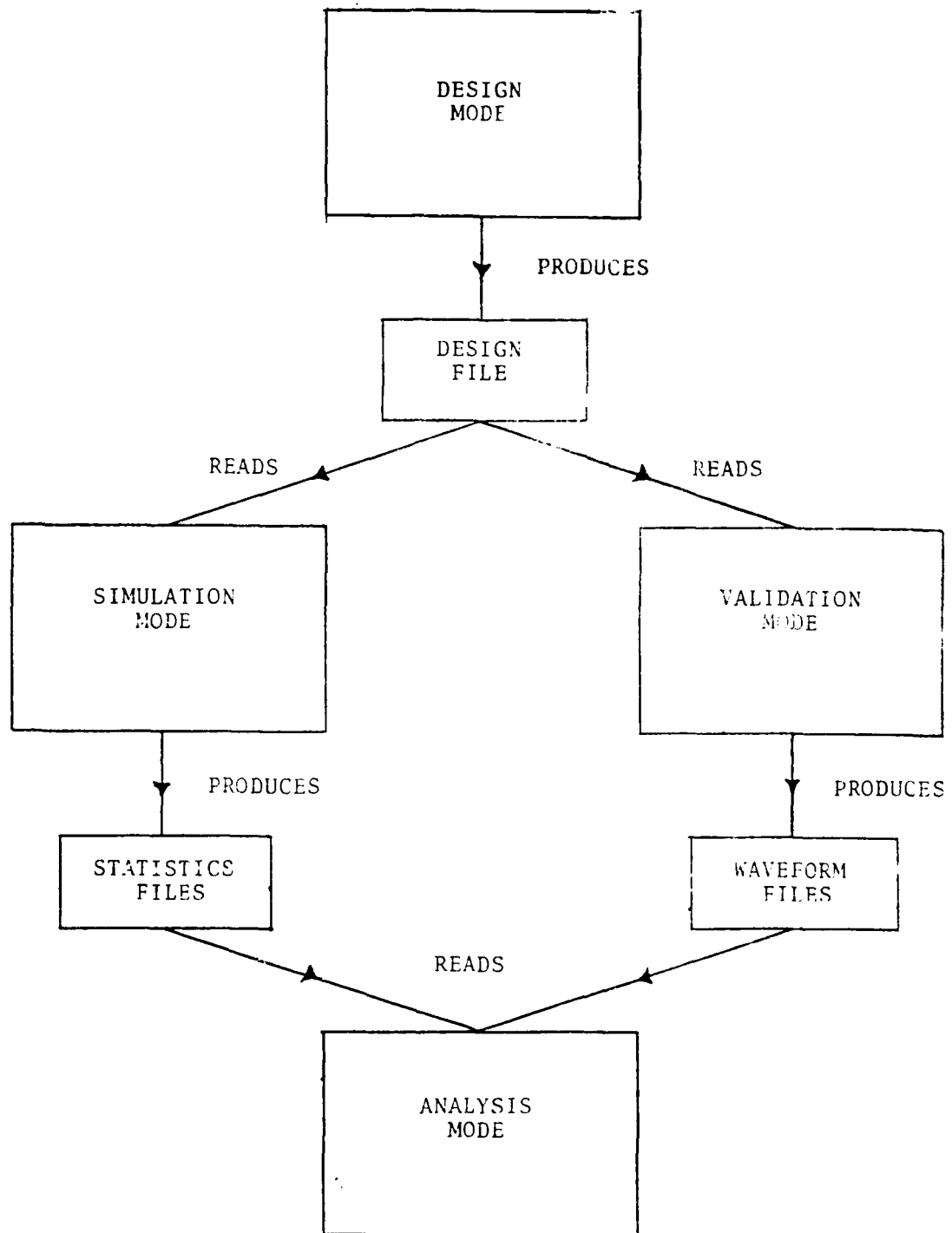


Figure 17
Data Flows in Operation of the ICS

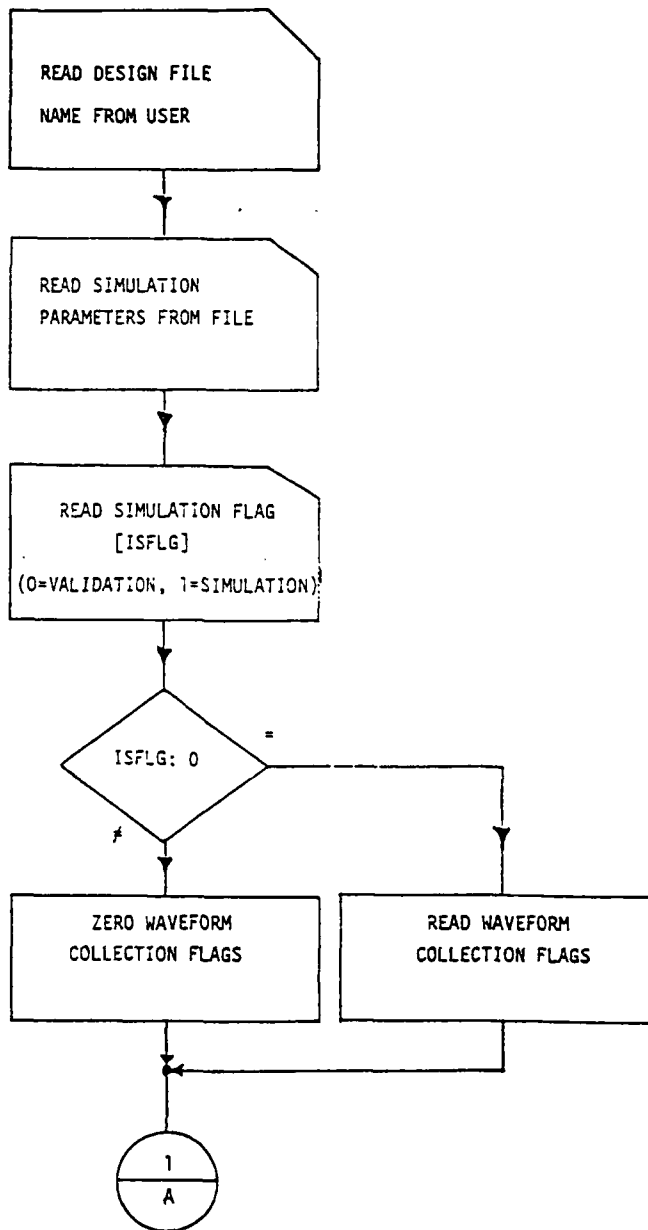


Figure 18.1
Overall Logic Flow of Driver Program

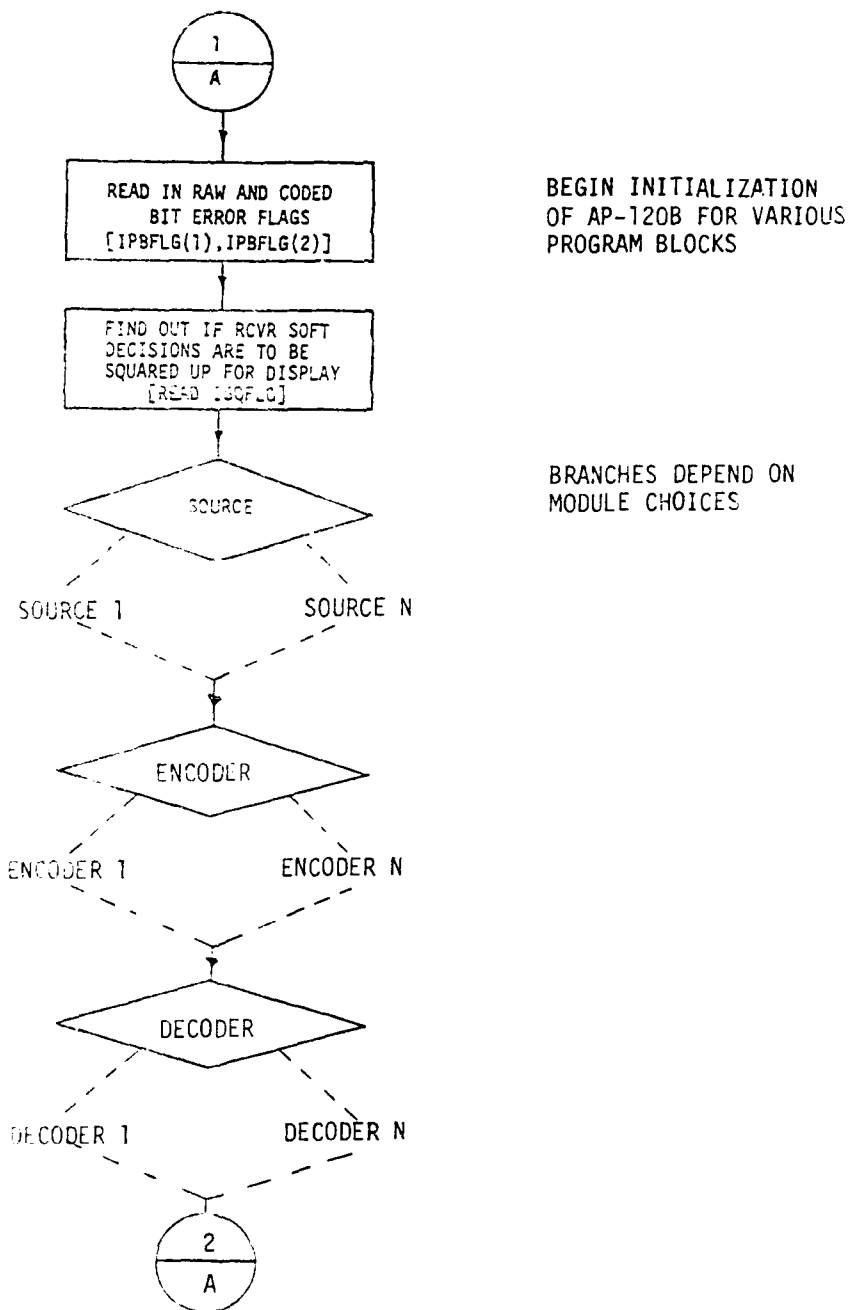


Figure 18.2
Overall Logic Flow of Driver Program

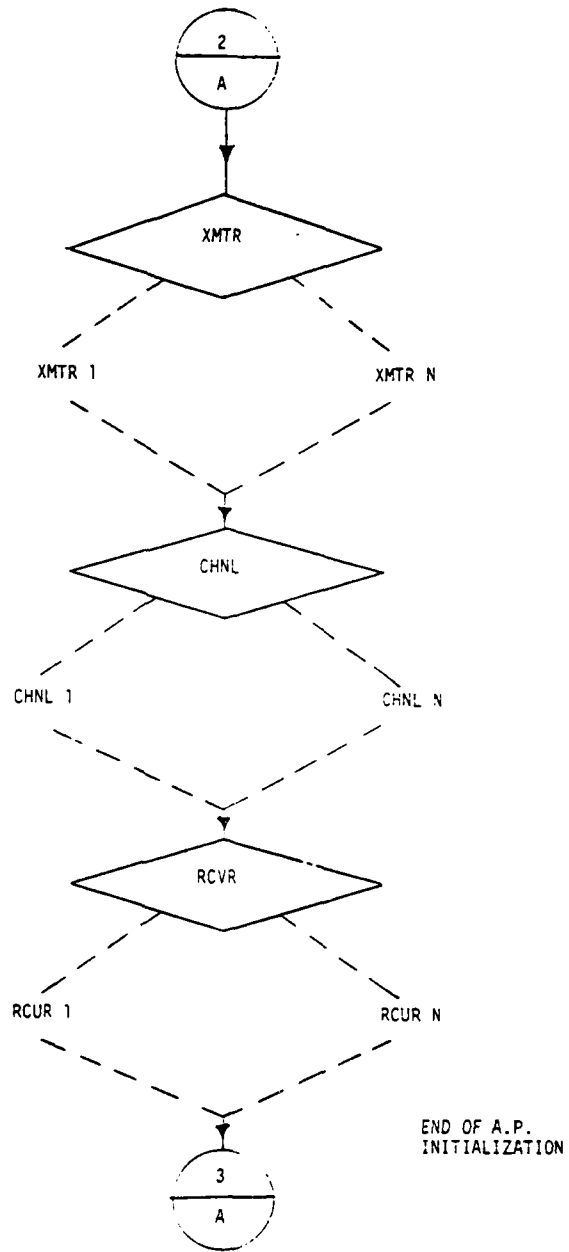


Figure 18.3
Overall Logic Flow of Driver Program

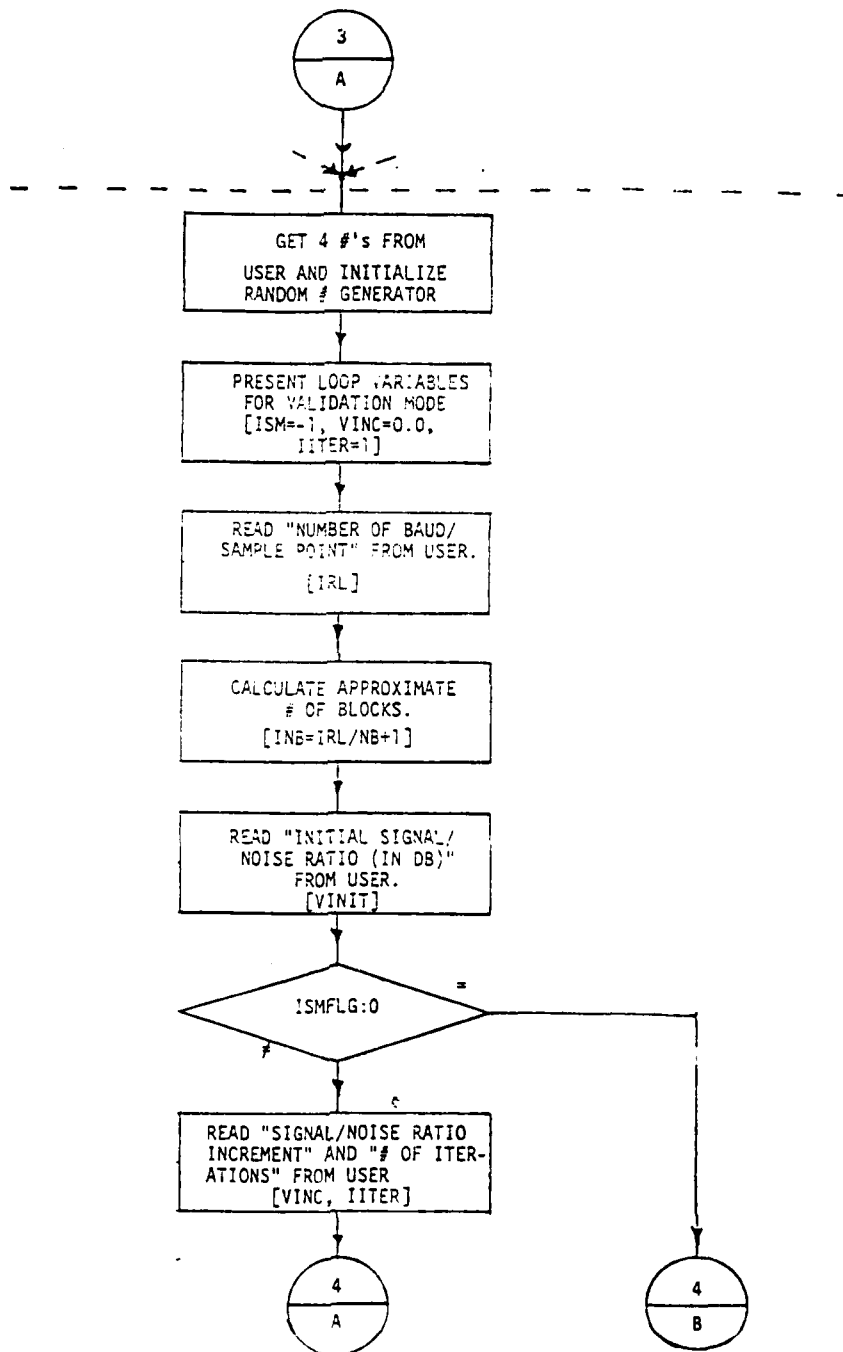


Figure 18.4
Overall Logic Flow of Driver Program

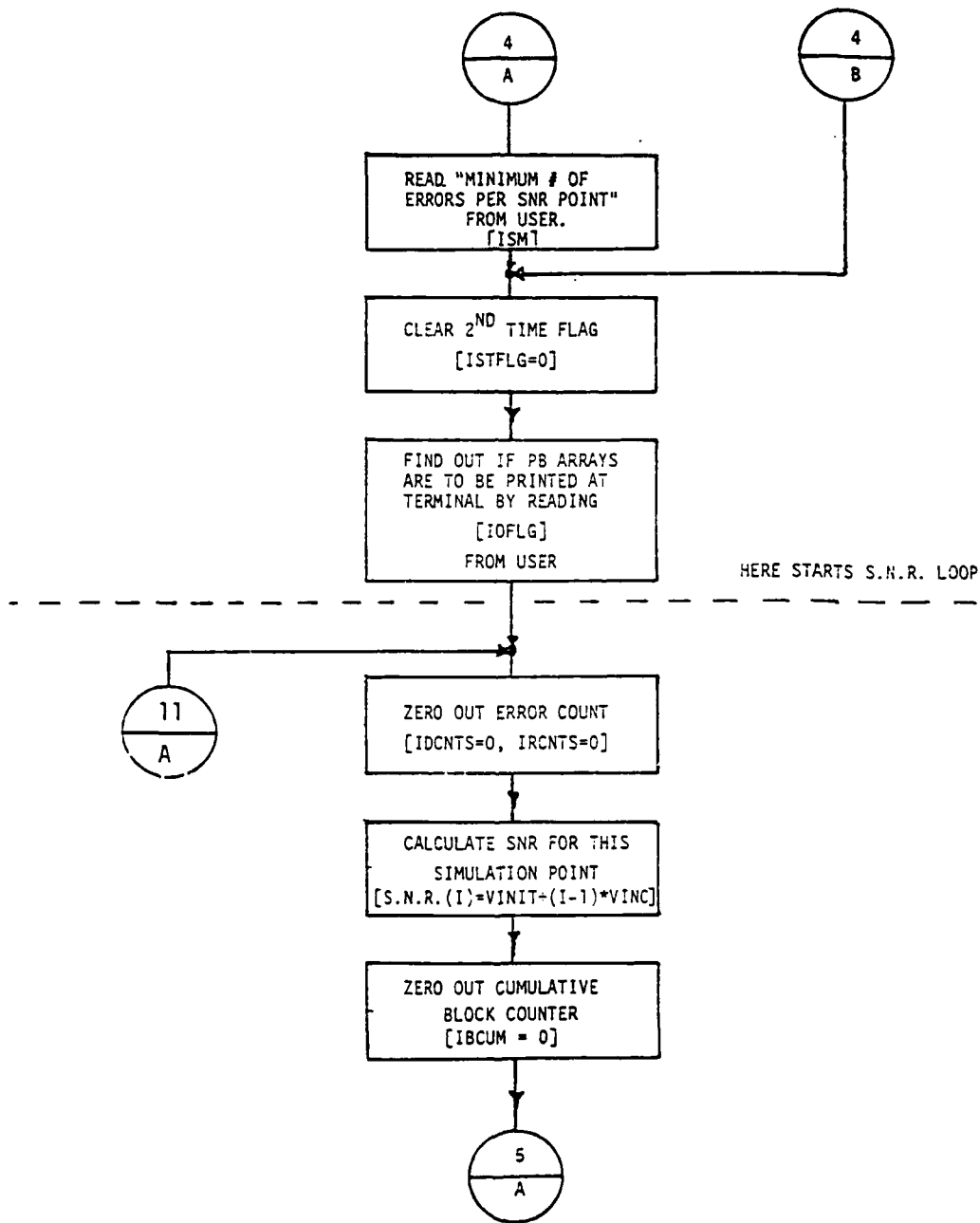


Figure 18.5
Overall Logic Flow of Driver Program

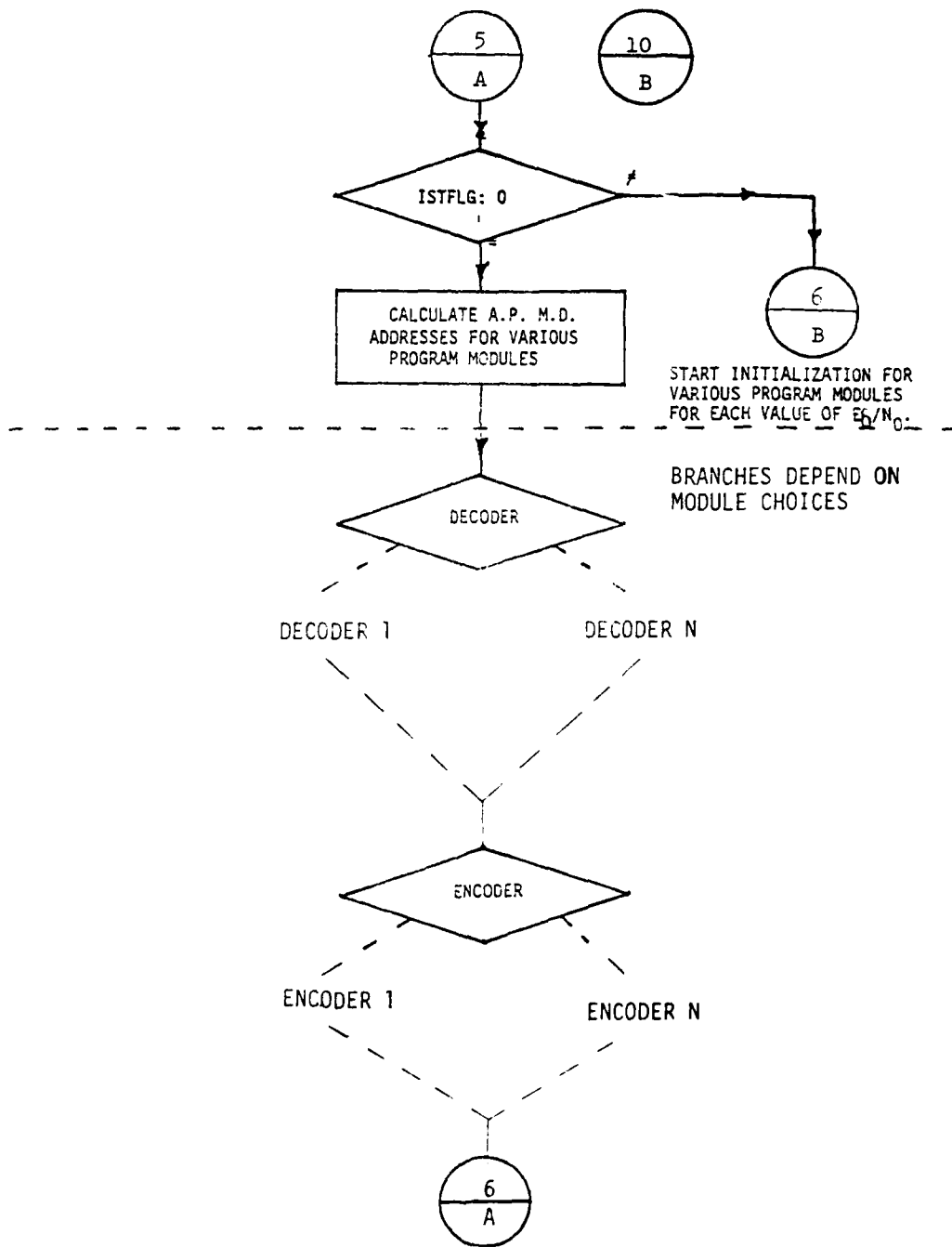


Figure 18.6
Overall Logic Flow of Driver Program

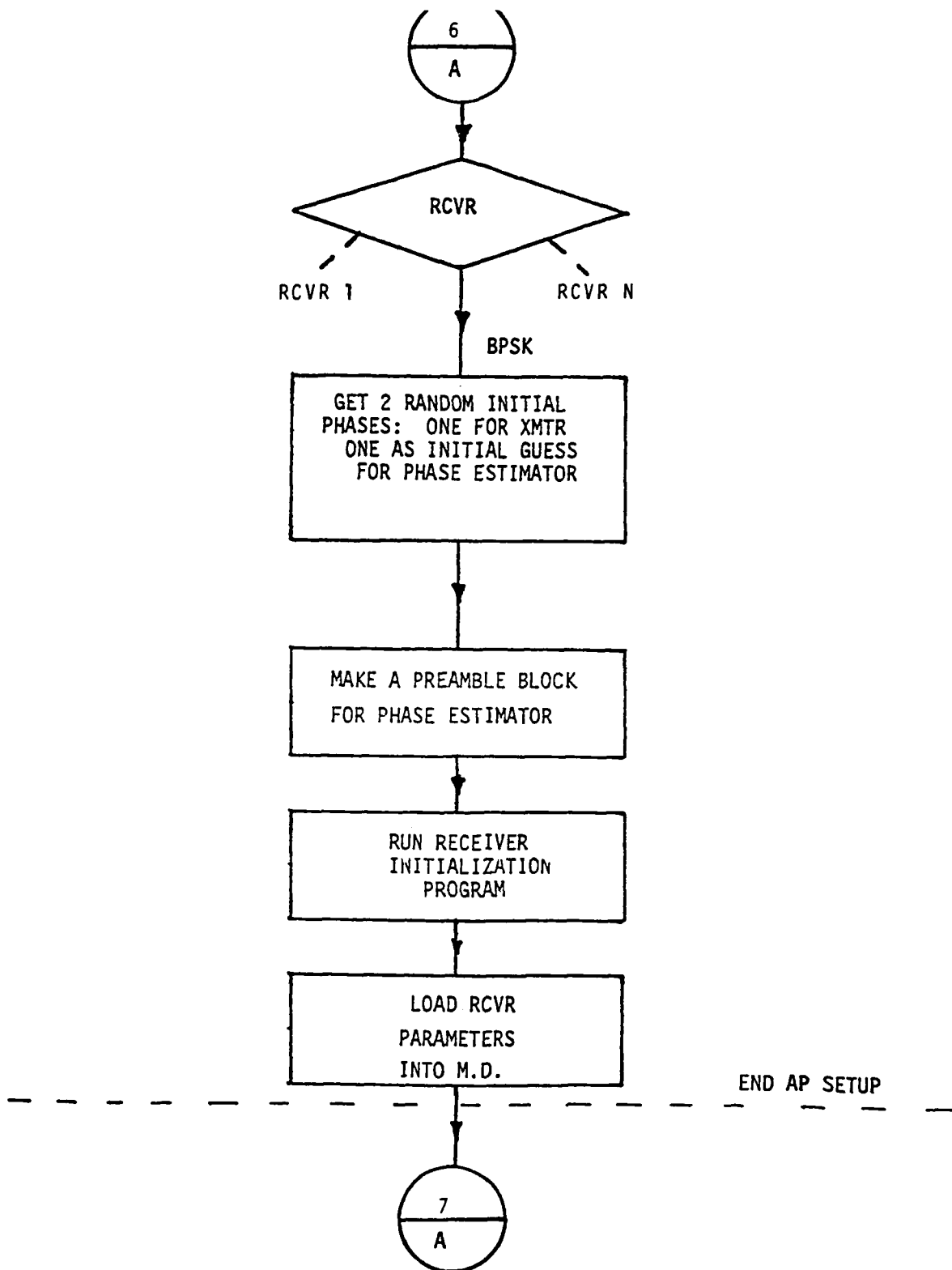


Figure 18.7
Overall Logic Flow of Driver Program

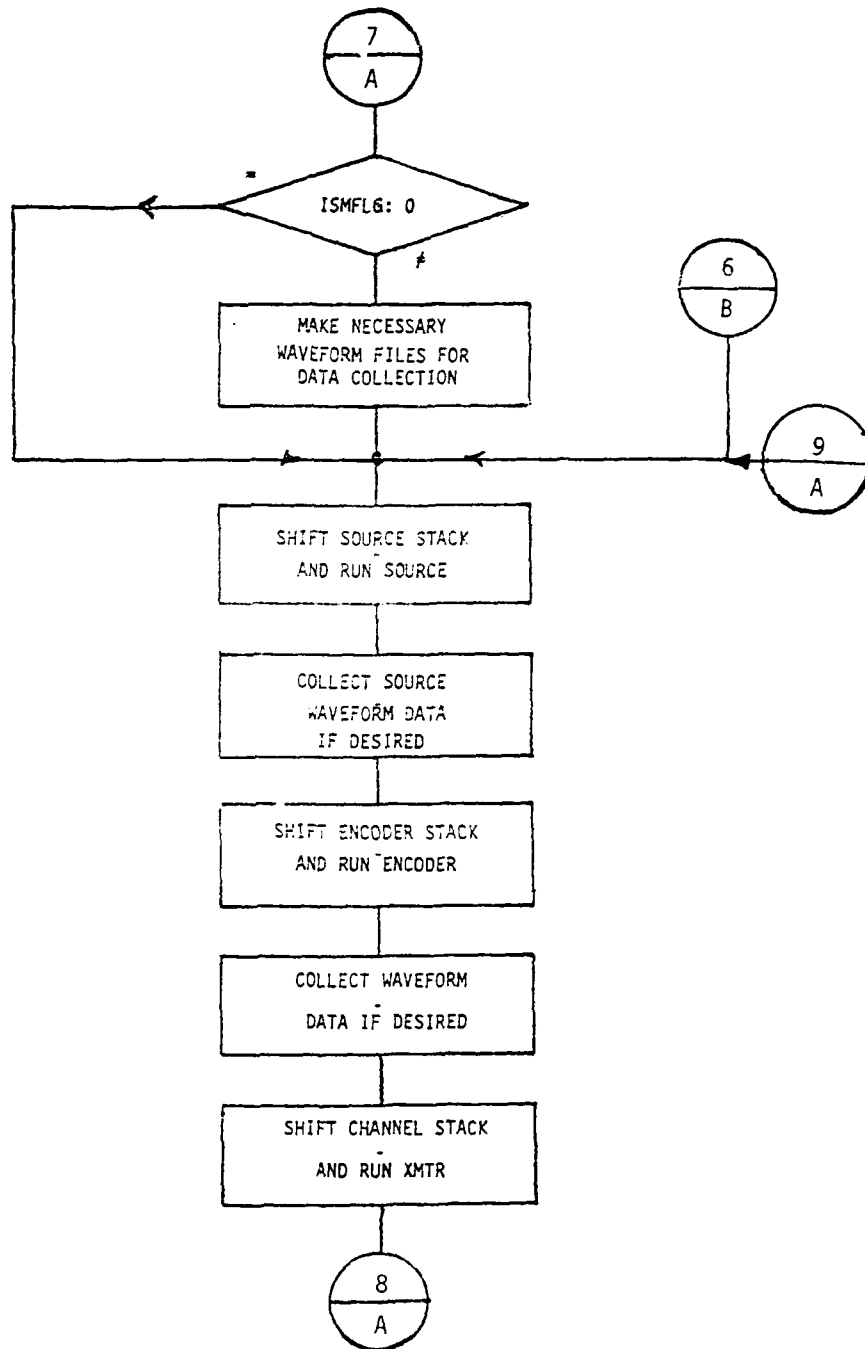


Figure 18.8
Overall Logic Flow of Driver Program

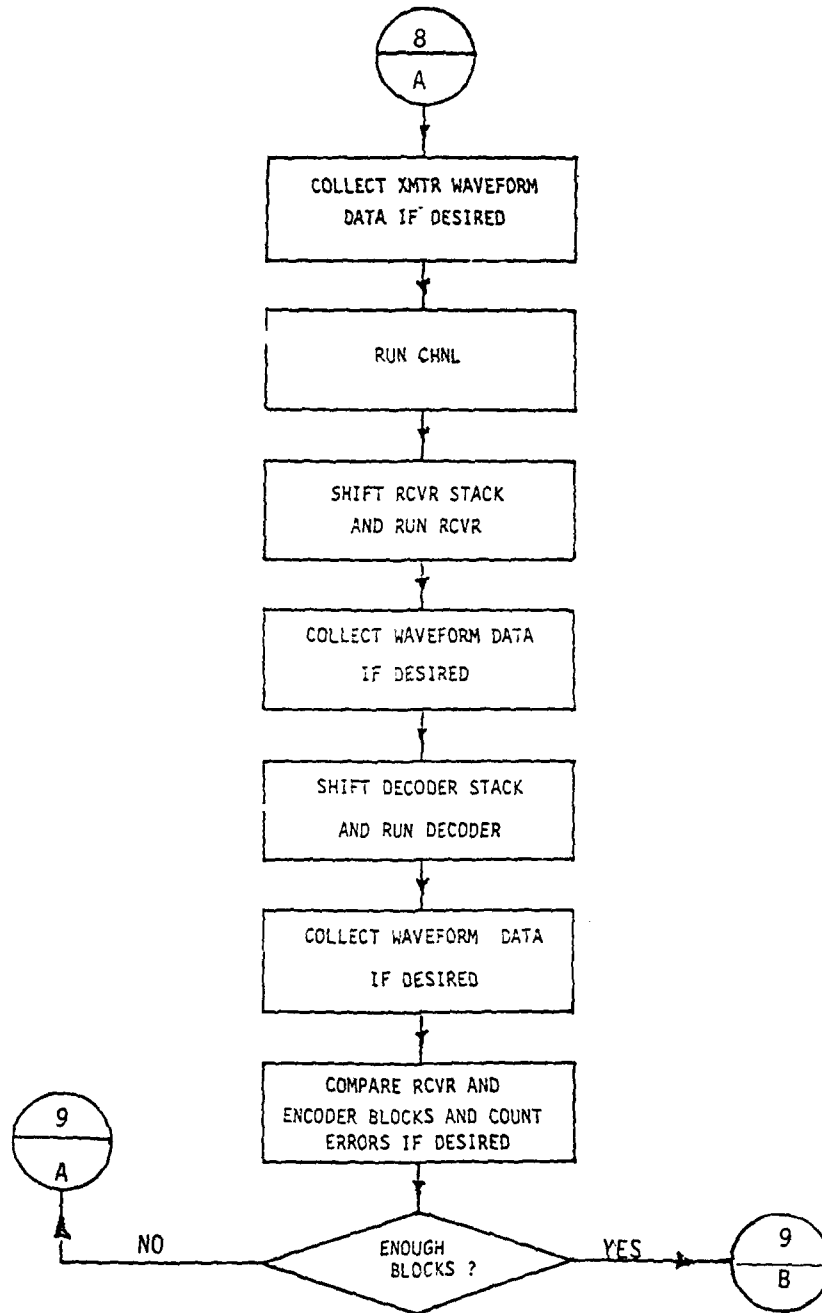


Figure 18.9
Overall Logic Flow of Drive Program

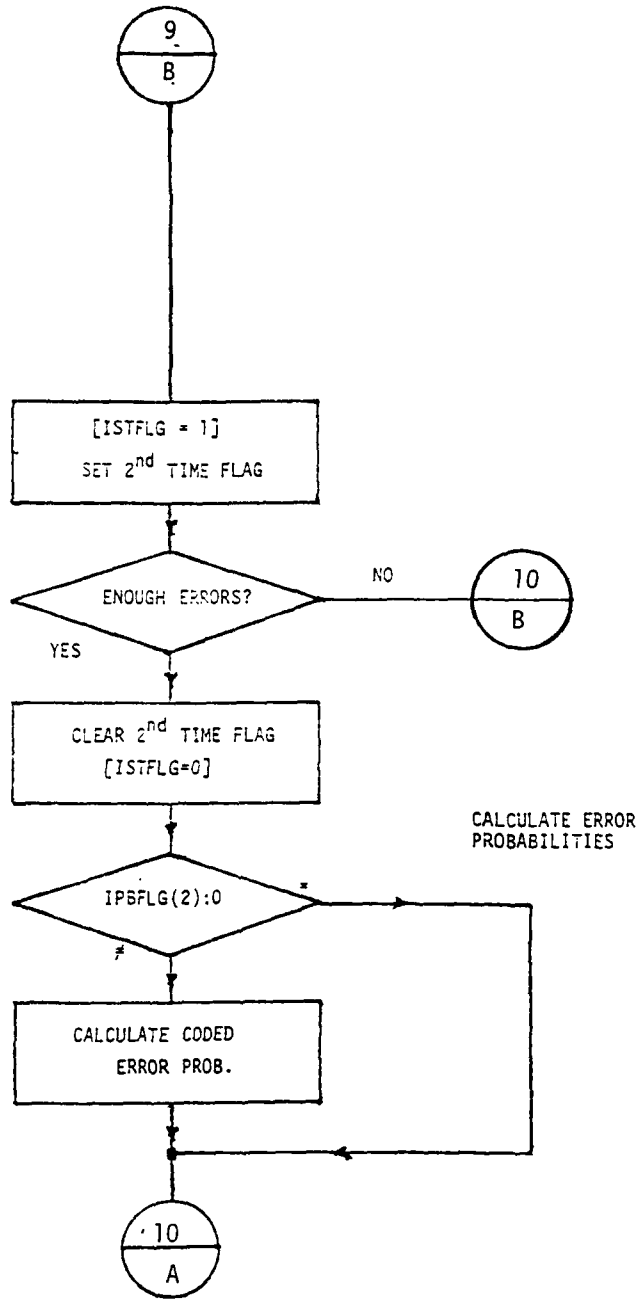


Figure 18.10
Overall Logic Flow of Driver Program

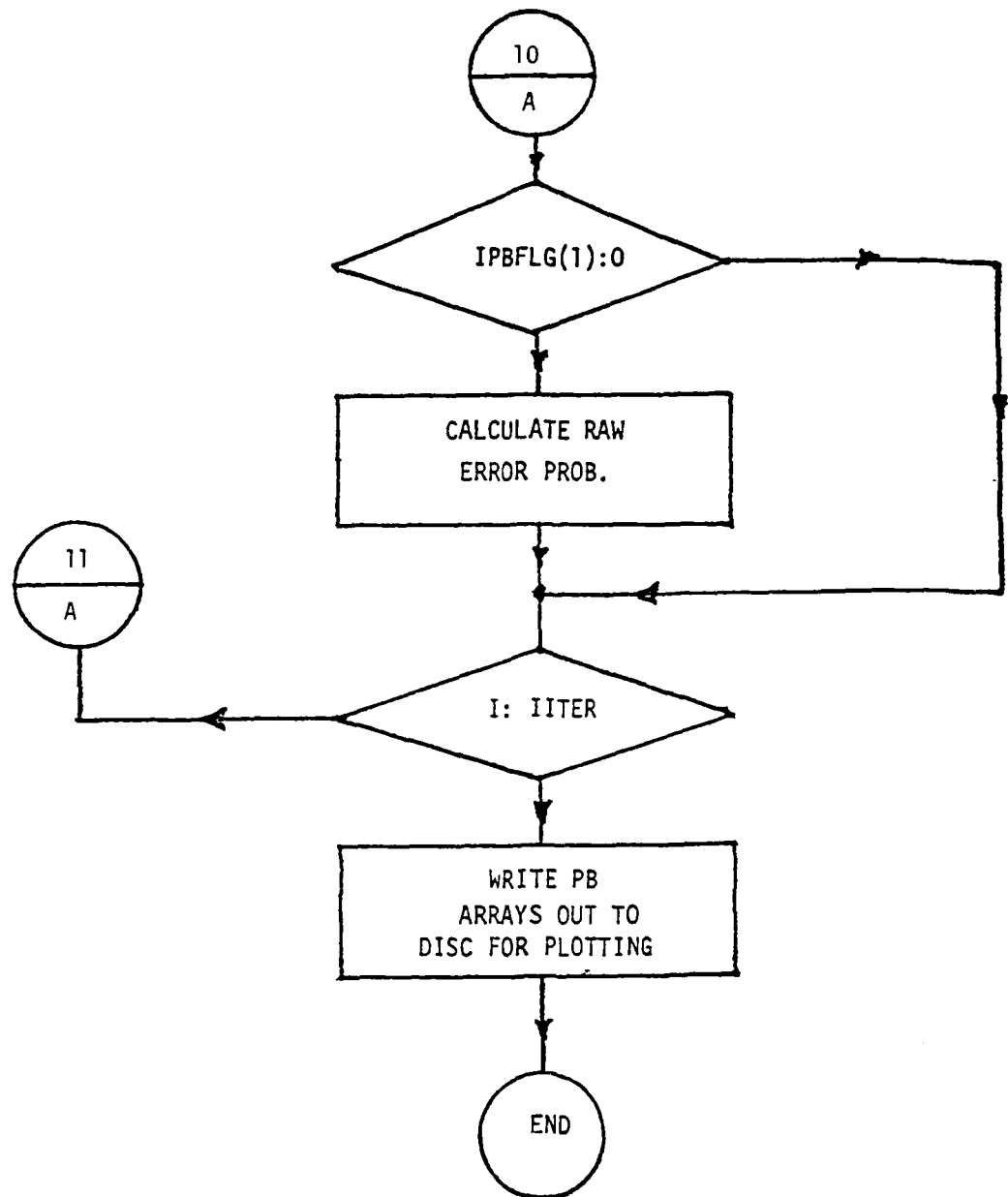


Figure 18.11
Overall Logic Flow of Driver Program

In the first scheme, we fix the data blocks and operate the programs on these various data blocks sequentially. The data blocks must also be produced and/or refreshed sequentially with this scheme. This method presents serious difficulties since addressing becomes complicated and programs which require a continuous observation window are not easily accommodated.

In the second scheme (the one presently employed in the ICS) the programs operate on fixed blocks in memory and the data stream is "slid" by the program's "observation window". With this scheme, it is easy to provide past and future data on either side of the observation window for programs that have such a requirement. This is necessitated by certain phase estimation and bit synchronization algorithms or any program module employing some type of memory (i.e., Viterbi algorithm).

Consider the channel block diagram illustrated in Fig. 19. In this diagram, IBLK, IBLK1, IBLK2, IBLKR, IBLKB, IBLKN and IBLKM are Fortran variables which are calculated by the driver at run time. IBLKR is the run address for the receiver program and is fixed at the top of the 'present' block.

The steady state mode of operation is illustrated in the flowchart in Fig. 20. (We ignore the start up mode since it is not germane to this discussion.) Before entering the flowchart, we assume block B has been processed by the receiver and we are now ready to process another block. At this time block A is full of data and block C has been processed one cycle earlier.

The following four steps are now performed:

1. blocks B and A are slid (moved) upward to be congruent to blocks C and B respectively. Previous data in block C is written over.
2. The transmitter program is exercised to fill block A with data.

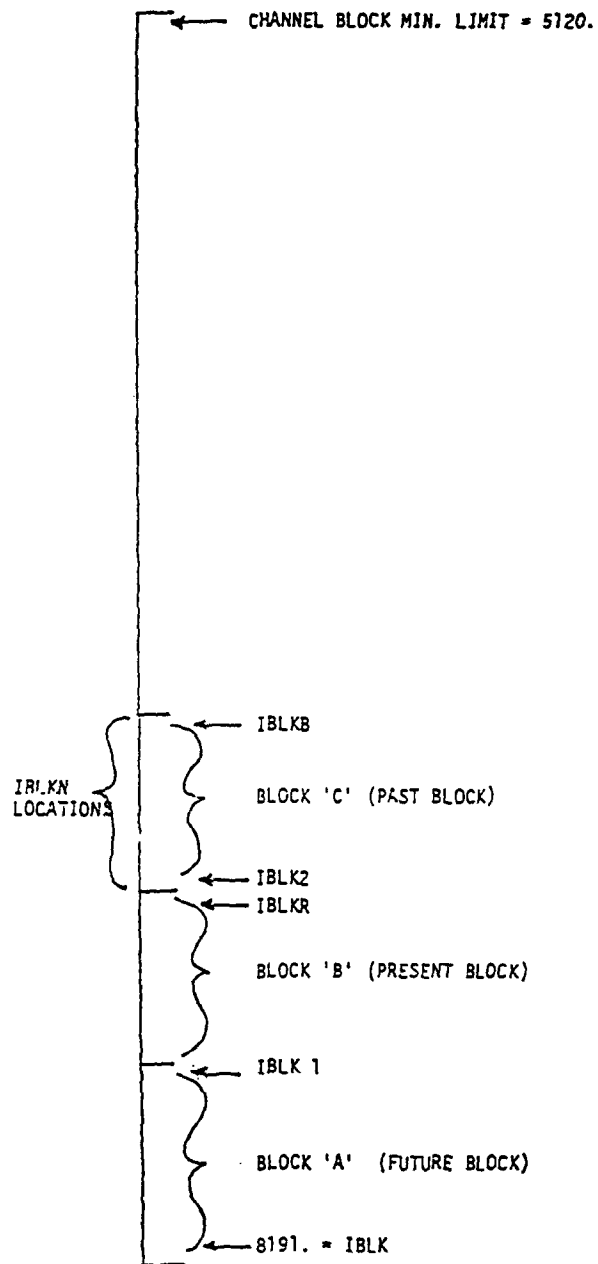


Figure 19
Channel Data Block Diagram

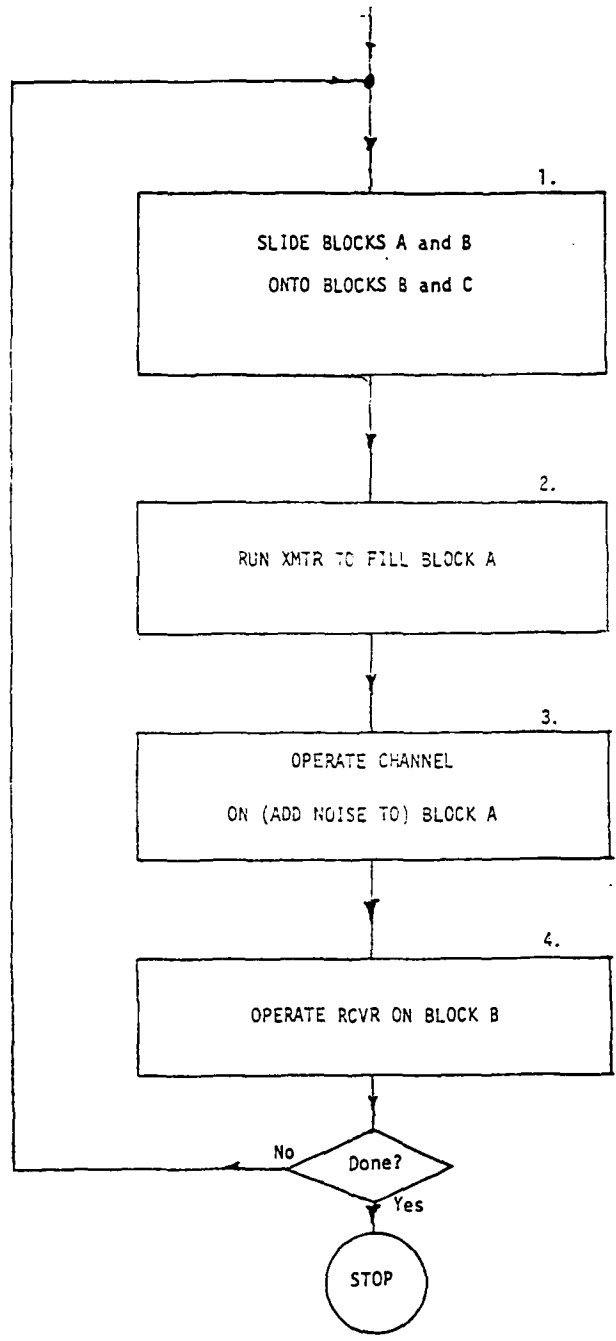


Figure 20
Logic for Sliding Data Stream Structure

3. The channel program is operated to corrupt the transmitter output with noise.

4. The receiver is again operated on block B.

If intermediate results (for waveform plots, etc.) are to be collected, this information may be accessed between the above steps by a DMA transfer to the host (and perhaps subsequently to disc or tape).

Since the block length (NB or NBIT) is normally a complicated function of variables such as the number of samples/ baud and the code rate, it is necessary to calculate run addresses and block lengths at run time.

We normally allot the maximum data memory space for each program module and then use a portion of this space during a particular simulation. Specifically, consider the example below:

The number of source bits to be generated is given by

$$NBIT = \frac{512}{NS}$$

where NS is the number of samples per baud and may range from 4 to 32. We may be required to produce from 16 to 128 source bits. Therefore, we must allocate at least 128 memory locations for the source.

The memory layout is shown in Fig. 21. We have allotted 384 memory locations to the source. This allows for storage of up to 3 blocks of source data. The important point to note here is that the run address (IADRBR) is dynamically calculated at run time, allowing for greater flexibility in the simulation. The only addresses which are fixed are the top and bottom source block boundaries. All other simulator modules operate in a similar fashion.

4.3 SYSTEM SOFTWARE CONNECTION:

The simulator driver program controls the AP-120B through the software hierarchy diagrammed in Fig. 22.

As the simulation evolves, each communication system module is called in turn by the driver program. (STEP 1)

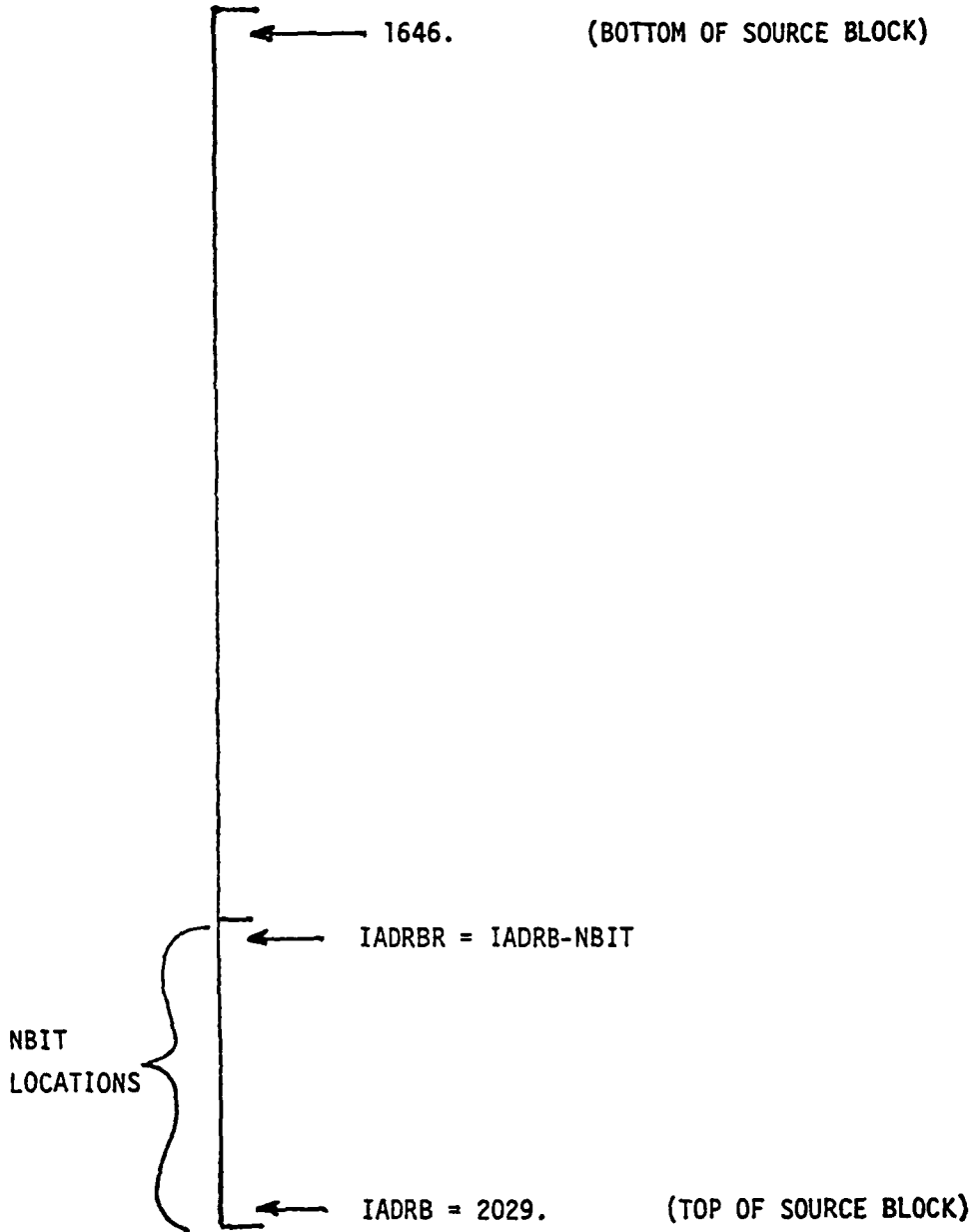


Figure 21
Dynamic Addressing Scheme in ICS

R.P.I. INTERACTIVE COMMUNICATIONS SIMULATOR

SOFTWARE CONNECTION

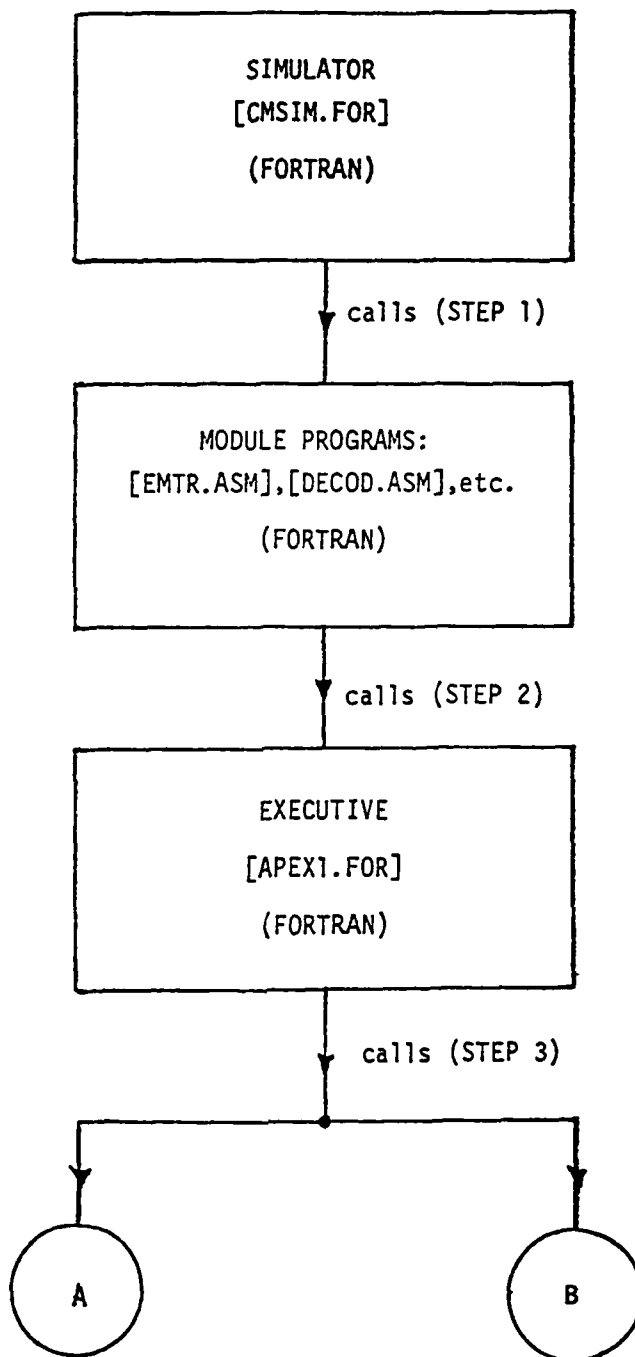


Figure 22.1
ICS Software Connection

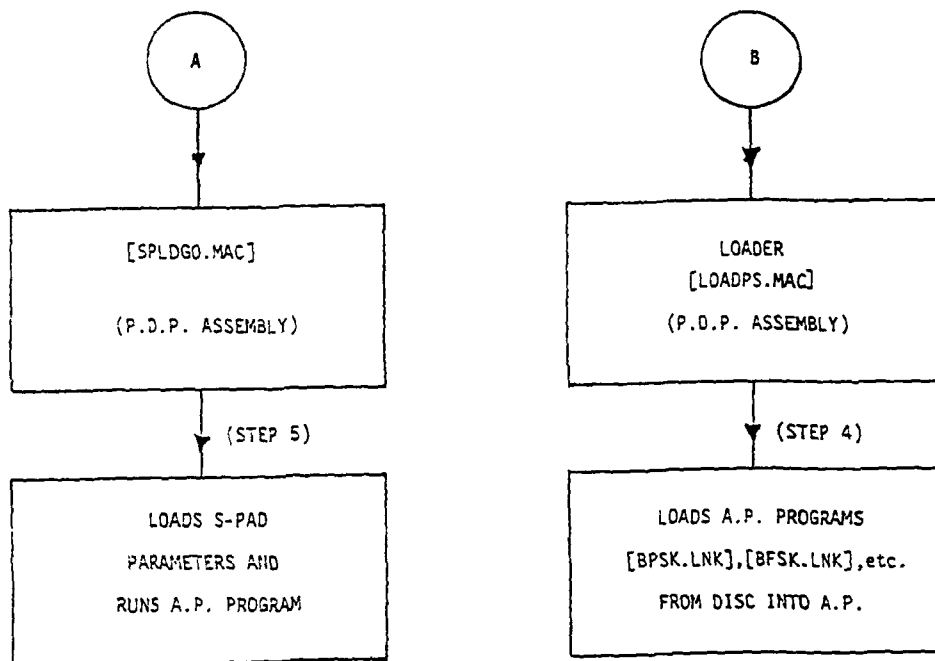


Figure 22.2
ICS Software Connection

Each of the individual module programs keep a table of PDP-11/40 disc file names and lengths corresponding to the various options implemented for that system block. For example, the XMTR module has a list of files which include BPSK.LNK and DPSK.LNK. These are load module files which have been prepared using APAL and APLINK.

As the system module program is called, it is also passed an option index which is originally derived from the DESIGN file. This option index acts as a pointer into the table of file names.

Once a file name has been selected, this name, along with any S-PAD parameters is passed to the executive program. (STEP 2)

The executive program now calls various PDP-11/40 assembly language programs (STEP 3) which load the appropriate files into the AP-120B (STEP 4) and then loads S-PADS and runs the AP. (STEP 5).

4.4 AP-120B PROGRAM REQUIREMENTS:

Although the array processor program characteristics may vary considerably, depending on their application in the simulator system, there are certain universal requirements which must be met in order to be consistent with the overall simulator structure.

Because of the dynamic addressing, each program must have the capability for its input and output data addresses to be loaded at run time. Similarly, since the data block length is a variable which is calculated at run time, it must be loaded as well.

This information can either be passed directly through the Fortran call by implicit use of the SPLO Go (S-PAD load and go) routine or through a DMA to predetermined AP memory locations. In the latter case, the AP program may need to subsequently transfer these memory locations to S-PADS depending on the structure and application of the AP program.

5.0 GRAPHICS SUPPORT PACKAGE:

To provide the interactive graphics capabilities of the ICS, a fairly extensive GRAPHICS SUPPORT package was developed and has been in a constant state of evolution. This package consists of a number of waveform plotting routines which access general purpose graphics subroutines which have been developed and are stored in a Graphics Library on the PDP-11/40. The purpose of this section then is to provide an overview of this GRAPHICS SUPPORT package. Additional details, including listings, can be found in [24], [25].

5.1 PLOTTING ROUTINES:

In the VALIDATION mode there are six separate time-domain plotting programs as illustrated in Table 3. The data structures are such that the same basic algorithm can be used for all six plotting schemes. In the VALIDATION mode we are interested in the output of fourteen separate modules. These fourteen modules are divided into three classes, the BASEBAND output, the CHANNEL output, and the SYNCHRONIZATION output (see Table 4). Each of these classes has the option of plotting either individually, in the SINGLE WAVEFORM plotting option, or in conjunction with other data of the same class, in the MULTIPLE WAVEFORM plotting option.

The flowchart for the basic plotting algorithm is illustrated in Fig. 23. Each program begins with a short description of its purpose and a list of all the object modules that it is linked to. Every variable is identified and its function in the program specified. The names of the programs to be linked to are encoded in Radix-50 at this point. The subroutine RCHAIN is called to determine whether or not the program was chained to; this function is not necessary for the operation of these programs, but it must be called in order to chain at the end of the program. The next phase involves clearing the display processor, activating it, and preparing it to display plots.

<u>PROGRAM NAME</u>	<u>CLASS PLOTTED</u>	<u>TYPE</u>
ANAL2.SAV	CHANNEL	SINGLE
ANAL3.SAV	CHANNEL	MULTIPLE
ANAL4.SAV	BASEBAND	SINGLE
ANAL5.SAV	BASEBAND	MULTIPLE
ANAL6.SAV	SYNCHRONIZATION	SINGLE
ANAL7.SAV	SYNCHRONIZATION	MULTIPLE

Table 3
Time-Domain Plotting Programs
Used in VALIDATION Mode

<u>BASEBAND OUTPUT</u>	<u>CHANNEL OUTPUT</u>	<u>SYNCHRONIZATION OUTPUT</u>
SOURCE	TRANSMITTER	PHASE TRACKING ERROR
ENCODER	TRANSMITTER FILTER	BIT SYNCHRONIZATION ERROR
MATCHED FILTER	CHANNEL	
DECODER	FIRST CHANNEL FILTER	
RECEIVER	ZERO MEMORY NON-LINEARITY	
	SECOND CHANNEL FILTER	
	NOISE	

Table 4

Breakdown of Different Plotting Schemes.

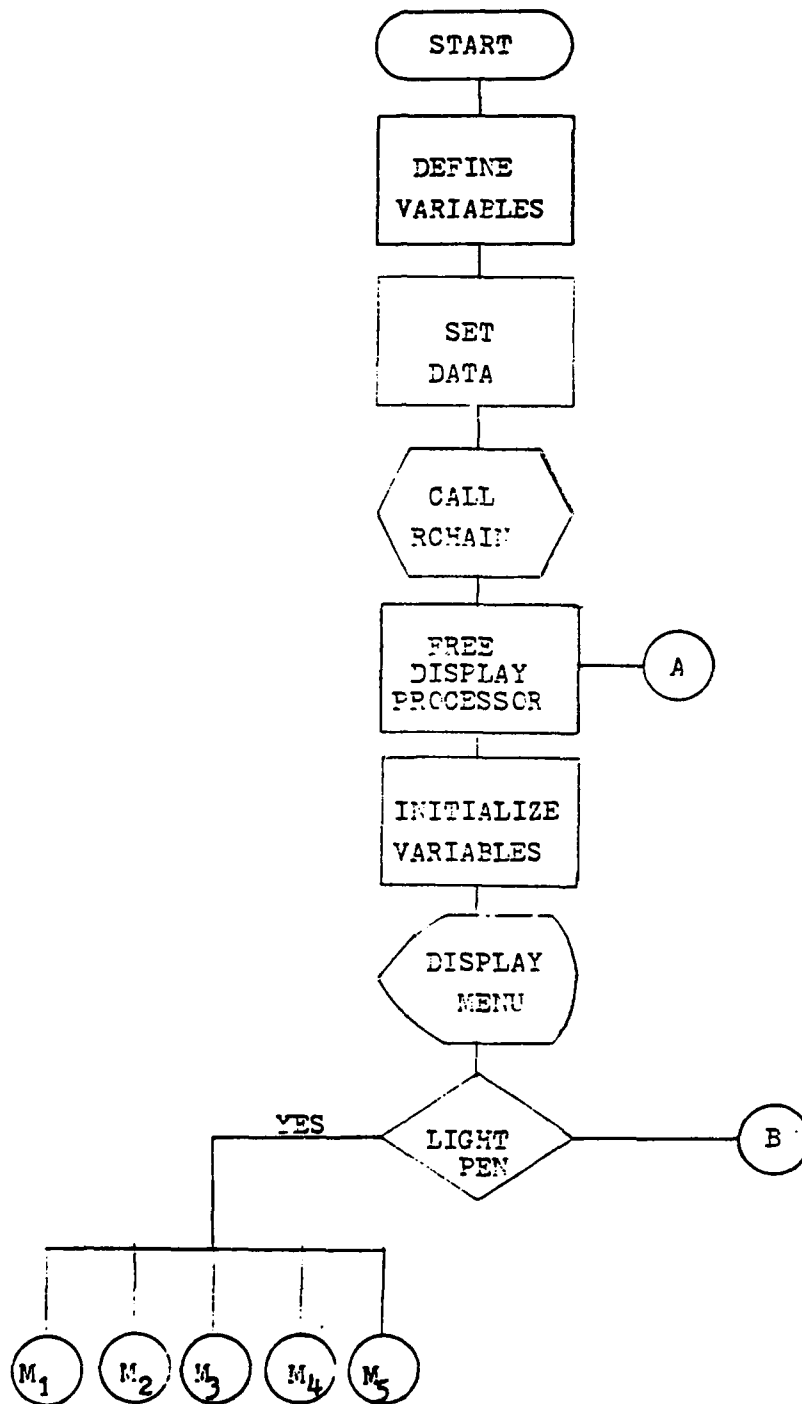


Figure 23.1
Flowchart for plotting routines

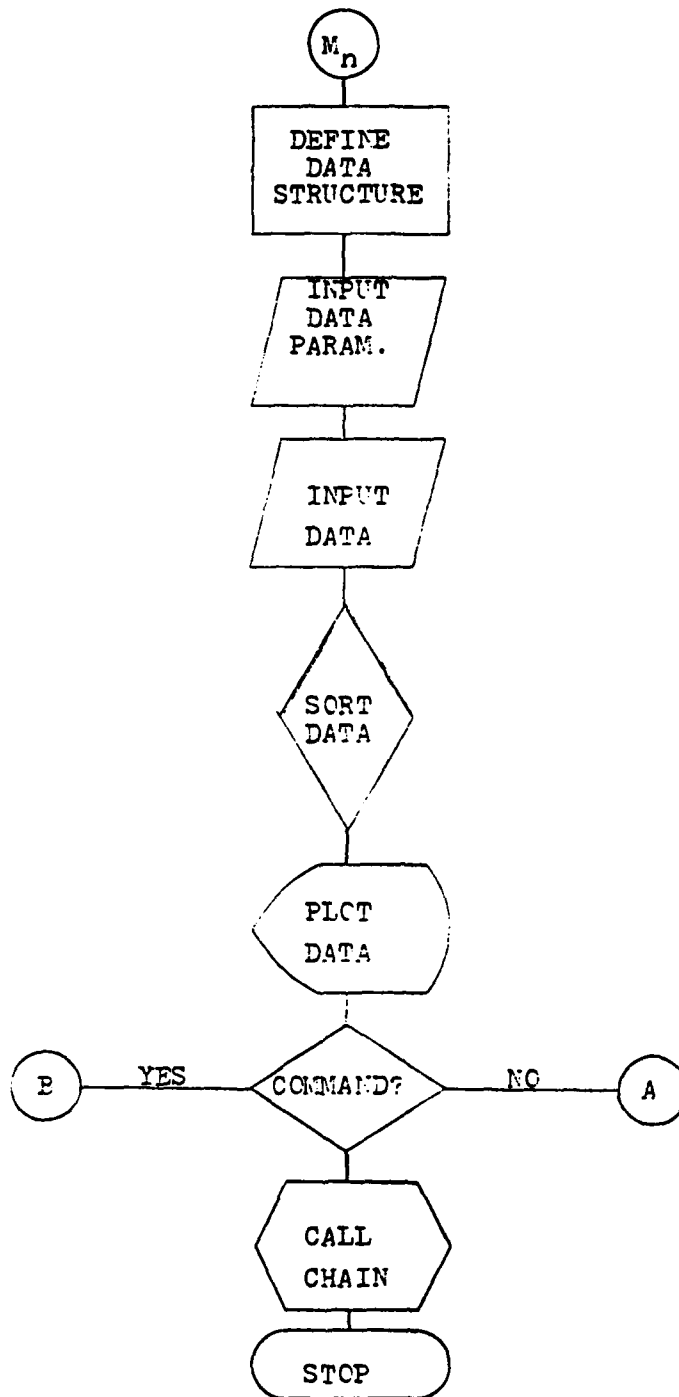


Figure 23.2
Flowchart for plotting routines

A number of variables are now initialized to their default values. The window size is set at either 4 or 7 baud and the record and element pointers are initialized to the first element of the first data record. The screen is activated and a menu of all the available plots is listed on the screen. When the user makes a choice the program branches to a section of the program that will obtain the data corresponding to the user's choice. The data structure of the chosen module is defined, parameters related to the data such as the length of the data file, the useful portion of the data file, etc. are read in by the program. The program then proceeds to call subroutine MSHFT; a subroutine that creates a continuous data file, i.e. a file without spaces or undesirable data.

The program is now ready to utilize the Graphics Library. The origin of each coordinate system must be established by calling subroutine ZINIT (see subroutine ZINIT). The X-axis is scaled according to the number of baud to be plotted. The Y-axis is scaled on the basis of the largest value in the data array. Negative values are also taken into account and their absolute value is used in determining the largest value in the array. Both scale factors are determined and stored in the common block SCFILE (see Common Block SCFILE). The next two subroutines called make use of these scale factors and also the maximum and minimum values in the array as they draw the two axis and a grid to facilitate viewing. Finally, the data is plotted as points and interconnecting lines are drawn.

The program now enters the Command mode, a mode where the display may be altered until the desired effect is achieved and a hard copy obtained. This mode offers the user the ability to exercise a maximum of 5 options. After viewing, the program then is chained back to the ICS DRIVER PROGRAM.

The Command Mode

- RECORD n - This allows the user to go to the beginning of the nth data record and look at the default number of baud at the beginning of that record. The number of records in the file is printed as a guide to the user.
- CHANGE SIZE - This allows the user to change the window size from 1 baud to a maximum determined on the basis of the number of samples per baud and the useful portion of the data per data record. The maximum is printed as a guide to the user.
- SHIFT WINDOW - This allows the user to shift the window and look at a different part of the data stream. Since there is a continuous stream of data the entire file may be viewed with this command. When the user attempts to exceed the boundaries of the file an error message is printed warning the user of the problem. At present there is no algorithm to determine shifting limits, but I believe that one is feasible.
- HARD COPY - This allows the user, through the Versatec, to make a hard copy of the display. This part of the program may also be accessed independently by saving the display in a file and running the program HCOPY.SAV.
- FREQUENCY DOMAIN - This option is available with the individual channel output waveforms only. It chains to a program developed to plot the FFT of the display in either polar or rectangular form. The FFT program is then chained back to the individual channel output waveform program.
- EXIT - This command is only an exit command as far as the command mode is concerned. If the individual plot mode was selected, then at this time this program would return and ask the user if he wished to view

any other individual plot. In the simultaneous plot mode the program would exit this program and chain with the simulator driver.

5.2 GRAPHICS LIBRARY :

The diversity of the types of plots needed for the ICS mandated that the plotting routines be as modular as possible. For this reason twenty-two separate subroutines were developed. Each subroutine serves a specific purpose and can be readily applied to particular applications.

It is well worth noting that because of the general nature of these subroutines they may be utilized in developing plotting capabilities for any system of moderate complexity.

For ease of understanding we list each subroutine, define its variables, and give a description of its function and algorithmic implementation. Listings can be found in [25].

Subroutine LHIT

IND - The number of the menu item selected.

ISTART - The number of the subpicture tag corresponding to the first menu item.

INUM - The number of items in the menu.

The purpose of this subroutine is to service a user specified menu by waiting for a light pen hit. The subroutine returns the number of the menu item selected.

Subroutine ZINIT

IXORG - X-axis origin in Raster's.

IYORG - Y-axis origin in Raster's.

This routine inserts the coordinates system origin into the common block SCFILE. The input is in Raster's; the program merely transfers these values into the common block.

Subroutine ZMMAX

ARRAY - Array containing data points.

NPTS - Number of points in ARRAY.

INC - Incremental value for points to be considered.

SMAX - Value returned by subroutine as maximum in ARRAY.

SMIN - Value returned by subroutine as minimum in ARRAY.

IND - Dummy variable.

This routine scans a user specified array and determines the largest and smallest values in the array. The routine will only consider every INC-number of points; thus to consider every point in the array a one should be entered.

Subroutine ZCNR

RN - Real number to be converted into raster units.

R - Raster equivalent of input.

IWHO - Identifies the appropriate axis

IWHO=1 then X-axis.

IWHO=0 then Y-axis.

The purpose of this subroutine is to convert real numbers into raster units. This is done by obtaining the scale factor for the appropriate axis and normalizing the number to the appropriate boundary.

Subroutine ZCRN

RN - Real number equivalent of input.

R - Raster number to be converted into a real number.

IWHO - Identifies the appropriate axis.

IWHO=1 then X-axis.

IWHO=0 then Y-axis.

This subroutine scales the appropriate axis by taking the maximum and minimum values, converting these to Raster's (see subroutine ZCRN), determining the difference, and then dividing by the axial length. These values are then stored in the common block SCFILE.

Subroutine ZSCALE

ARRAY - Array containing data points.

NPTS - Number of data points in array to be considered.

IAXL - Axis length in Raster units.

INC - Increment value of array to be considered.

IWHO - Identifies the appropriate axis.

IWHO=1 then X-axis.

IWHO=0 then Y-axis.

The purpose of this subroutine is to determine the scale factor for the appropriate axis. This differs from subroutine ZQSCAL in that for this subroutine the actual array and number of points are input. In subroutine ZQSCAL the maximum and minimum values are entered whereas here they are determined.

Subroutine ZLSCAL

ARRAY - Array containing data points.

NPTS - Number of points in array to be plotted.

IAXL - Axis length in Raster units.

INC - Increment value of array to be considered.

IWHO - Identifies the appropriate axis.

IWHO=1 then X-axis.

IWHO=0 then Y-axis.

This subroutine determines the scale factors for a logarithm plot. This program invokes subroutine ZTENSO to determine the boundary values. Then subroutine ZQSCAL is called to determine the scale factors to be inserted into the common block SCFILE.

Subroutine ZYAXIS

LABEL - ASCII character string label for Y-axis.

Maximum=20 characters (including blanks)

NCHAR - Number of characters in LABEL.

IPER - Number of decimal places on values on Y-axis.

The purpose of this subroutine is to draw a linear Y-axis on the screen. This is done by recalling the values in the common block SCFILE. Placement of the label is made on the basis of the axis boundaries.

Subroutine ZLYAX

LABEL - ASCII character string label for Y-axis.

Maximum=20 (including blanks)

NCHAR - Number of characters in LABEL.

The purpose of this subroutine is to draw a logarithmic Y-axis on the screen. This is done by recalling the values in the common block SCFILE. Placement of the label is made on the basis of the axis boundaries.

Subroutine ZXAXIS

LABEL - ASCII character string label for X-axis.

Maximum=20 (including blanks)

NCHAR - Number of characters in LABEL.

IPER - Number of decimal places on values on X-axis.

INUM - Flag to indicate whether or not to print the LABEL

INUM=0 then do not print the LABEL.

INUM=1 then print the LABEL.

This subroutine draws out a linear X-axis on the basis of the maximum and minimum values in an array. Using the scale factors it will continue to draw vertical lines until such time that the spacing between the lines is less than 75 Raster units. (This distance may be changed @ line 51 of this routine.)

Subroutine ZHOLD

ARRAY - Array containing data points.

NPTS - Number of points to be plotted.

IDX - X-axis increment.

This subroutine is used to plot binary data. It will maintain either a one or a zero for IDX number of points. When utilizing this subroutine it must be remembered that IDX is the number of points to be tracked, and not the number of changes to be recorded.

Subroutine ZDATA

YARA - Array containing Y-axis data.

NPTS - Number of points in YARA.

INC - Increment value of array to be considered.

INT - Intensity of data points and interconnecting lines.

IFL - Flashing parameter for data points.

ITYPR - Type of lines to be printed to connect the points.

This subroutine connects YARA data points. There is an underlying assumption that these points are evenly spaced relative to the X-axis. The program calls the subroutine ZCRN and converts the data to Raster units. The increment value for the X-axis is determined by the difference between the first two points and a line is drawn between these two points. The procedure continues until all the points have been processed.

Subroutine ZSDATA

XARA - Array containing X-axis data.

YARA - Array containing Y-axis data.

NPTS - Number of points in XARA and YARA.

INC - Increment value of array to be considered.

ICHAR - Character type

ICHAR=1 then 'X'.

ICHAR=2 then 'I'.

ICHAR=3 then '+'.

ICHAR=4 then 'C'.

This subroutine plots a character at the data points. It will not draw interconnecting lines between these points.

Subroutine ZDATA1

XARA - Array containing X-axis data.

YARA - Array containing Y-axis data.

NPTS - Number of points in XARA and YARA.

INC - Incremental value of array to be considered.

INT- Intensity of data points and interconnecting lines.

IFL - Flashing parameter for data points.

ITYPR - Type of lines to be printed to connect the points.

This subroutine will plot and connect points with non-constant incremental X-axis values. The program reads in the coordinates for the first point; on the basis of the scale factors and the boundary conditions located in the common block SCFILE, it plots the point.

Subroutine ZSAVE

The purpose of this subroutine is to save a graphical display on the disc for later viewing. The subroutine requests a name for the display; a name of the form "RKL:XXXXXX.PIC" is entered where XXXXXX is a user specified name of no more than six characters.

Subroutine ZSDEC

ARRAY - Array containing data points.

NPTS - Number of points in array.

IAXL - Axial length in Raster units.

INC - Incremental value of data to be considered.

IWHC - Identifies the appropriate axis.

IWHO=1 then X-axis.

IWHO=0 then Y axis.

This subroutine sets up a decibel scale on the specified axis. By converting ARRAY into decibels and then converting these into Raster units, we are able to determine the scale factor. These values are then stored in the common block SCFILE.

Subroutine ZNUMBER

VAL - Number to be printed.

IPREC - Number of numbers to follow decimal point.

i.e. degree of accuracy

This subroutine is used to place the user defined degree of accuracy on the screen. The user specifies IPREC-digits to follow the decimal point, the routine calls the system function 'ENCODE' and it is placed on the screen as an axis label.

Subroutine ZAUTOF

VAL - Number to be formatted.

IFOR - Format type for VAL.

NFRM - Field length of IFOR for VAL.

IPREC - Number of numbers to follow decimal point.

i.e. degree of accuracy.

This routine given that VAL is to be printed on the screen with IPREC degree of accuracy will determine the format type of VAL and the field length required. By searching VAL and determining the location of the decimal point the routine then decides on whether to label it INTEGER, FLOATING, or GENERAL.

Subroutine WAIT

INUM - A relative indicator of the length of time for the system to halt.

This routine performs a predetermined computation $(200) * (\text{INUM})$ times. It will perform the computation and then return to the calling routine.

Subroutine TRACK

This subroutine is used for documenting graphs on the screen. A set of options are made available through this routine directly. For example, he may draw a line on the screen,; he may input characters on the screen in either a vertical or horizontal direction. The positioning of everything is done with the use of the light pen and the tracking object.

Common Block SCFILE

Since this common block appears in nearly all of the packages subroutine, an explanation of its content is in order.

ZXSF - X-axis scale factor.

ZYSF - Y-axis scale factor.

ZXORG - X-axis origin in Raster's.

ZYORG - Y-axis origin in Raster's.

ZXMAX - Maximum X-axis value in Raster's.

ZYMAX - Maximum Y-axis value in Raster's.

ZRXMIN - Minimum X-axis value in real numbers.

ZRYMIN - Minimum Y-axis value in real numbers.

6.0 FUTURE ICS ENHANCEMENTS:

Upon review of the present status and capabilities of the Interactive Communications Simulator (ICS) there are several obvious directions in which this system could be updated. The following is a partial list of several of the more straightforward extensions:

- 1.) Source Coding: At present the ICS includes a binary memoryless source producing equiprobable symbols. Since the ultimate goal of many digital communications systems is to provide a transmission capability for analog sources, it is of some interest to consider the addition of a source coding capability. We list here several options which could easily be provided.
 - a.) PCM/DPCM including delta modulation.
 - b.) Block transform coding using various unitary transforms.
 - c.) Tree-encoding schemes which offer the potential of performance approaching the rate-distortion bound.
 - d.) Minimum mean-square coding concepts.
- 2.) Spread-Spectrum Capability: The existing channel model incorporated into the ICS consists of a three-component fading-dispersive channel with an impulsive noise capability. It would be of some interest to provide the ability to simulate spread-spectrum modulation under various interference or jamming conditions. Specifically, it would be possible to simulate direct sequence (DS) or frequency-hopped (FH) modulation by PN sequences. The various interference or jamming conditions could be appropriately modeled as finite-state burst channels.
- 3.) Explicit Diversity: At present we can simulate a single fading-dispersive channel. It would be of significant utility if a multichannel capability were provided. This would be useful, in particular, for modulation/coding studies.

- 4.) Spectrum Efficient Modulation: The ICS presently includes a number of classical modulation strategies such as BPSK, QPSK, OQPSK, MSK, etc. While relatively efficient on the basis of symbol error probability for a given expenditure of E_b/N_0 , these modulation strategies are not necessarily efficient in terms of spectral occupancy. There is presently a great deal of interest in the use of combined amplitude/phase coding techniques which offer bandwidth efficiencies in excess of 2 bits/sec/Hz. It would be relatively easy to include such modulation strategies in the ICS. Indeed, it would be possible to allow the user freedom to choose quite arbitrary signaling constellations in amplitude/phase space.
- 5.) Cross-Polarization Interference: In an effort to improve bandwidth, efficiency of microwave line-of-sight (LOS) digital radio systems use of independent cross-polarized transmission channels have been used. Unfortunately, non-ideal propagation conditions result in mutual or co-channel interference which significantly degrades performance. These propagation induced interference effects could be realistically and flexibly simulated in the ICS.
- 6.) Adaptive Nulling: A wide variety of future military communication systems will make use of adaptive array nulling systems of one form or another. Unfortunately, the dynamic effects of these nulling systems can have a profound influence on communication system performance. These effects are often not amenable to analytical evaluation. It would be of some interest, then, to provide the capability of simulating these effects within the ICS and thus provide explicit means for evaluating the effects on communication system performance.
- 7.) Performance Monitoring: Future military communication systems will undoubtedly be provided with some form of performance monitoring

capability to predict link outages and/or reliability. Many of the more promising performance monitoring techniques have not been evaluated in a realistic communications environment. This could readily be accomplished by incorporating performance monitoring techniques into the ICS.

- 8.) Nonlinear Receivers: The existing ICS has a fairly extensive impulse noise simulation capability. The receiver structures for the various modulation capabilities are invariably linear. These receiver structures suffer drastically from the effects of impulse noise. Various non-linear receiver structures of the limiting variety have been used to mitigate these effects. Several of these non-linear structures have been included in the ICS. It would be a relatively simple matter to incorporate a wider range of nonlinear receivers into the ICS. This would be extremely useful in evaluating modulation/coding concepts in impulsive noise environments.
- 9.) Adaptive Equalization: The ICS has a variety of fixed equalizer capabilities. At present, we utilize a "cheater" line to tell the equalizer what the channel structure is. In practice this should be automatic. That is, the equalizer should acquire and track the unknown channel parameters. It is proposed, then, that a truly adaptive equalizer capability be provided as part of any future upgrading of the ICS.
- 10.) Optimum Receiver Structures in AWGN: In addition to data demodulation, various ancillary functions such as equalization, phase/timing synchronization, etc., must be accomplished in typical receiver module. On the AWGN channel, linear matched filter outputs sampled at the baud rate provide sufficient statistics for accomplishing this. More general receiver structures could easily be provided in the ICS to allow

flexible simulation of optimum receivers. This would allow use of the ICS as a test-bed for exploring new modem concepts.

11.) Optimum Receiver Structures for Impulsive and/or Fading-Dispersive Channels:

Little work has been done in characterizing the structure and performance of optimum receivers for impulsive and/or fading-dispersive channels.

On the basis of recent work in this area it would be a relatively straightforward task to provide simulation modules for fairly general optimum structures in the ICS. This would allow investigation of various suboptimum structures which approximate optimum and presumably more complex receivers.

12.) Channel Coding/Decoding: There are several directions in which the channel coding/decoding capabilities could be upgraded. We list several obvious directions:

- a.) Additional binary block codes and associated decoders.
- b.) Provide a concatenated coding/decoding capability.
- c.) Implement Chase's soft-decision decoding algorithm.
- d.) Provide a sequential decoding capability.

13.) Improved Graphics: The interactive computer graphics provided with the ICS is in a constant state of evolution. There are always some things that can be added to enhance and otherwise improve the operation of the ICS. We mention a few:

- a.) Provide additional capabilities for plotting and displaying error statistics.
- b.) Provide 3-D plotting capabilities for scattering functions, 2-D histograms of phase and bit timing errors, etc.
- c.) Provide capabilities for computation and display of amplitude probability distributions (APD's) of noise waveforms.
- d.) Allow plotting and display of eye-diagrams showing the effects

AD-A101 428

RENSSELAER POLYTECHNIC INST TROY NY DEPT OF ELECTRIC--ETC F/0 17/2
DESIGN AND IMPLEMENTATION OF THE INTERACTIVE COMMUNICATIONS SIM--ETC(U)
APR 81 J W MODESTINO, K R MATIS, K Y JUNG F30602-78-C-0083

UNCLASSIFIED

TR-80-1

RADC-TR-81-37

NL

2 of 2

APR 81



END

DATE

FILED

8-81

DTIC

of intersymbol interference (ISI).

e.) Improved waveform plotting capabilities in both the time and frequency domain.

f.) Provide software improvements to enhance computational speed.

14.) Narrowband Filtering: At present the ICS possesses an infinite impulse response (IIR) narrowband filtering capability. The design procedure employed allows control over magnitude response but not phase response. It would be of some interest to provide a finite impulse response (FIR) capability as well. This would allow control over both magnitude and phase response and thus provide added flexibility.

15.) Interleaving Capability: With the addition of expanded data memory it would be a relatively simple matter to provide an interleaving capability. This should include both block and convolutional interleaving capabilities.

6.1 HARDWARE IMPROVEMENTS:

In order to implement many of the suggested enhancements indicated above, consideration should be given to simultaneous hardware improvements. At the very least this should include:

- 1.) Program Source Memory: Should be increased from present 1K to 1.5-2K. This will allow increased efficiency/throughput for ICS.
- 2.) Data Memory: Should be increased from present 16K to 32K. This would allow explicit diversity and/or interleaving capability.
- 3.) Graphics Terminal: The existing VT-11 terminal should be replaced by a higher resolution CRT with improved writing speed and graphics support software.
- 4.) Real-Time Interface: The real-time interface being installed on the AP-120B should allow interface of the ICS to real channels. This possibility should be explored.

References

1. J. P. Odenwalder, "Optimum Decoding of Convolutional Codes", Ph.D. Dissertation, Syst. Sci. Dept., Univ. California, Los Angeles, 1970.
2. K. J. Larsen, "Short Convolutional Codes with Maximal Free Distance for Rates $1/2$, $1/3$ and $1/4$ ", IEEE Trans. Inform. Theory, Vol. IT-19, pp. 371-372, May 1973.
3. W. C. Linsey and M. K. Simon, Telecommunication Systems Engineering, Prentice-Hall, Englewood Cliffs, N.J., 1973
4. J. J. Spilker, Jr., Digital Communications by Satellite, Prentice-Hall, Englewood Cliffs, N.J., 1977.
5. P. A. Bello, "Characterization of Randomly Time-Variant Linear Channels", IEEE Trans. Commun. Syst.; Vol. CS-11, pp. 360-393, Dec. 1963.
6. R. S. Kennedy, Fading Dispersive Communication Channels, Wiley-Interscience, New York, 1969.
7. J. W. Modestino and B. Sankur, "Modeling and Simulation of ELF/VLF Noise", Proc. Seventh Annual Pittsburgh Conf. on Modeling and Simulation, Pittsburgh, PA., pp. 189-194, April 1976.
8. J. W. Modestino and B. Sankur, "Modeling and Analysis of Impulsive Noise", Proc. of NATO Conf. on Communication and Signal Processing, Darlington, England, pp. 619-636, Aug. 1977.
9. G. D. Forney, Jr., "Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference", IEEE Trans. Information Theory, Vol. IT-18, pp. 363-377, May 1972.
10. J. G. Proakis, "Advances in Equalization of Intersymbol Interference", Chapt. 3 in Advances in Communication, Ed. by A. J. Viterbi, Prentice-Hall, Englewood Cliffs, N.J., 1975.
11. P. Monsen, "Feedback Equalization for Fading Dispersive Channels", IEEE Trans. Inform. Theory, Vol. IT-17, pp. 56-64, Jan. 1971
12. P. Monsen, "Digital Transmission Performance on Fading Dispersive Diversity Channels", IEEE Trans. on Commun., Vol. COM-21, pp. 33-39, Jan. 1973.
13. P. Monsen, "Theoretical and Measured Performance of a DFE Modem on a Fading Multipath Channel", IEEE Trans. on Commun., Vol. COM-25, pp. 1149-1153, Oct. 1971.
14. A. J. Viterbi, "Convolutional Codes and Their Performance in Communication Systems", IEEE Trans. Commun. Tech., Vol. COM-19, pp. 751-772, Oct. 1971.
15. G. D. Forney, Jr., "The Viterbi Algorithm", Proc. of IEEE, Vol. 61, pp. 268-278, March 1973.
16. J. K. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis", IEEE Trans. Inform. Theory, Vol. IT-24, pp. 76-81, Jan. 1978.

17. J. A. Heller and I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communication," IEEE Trans. Commun. Tech., Vol. COM-19, pp. 835-848, Oct. 1971.
18. J. W. Modestino, "Error Probability Bounds for Soft-Decision Decoding of Block Codes," TM 78-30, unpublished RPI report, Nov. 1978.
19. D. G. Daut, J. W. Modestino, and L. D. Wismer, "New Short Constraint Length Convolutional Code Constructions for Selected Rational Rates", submitted to IEEE Trans. on Inform. Theory.
20. K. R. Matis and J. W. Modestino, "Reduced-State Soft Decision Trellis Decoding of Linear Block Codes", submitted to IEEE Trans. on Inform. Theory.
21. AP-120B Programmer's Reference Manual.
22. RPI Technical Memo TM78-17.
23. AP-120B Internal Interface Manual.
24. PDP-11/40, RT-11 Systems Reference Manual.
25. T. T. Murakami, "The Design and Analysis Stages for the Interactive Communications Simulator", M.S. Thesis, ESE Dept., Rensselaer Polytechnic Institute, Troy, N. Y., July 1979.
26. E. Godreau, III, "Interactive Graphics Support for the Communications Simulator", Project Report, ESE Dept., Rensselaer Polytechnic Institute, Troy, N.Y., June 1980.



MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.