To the Graduate Council:

I am submitting herewith a thesis written by Prabir K. Das entitled "Computer Simulation of Communication Systems using a Visual Programming Language." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Daniel B. Koch

Daniel B. Koch, Major Professor

We have read this thesis
and recommend its acceptance:

Donald W. Bouldin

Accepted for the Council:

Associate Vice Provost
and Dean of the Graduate School

COMPUTER SIMULATION OF COMMUNICATION SYSTEMS USING A VISUAL PROGRAMMING LANGUAGE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Prabir K. Das

August 1991

ACKNOWLEDGEMENT

The author wishes to express great appreciation to his advisor, Dr. Daniel B. Koch, for his invaluable guidance, encouragement, and patience throughout this study. Appreciation is also extended to Dr. Paul B. Crilly and Dr. D.W. Bouldin for their useful advice and comments. Thanks are due to the authors fellow graduate students and members of the Communication, Information and Signal Processing (CISP) Group of the University of Tennessee, Knoxville for many helpful discussions.

The author also acknowledges with thanks the support of the BellSouth.

ABSTRACT

The LabVIEW software language is a general purpose simulation tool using a new user friendly interface. Replacing lines of cryptic code with intuitive graphic objects, LabVIEW has simplified scientific programming to a great extent. The present work is an attempt to use this data flow concept in the simulation of communication systems. Programs for simulation of various analog and digital communication systems have been developed. Comparison of the simulated and theoretical results led to the conclusion that LabVIEW can be a very effective and interesting tool for communication systems simulation.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The design and analysis of a communication system is very difficult because of the number of different parameters and subsystems that can be chosen and varied. Although powerful methods are available to system engineers for a theoretical analysis, in many cases computer aided techniques for the simulation of a communication system as well as some of its parts is a convenient practical approach.

There are many problems, especially those with nonlinear elements and severe bandlimiting, which cannot be solved analytically. Digital simulation can often provide a useful and effective adjunct. Situations arise where explicit performance evaluation defies analysis and meaningful results can be obtained only through either actual hardware evaluation or digital computer simulations. Hardware evaluation has been found to be cumbersome, expensive, time consuming, and relatively inflexible. Digital simulation techniques therefore, have proven to be more practical.

Once a simulation model is developed, one can play with all kinds of "what if's ?". As the parameters can be easily varied, parametric studies are conducted without much work. Apart from guiding the analysis of the system modelled, the simulation environment provides for advanced systems study.

In a computer simulation approach the computer is used to simulate the voltage and current waveforms or signals that flow through the system. Waveform level simulation of a communication system consists of generating samples of all the input signals, performing discrete time signal processing operations, generating and storing the sampled values of the waveforms at selected points, and analyzing the waveform at the end of the simulation.

Several software packages have been developed that can be used for communication system simulation. Most of the packages have two major components: a simulation

component and an application library component [1]. The simulation component contains the software that enables the user to configure a communication system model, simulate it, and process the results of the simulation. The model library contains preprogrammed models of functional blocks of communication systems (Fig. 1.1). Typical elements in a model library are different types of signal sources, modulators, encoders/decoders, noise sources, filters, channels, demodulators, etc.

Typically the simulation component is made up of three sub-systems: the model configurator, the exercisor/simulator, and the postprocessor. The model configurator provides the means for a user to configure a model by specifying which library modules are to be included in the model as the module interconnections, which ports data are to be collected for use by the analyst or the postprocessor sub-system. It also allows the assignment of values to any settable parameters in a module and the specification of any checkpoints that are to be used by the exercisor subsystem. Finally it allows modification of a model previously configured.

The exercisor subsystem runs the simulation of the model and records the result in data files for subsequent use by the postprocessor sub-system.

The postprocessor provides the interactive facility with which the user may specify additional processing to be done on the simulation configuration. It aids the user through an on-line help; accepts values for user-settable parameters; allows modification of user specified parameter values, data, and processing operations; and displays graphic and/or tabular output of postprocessing results.

A simulation model can either consist of a rather complicated program written in a general purpose language expressly for the analysis of the communication system of interest, or it can consist of simple programs that use a software package providing a very

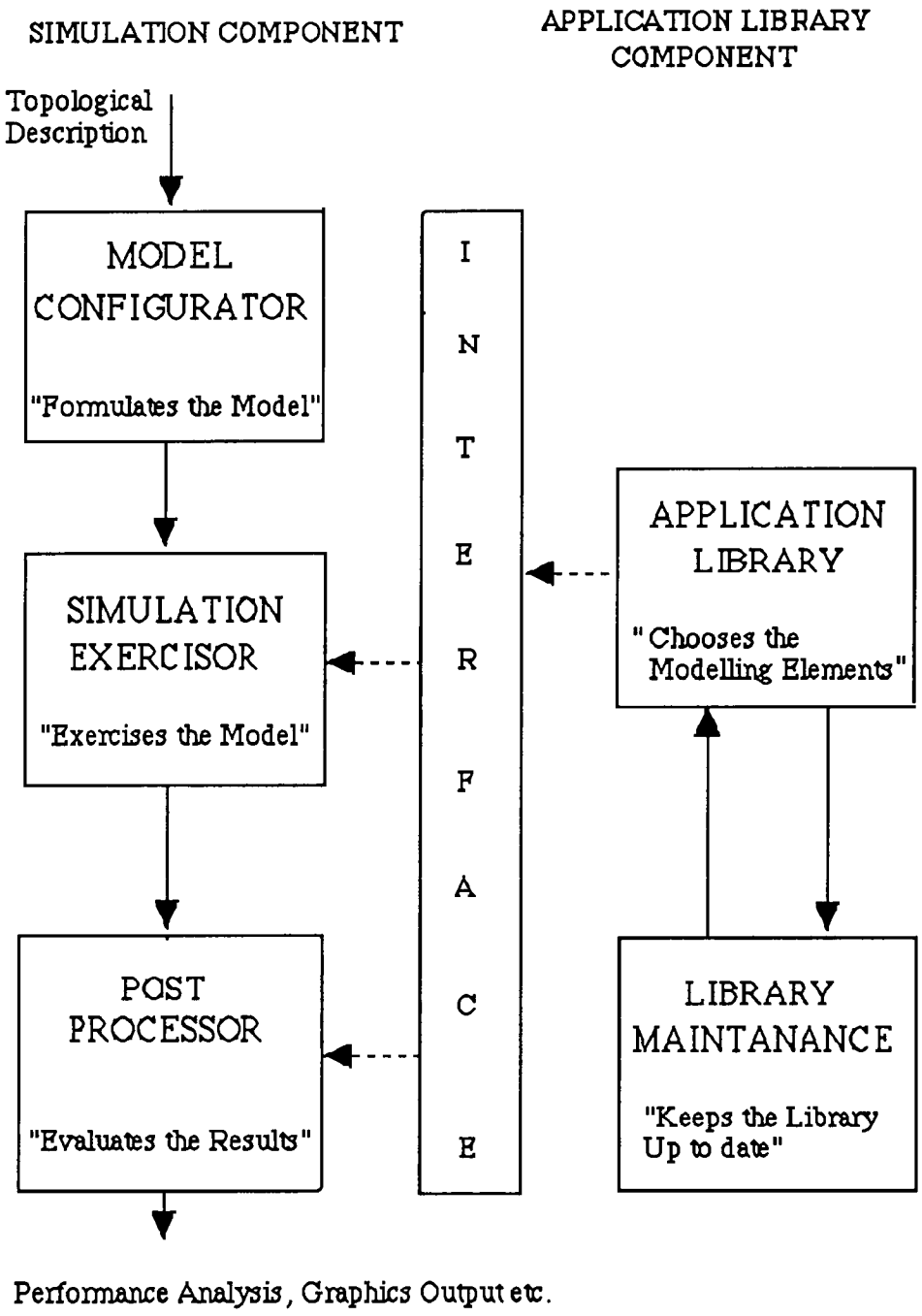


Figure 1.1 Operational Flow of a typical communication system simulation (after[1])

high language for the description of the system operations. Several popular software packages currently in use have been discussed in the literature. In any case a simulation tool for communication systems must be very versatile. It must allow the simulation of many different system parameters and structures with sufficient accuracy. An even more important requirement however, is that the package be very easy to use, in order not to discourage potential users. In the present work the capability of such a versatile and easy to use communication system simulation software has been evaluated. Development of simulation models and their performance for analog and digital communication systems using LabVIEW software will be presented.

CHAPTER 2

LITERATURE REVIEW

2.1. SOME POPULAR SOFTWARE PACKAGES

In order to ascertain the general state-of-the-art in communication system simulation, material on a wide variety of tools developed by different organizations has been reviewed. Historically these tools have ranged from packages that are language oriented and designed to operate in batch mode on mainframe computers to interactive and menu driven ones designed to run on workstations. Important features of some of the popular software packages currently in use for the simulation of communication systems are discussed below.

A. SYSTID [2]:

SYSTID is one of the earliest and most comprehensive software packages developed for communication system modeling and analysis. This software was developed by Hughes Aircraft Company under a contract from NASA. During 1969 a NASA contract initiated development of a topological subroutine (Model) specifically for communication system simulation. This work, which enabled efficient structuring of system simulation from library models, resulted in the initial version of SYSTID. SYSTID was enhanced for NASA in 1971 and 1974 resulting in the versions ASYSTD I and SYSTID-3 respectively. Up to this point SYSTID was written specifically for use on UNIVAC 1100 computers. Continued advances in solid state electronics led to the design and acquisition of a dedicated communication system simulator (CSS) facility for SYSTID in 1978. The CSS was

centered around a large scale minicomputer and array processor. Subsequently, SYSTID-3 was upgraded to the present version of SYSTID-4 in 1980.

In order to provide maximum flexibility, SYSTID maintains a modular structure. The operational flow has distinct phases like translation, simulation, and analysis. The process begins with a problem description constructed with a system editor. The problem is expressed in the SYSTID language and includes the simulation topology, run parameter definitions, and specification for the necessary input and output. The problem description is submitted to the SYSTID language processor which adds simulation control information and translates it into FORTRAN source code. In this process, reference to model libraries are identified and validated, input errors are diagnosed and the description is checked for correctness and consistency. The resulting output is combined with any necessary problem-specific FORTRAN routines and is submitted to the host FORTRAN compiler. If no error is detected, the object modules generated by this compilation are submitted to the host loader for assembly into an executable simulation program. The simulation phase typically consists of a batch mode execution of the program. It can produce a variety of outputs, including printed results, plots, and disk files containing time histories of the simulation. All of these outputs are employed in the final analysis phase, which is supported by a diverse set of interactive analysis programs.

In order to provide a simple user interface, SYSTID has developed a language processor coupled with a comprehensive model library. The language processor accepts a topological system description analogous to the system block diagram. The model library allows the system engineer to construct a simulation from commonly used elements like filters, modulators, coders, nonlinear amplifiers, and so forth. SYSTID offers great flexibility in the representation of system elements. A simulation element may be defined as a library model, a user written model, or a FORTRAN expression. In addition

FORTRAN declaration and executable statements may be intermixed with the topological problem description. In general, the simulation system is analogous to a FORTRAN main program and the "model" structures are analogous to FORTRAN subroutines. The capability of any model to call any other model to whatever level of nesting desired, is a striking feature. Data parameters can be communicated between the system and its models as well as between models.

In a simulation a signal is propagated through the models that represent the elements of the system. The resulting signal can be analyzed by one of the estimator models or postprocessor for evaluating some performance criteria. The primary estimators used with SYSTID include bit error rate, power spectrum, signal distortion, correlation, noise power ratio, crosstalk, and adjacent channel interference. Several support processors are used with SYSTID to aid in the preparation of results. An interactive graphics postprocessor, designed to run on the HP family of graphics terminals is used to display results using a variety of display formats.

SYSTID has proven to be a very powerful tool in the hands of an experienced user. It has been used extensively by Hughes Aircraft Company to design communication payloads for a variety of advanced communication satellites. CSS/SYSTID plays a key role in the system and subsystem specification phases of a project. It can provide a "quick look"and parametric analysis to support design decisions as well as supply data which aid in system and subsystem testing. It has also proved to be a timely and cost effective vehicle for the exploration of new concepts.

The major shortcoming of SYSTID is that it requires the user to write programs and remember a large amount of mundane details. On-line help is very limited. Another drawback is that simulation programs, results, and run-time parameters are not linked together meaning that the user is responsible for these and other bookkeeping operations.

B. TOPSIM III [3]:

TOPSIM III, developed at the Politecnico di Torono, Italy, is a very versatile simulation package for communication systems. It allows the simulation of many different systems and permits easy modification of the system parameters and structures. The digital simulation of both digital and analog communication systems and analysis of the interaction between analog and digital and digital signals within a single system is possible. The simulation is done strictly in the time domain. Filters specified in the frequency domain are converted to their time domain equivalents.

In the simulation operation the TOPSIM processor translates the source program written using TOPSIM III statements into a FORTRAN source program, which is then compiled to produce one or more relocatable modules. These modules are linked and relocated together with a standard MAIN program and with useful members of a library of relocatable TOPSIM III sub-programs. A PRERUN phase is then executed to evaluate the storage requirements of the simulation model. The PRERUN phase produces the simulation MAIN program that is compiled and successively linked and loaded together with the necessary relocatable modules. The simulation program is then executed, and results are presented on appropriate devices.

A user defines a model by writing a sequence of simple statements that closely match the block diagram of the system to be simulated. Each TOPSIM program is made up of three sections, named INITIAL, DYNAMIC, and TERMINAL. The DYNAMIC section is the most important part of the program and contains a description of the model given within a list of statements. In the INITIAL section the user defines the simulation timing parameters, as well as parameters of the various system blocks by means of

assignment statements similar to those used in FORTRAN. The TERMINAL section contains computations that must be performed at the end of simulation.

TOPSIM provides a wide selection of blocks that are often used in the simulation of communication systems. The library contains about 160 different blocks that allow the simulation of communication equipment, and about 21 performance evaluation subprograms that can perform on-line processing of signal samples. However, performance measures are derived off-line using signal samples stored in the mass storage device. Apart from the most commonly used blocks, or classes of blocks like linear filters, modulators and demodulators, carrier and symbol synchronizers, etc., TOPSIM III has many other specialized blocks, such as bandpass nonlinearities, voltage controlled oscillators, adaptive linear equalizers, and bit scramblers. TOPSIM allows user to build a private ad-hoc library using standard blocks and by writing new compatible software in FORTRAN. The general purpose blocks in the library e.g., signal adders, multipliers, delay elements, phase shifters, rectifiers, envelop detectors, etc. can be used to build specialized blocks not included in the library.

A performance evaluation library in TOPSIM provides facilities like error probability, power spectra, SNR, eye pattern, scattering diagram, histograms, etc. The ability to store samples of signals on a mass storage device for later retrieval allows for versatile postprocessing capabilities.

The first two versions of TOPSIM were embedded into IBM's continous system modelling program (CSMP III). TOPSIM III is instead a stand-alone FORTRAN program and can be installed on a wide range of different computers. Currently it is in use at a number of European installations.

Some additional features that are being implemented in the TOPSIM package include multirate sampling, block processing, and improved input/output interfacing.

While the main advantage of TOPSIM over other packages is its highly portable nature, it has drawbacks similar to SYSTID including lack of on-line documentation and help, and a considerable amount of burden placed on the user when it comes to programming and bookkeeping details.

C. ICS and WCS [4,5]:

An accurate simulation of most modern communication systems of even moderate complexity can be very time consuming if executed on general purpose machines 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. Researchers therefore, tried to develop a hardware configuration where an array processor could be employed as a floating point processor in conjunction with a general purpose host computer providing a unique opportunity for accurate and statistically meaningful digital simulation at a very high speed. This initial development effort, under the support of the U.S. Air Force Rome Air Development Center, resulted in the Interactive Communication System simulator (ICS) as a graphics oriented, highly inter-active hardware/software system consisting of a typical minicomputer acting as a host to a peripheral array processor (AP-120B) manufactured by Floating Point Systems, Inc.

In order to minimize the programming effort on the part of the user, ICS uses an interactive menu-driven approach. It assumes a fixed topology for the digital system being simulated, consisting of a serial connection of a source, encoder, modulator, channel, demodulator, decoder and sink. Using menus the user can select types of modulators, filters, etc., and set parameter values for each of the functional blocks. By using a fixed

topology and by implementing a large portion of the code in assembly language on an array processor, ICS trades off flexibility and portability for processing speed.

The ICS has been designed to be used in four distinct modes of operation: design mode, validation mode, simulation mode and analysis mode. In the design mode the user assembles a simulation model in building block fashion from basic function modules. Specifically, the block diagram of a general communication system is displayed on a graphics CRT. Using the light pen, the user selects successive blocks where upon 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 for a complete end-to-end communication system.

In the validation mode the user is allowed to interactively exercise a simulation model developed in the design session in order to verify the model before a commitment to time consuming simulation. In this phase the user is allowed to view the time waveform and/or frequency domain quantities which enable the user to verify the end-to-end behavior of the model or input/output behavior of any specific submodule. If modification or other changes are required, the user must leave the validation stage and re-enter the design mode.

In the simulation mode, an executable design file corresponding to a simulation model is identified together with appropriate parameters. Appropriate data logging and bookkeeping software are provided to allow tabulation of bit error histories and evaluation of bit error probabilities as a function of E_b/N_o in the simulation mode. Statistical files are created for generating histograms of other parameters during a simulation session. Analysis, the last and the final stage, provides the graphical display interpretation of results obtained during a typical simulation session.

While ICS has proven useful as both a research and an educational tool, the following factors motivated the search for a more effective cost/performance solution to communication system simulation:

(1) The hardware configuration required to support the ICS is expensive and difficult to justify for a dedicated simulation facility.

(2) The software is not readily portable to other host configurations.

(3) There does not exist a rich programming environment for efficient development of AP 120B assembly language code for new signal processing modules.

The Workstation Communication System (WCS) for interactive simulation of point-to-point digital communication links has provided solutions to many of the above problems. The WCS system, which is based on Fortran-77, is hosted on an IBM PC/AT or compatible microcomputer. At the heart of the system is a single board floating point array processor offering peak potential throughputs as high as 20 million floating point operations per second(MFLOPS) - a considerable improvement over the large minicomputer hosted array processor such as AP-120B with 12 MFLOPS. This advanced processor called the VORTEX, manufactured by SKY Computer Corporation, offers a unique method of interfacing to a PC. This hardware configuration provides a very attractive cost/performance alternative. These workstations come with standardized operating systems, together with device-independant drivers, resulting in highly portable software. Furthermore, these workstations provide a rich program development environment allowing relatively easy addition of new software modules.

The basic system model and functional capabilities of WCS are similar to ICS. Some enhancement in WCS over the basic ICS include capabilities to simulate trellis-coded modulation/demodulation, diversity channels, spread spectrum systems, and ECM channels. The interactive graphical capabilities of the original ICS have been significantly

enhanced by improved graphics hardware and software available for the IBM PC/AT computers.

D. MODEM [6,7]:

MODEM was developed at Motorola's Communication Research facility as a fixed topology simulation program written in PASCAL under the DOS operating system. Previous versions were written in BASIC for the Atari-800 and HP 9816. The current program requires an IBM PC or compatible machine with 512 K memory with a math co-processor. MODEM is capable of simulating digital communication systems using various quadrature modulation schemes.

The program consists of a series of simulation and support subroutines tied together by a short main program. Each of the analysis subroutines (code generator, encoder, modulator, filter etc.) simulates one of the functional elements of the system. Using menus provided by the program, the user can select different types of modulation formats, filters etc., and set parameters such as bandwidth. Again, each subroutine provides its own menu for operation, and when an operation is completed, control returns to the menu. From here the user can either stay in the subroutine for additional operations or return to the main menu. Simulation outputs are generated by accessing the support routines which are resident in the main memory at all times. These in turn access graphics routines which generate the actual display. Eye diagrams, envelopes, phasor diagrams, and bit error curves can be generated and displayed at any point.

MODEM allows user generated programs that represent analysis routines and models not included in the basic program. All of the routines resident and all of the signals

generated within MODEM are available to user defined programs. A compatible PASCAL compiler is required in order to develop the object code for user-supplied models. Additional features include the ability to quickly obtain a series of communication system performance characteristics as a function of parameters such as filter bandwidth, the degree of nonlinearity in the power amplifier, or the level of adjacent channel interference.

The MODEM program has proven to be a valuable aid for designing and understanding communication systems. It is currently in use at several universities, industries, and government agencies.

E. SOME OTHER PACKAGES [8,9,10]:

Along with ICS, the Interactive Communication System Simulation Model (ICSSM) software package was developed under contract let by the Rome Air Development Center of the US Air Force in the mid 1970's. ICSSM permits the connection of functional blocks in any desired topological interconnection. This free topology, which may complicate the simulation software structure, provides maximum flexibility. Whereas SYSTID, TOPSIM, and ICS are time driven simulators where processing is initiated at every "tick" of the simulation clock, ICSSM is an event driven simulator where processing takes place only when an event of interest (e.g., an application module output signal to be recorded in the exercisor result file, or simply the simulation termination event) takes place. Its flexibility and event driven capability permits a wider variety of systems to be simulated. Usefulness of ICSSM is also highlighted by its very good model library. It also allows the user to develop his own model. However, creating and entering ICSSM library modules require a considerable amount of programming background on the part of the user.

Block Simulator (BLOSIM) is another general purpose simulation program for sampled data systems. It is a time driven simulator, and hence is efficient for simulating systems which operate on data at regular intervals. BLOSIM itself does not include internal models for any particular system to be simulated; these must be provided by the user. Thus, complete generality is maintained. However, it makes BLOSIM use similar to writing a special purpose simulation program for the system.

The Communication Link Analysis and Simulation System (CLASS) is an integrated set of FORTRAN programs capable of predicting the compatibility and performance of the communication and tracking links for all services and signal formats supported by the Tracking and Data Relay Satellite System (TDRSS). CLASS contains the detailed models for the TDRSS spacecraft and ground terminal hardware and the effect of the transmission medium (rain attenuation, multipath radio frequency interference, etc.). It uses a combination of waveform level and formula based analyses for evaluating the performance of satellite communication links.

2.2. LabVIEW SOFTWARE SYSTEM

The Laboratory Virtual Instrument Engineering Workbench (LabVIEW) software system, developed by National Instruments has been designed to harness all the power and features of traditional programming languages in a new friendly environment. Replacing lines of cryptic code with intuitive graphic objects, LabVIEW has simplified scientific programming to a great extent. It features a full-function graphical programming system which provides all of the tools needed for data acquisition, data reduction, data analysis, data management, and data presentation. As LabVIEW is a full-structured programming system it can be used standalone for general purpose scientific computation such as

simulation and modeling as well as with instrumentation. LabVIEW provides an ideal mechanism for controlling remote instruments with reusable virtual instruments. LabVIEW runs on the Apple Macintosh family of computers with at least four megabytes of memory and a hard disk.

The core concept around which LabVIEW has been developed is the Virtual Instrument (VI). A VI is simply a software program which behaves in a manner similar to physical instruments in modularity and function. In LabVIEW, VIs are used to develop applications. VIs can control physical instruments, perform computations, simulate physical instruments, and can do all kind of jobs that can be done with conventional language programming techniques. VIs maintain a hierarchical structure. Each VI can be used as a sub-VI while building higher level VIs. Thus, VIs can be used interactively from their front panels or programmatically in the block diagram of a higher level VI. The ability of a VI to be used as a sub-VI is a powerful feature of LabVIEW. The sub-VI combines the benefits of subroutines: reusable code, code sharing, separation of tasks, etc., with the benefits of VI themselves: graphical user interface, interactive execution, execution highlighting, etc.

A VI has three main parts: front panel, block diagram, and icon/connector. The front panel defines the input and output and provides a mechanism for interactive operation. Block diagrams, essentially the source code, specify the flow of control and data. The icon is a representation of the VI, and the connector acts as a port through which data passes from the block diagram to the VI.

To program in LabVIEW means to draw the block diagram. A block diagram is a natural first step taken by engineers and scientists when they design applications. Normally the data flow block diagrams need to be converted to a control flow program written in conventional languages. However, in LabVIEW the initial block diagrams are

themselves executable programs. The flow of control and data specified in the block diagram is implemented in a graphical programming language named G. G uses dataflow concepts, traditional program control structures, and an instruction set consisting of graphical elements. Thus the block diagram is the VI program and may be executed at any time.

The block diagrams are created by selecting and arranging graphical objects using menus and connecting them together. Program objects are categorized as nodes, terminals, and wires. Nodes are the program execution elements. Nodes are analogous to statements, operators, functions, and subroutines in conventional languages. The five types of nodes are functions, sub-VI nodes, structures, code interface nodes, and formula nodes. Functions are elementary nodes which perform operations like adding numbers, file I/O, and string formatting. A sub-VI node is a reference to a sub-VI, analogous to a subroutine call statement. Code interface nodes are interfaces between the block diagram and user supplied code written in conventional languages such as C or Pascal. Structures supplement LabVIEW's dataflow architecture to control execution order. The formula nodes supplement the functions by giving a convenient alternative method to execute computations.

Terminals are data ports through which data passes between the block diagram and the front panel and between nodes in the block diagram. Terminals are analogous to operands, parameters, arguments, and constants. Wires are the data paths between terminals. Wires are analogous to variables.

In LabVIEW, the front panel is usually designed before drawing the block diagram. The front panel of a VI is built with combinations of controls and indicators. Controls simulate an instrument's input device and are the means of supplying data to the VI. Indicators simulate an instrument output device that display data generated by the VI.

When the VI operates as a sub-VI inside another VI, the controls and indicators receive data from and return data to the calling VI. Selection over a wide variety of types of controls and indicators can be made from the controls menu. Major types include numeric, boolean, string, arrays and clusters, charts, and graphs. Some of the objects available from the controls menu are offered as controls only, while others as indicators only. However, some are offered in both configurations. As soon as the required controls and indicators are chosen, the corresponding terminals are automatically provided in the diagram window.

There are several benefits of using LabVIEW. To mention a few: 1) LabVIEW makes the block diagram itself an executable program. Thus the requirement of converting the dataflow diagrams to control flow programs to be written in conventional languages is eliminated. This reduces the labor, time consumed, and error in developing applications; 2) LabVIEW VIs are reusable as any instrument itself; therefore, as each new one is added to the library the cost of future projects is lowered; 3) LabVIEW can increase the performance of instruments by increasing its functionality. It can turn less sophisticated instruments into more sophisticated ones; 4) As block diagrams are used universally to explain textual descriptions, LabVIEW programs are self explanatory to all users.

CHAPTER 3

ANALOG SYSTEM SIMULATION

Systems are determined by the type of the modulation used. Modulation can be defined as the process by which some characteristics of a carrier is varied in accordance with a modulating wave. The baseband signal is referred to as the modulating wave, and the result of the modulation process is referred to as modulated wave. In analog systems the modulating wave consists of an analog signal (e.g. voice signal, video signal), and the carrier consists of a sine wave. Depending upon whether the amplitude, phase or frequency of the sinusoidal wave is varied, analog systems are classified.

The choice of a modulation technique is determined by the characteristics of the message signal, characteristics of the channel, the performance desired from the overall communication system, the use to be made of the transmitted data, and economic factors. In the following pages different modulating schemes have been discussed and programs for simulation of those schemes have been developed using LabVIEW software.

3.1 DOUBLE SIDEBAND - SUPPRESSED CARRIER (DSB-SC) SYSTEM

Assuming the carrier to be sinusoidal, a general linear modulated carrier can be represented by

$$x_c(t) = A(t)\cos(\omega_c t) \qquad (3.1)$$

where ω_C is the carrier frequency and $A(t)$ is the amplitude of the carrier. For the DSB case $A(t)$ will be proportional to the message signal, $m(t)$. Hence, the output of a DSB modulator can be represented by

$$x_c(t) = A_c m(t)\cos(\omega_c t) \qquad (3.2)$$

The above expression illustrates that DSB modulation is essentially a process of multiplication of the carrier, $A_C cos(\omega_C t)$, by the message signal, $m(t)$ as shown in Figure 3.1. In the receiver, if the received signal is multiplied by the reinserted carrier, that has the same frequency as the original carrier and is phase-coherent with it, the result will be

$$\begin{aligned} y(t) &= [A_C m(t) cos(\omega_C t)\,]2cos(\omega_C t) \\ &= A_C m(t)\,[2cos^2(\omega_C t)\,] \\ &= A_C m(t) + A_C m(t) cos(2\omega_C t) \end{aligned} \qquad (3.3)$$

It can be visualized from (3.3) that if the output of the multiplier is passed through a low pass filter the second term in the expression can be eliminated. Further amplitude scaling will give the original message signal.

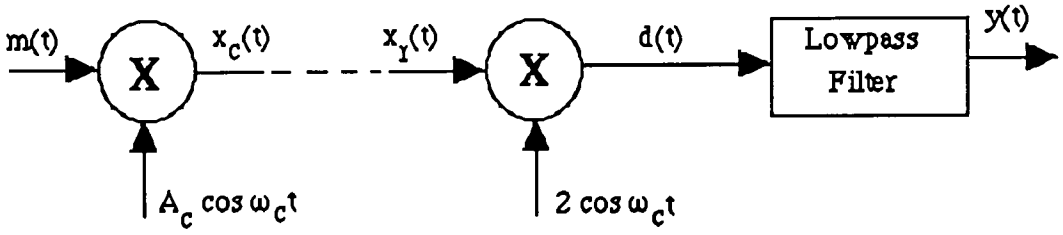
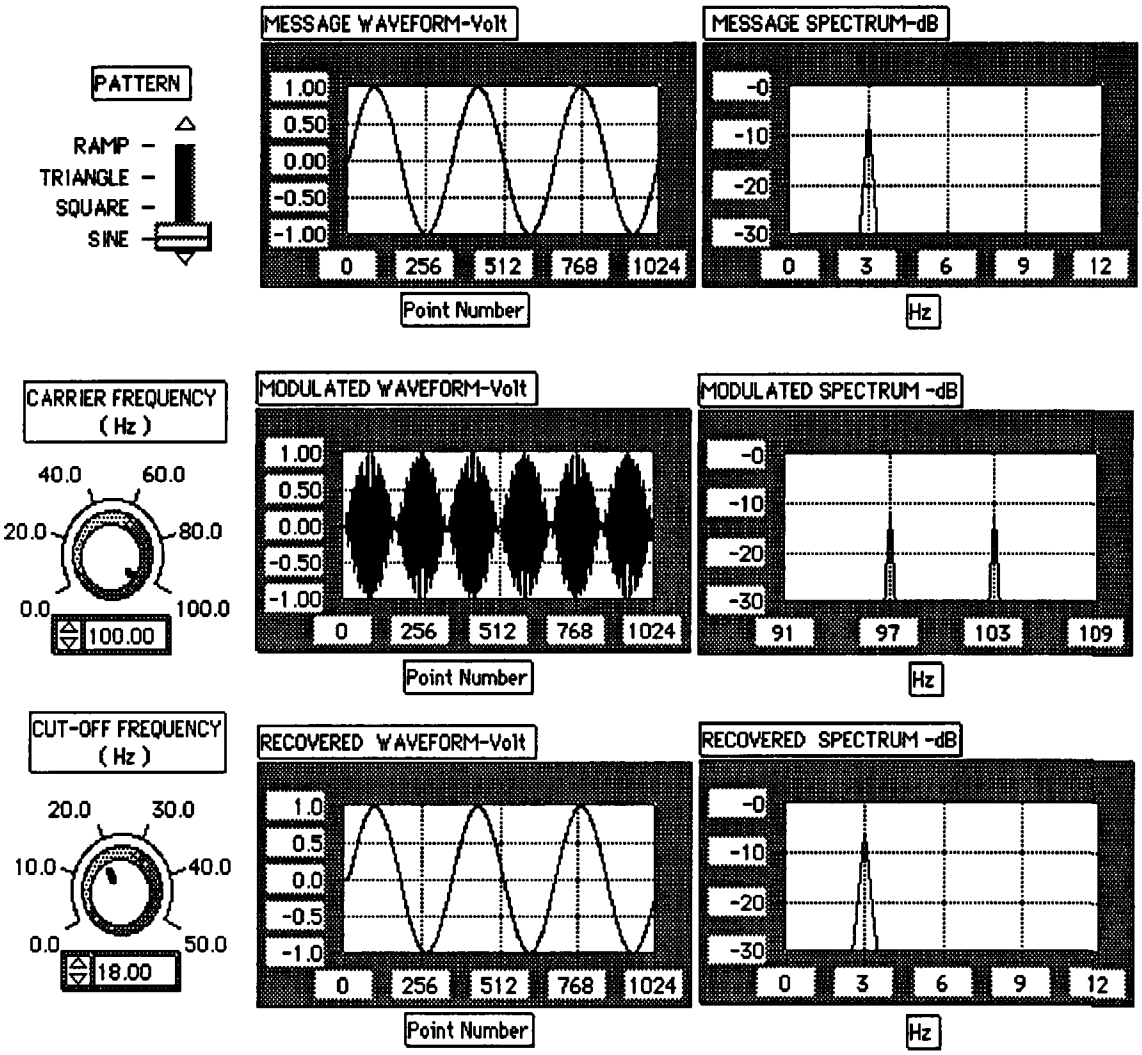


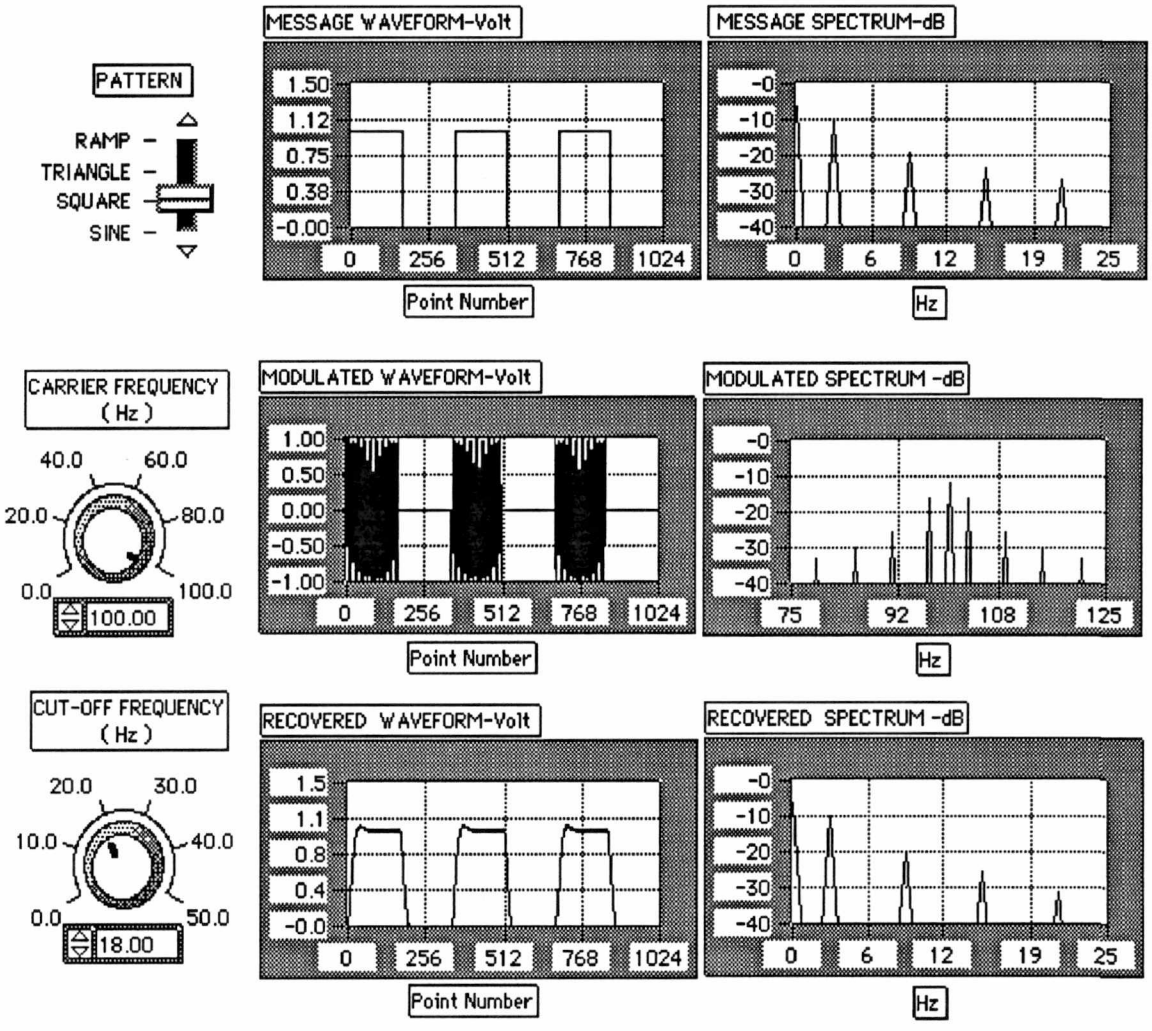
Figure 3.1 Schematic representation of DSB-SC Modulation and Demodulation

For simulation of the above process using LabVIEW, the front panel and block diagram of the DSB-SC VI is shown in Figure 3.2 and Figure 3.3 respectively. In the front panel the controls "Carrier Frequency Control" and "Cut-off Frequency Control" allow variation of the carrier frequency and cut-off frequency of the lowpass filter (LPF). With "Pattern Control" the type of message waveform (sinusoidal, square wave, etc.) can be selected. Indicators, all in the form of graphs, show the message waveform,



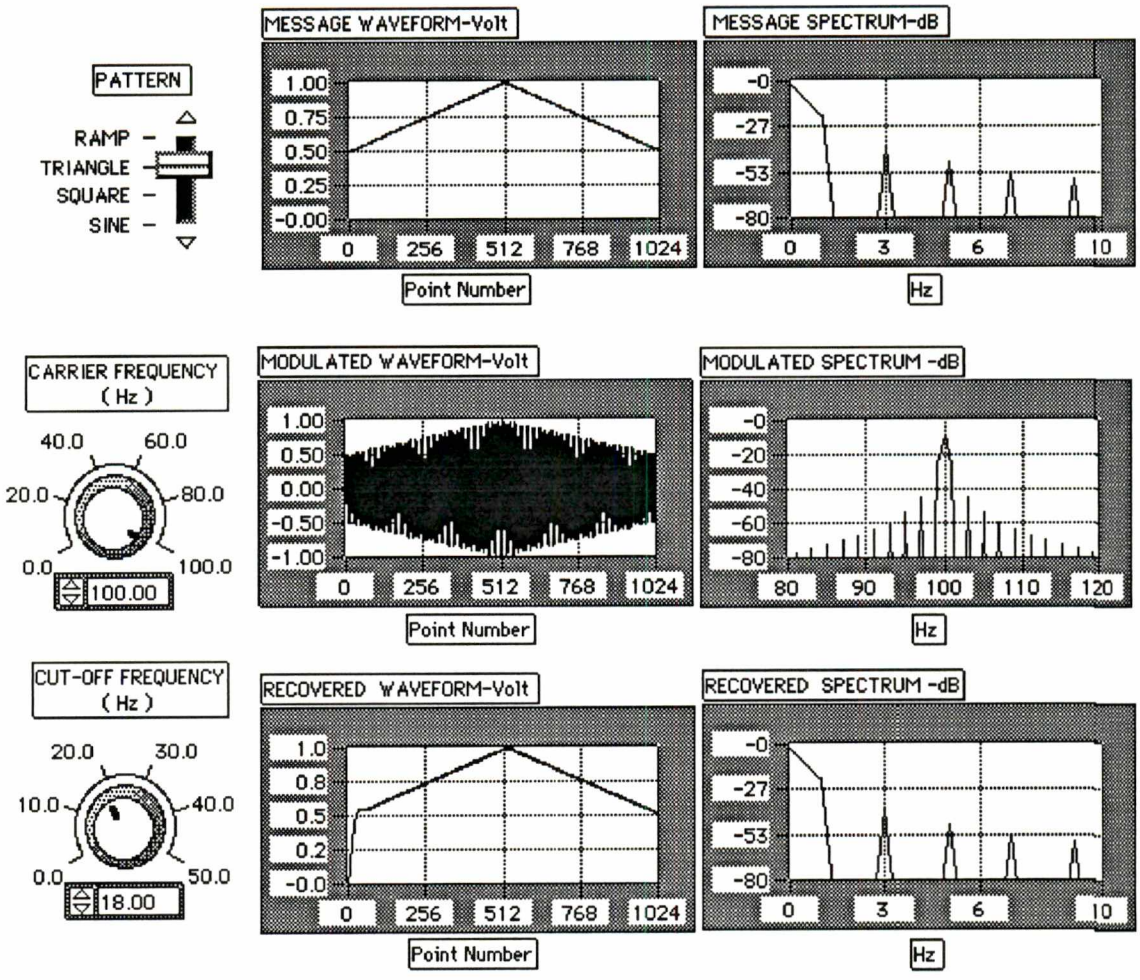
(a) Sine Pattern

Figure 3.2 Front Panel of DSB-SC System



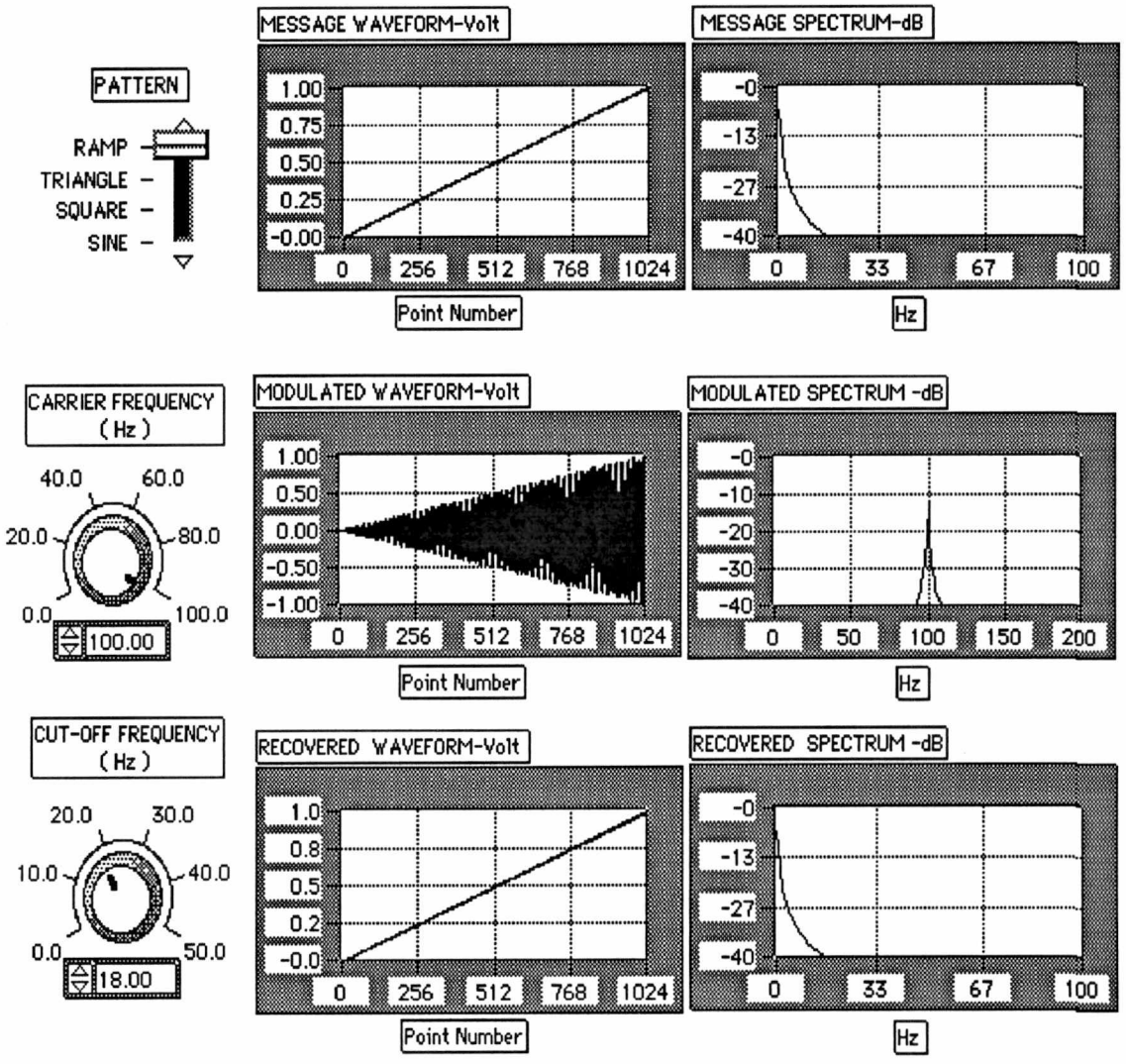
(b) Square Pattern

Figure 3.2 (Continued)



(c) Triangle Pattern

Figure 3.2 (Continued)



(d) Ramp Pattern

Figure 3.2 (Continued)

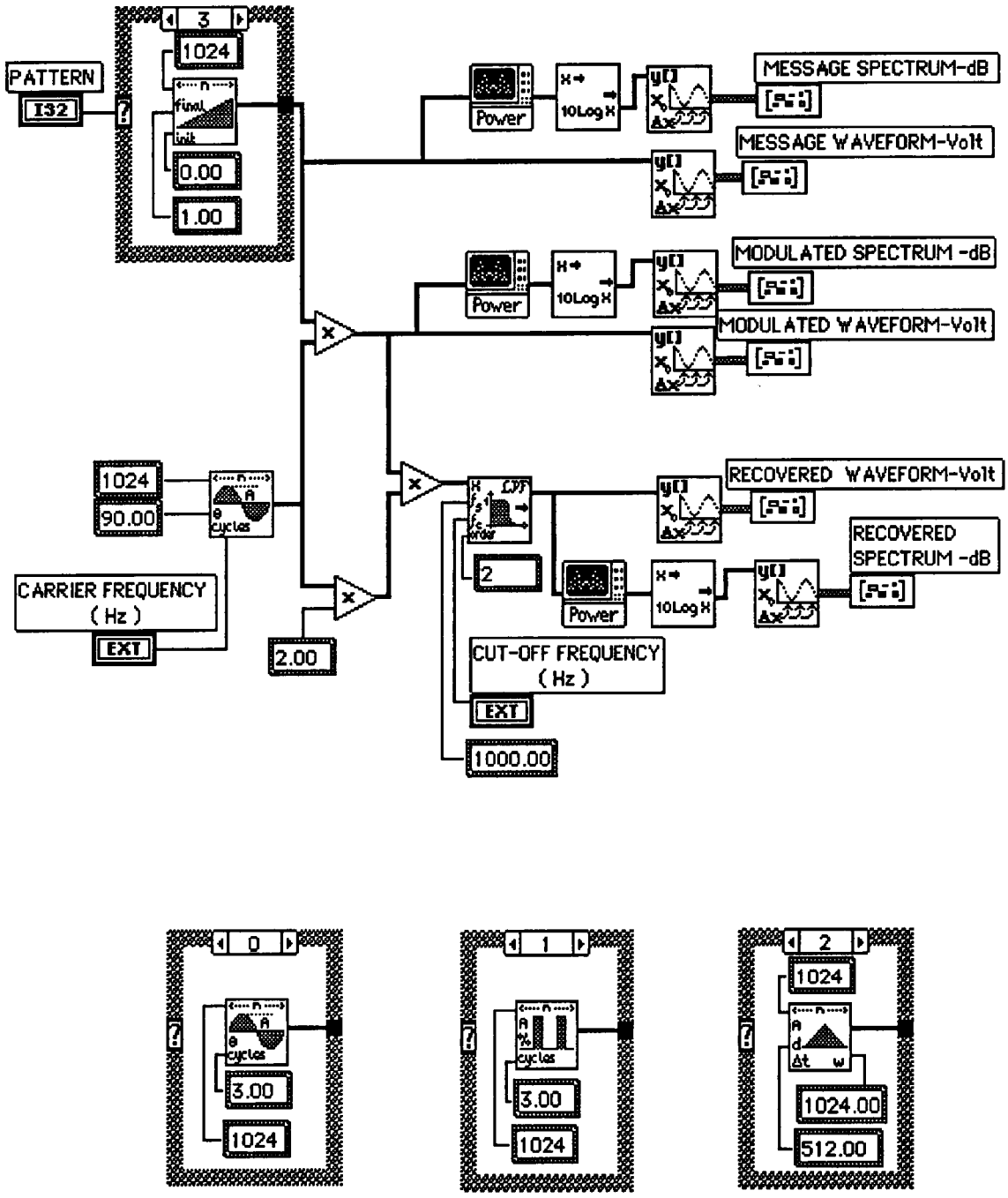


Figure 3.3 Block Diagram of DSB-SC System

modulated carrier waveform, and demodulated signal waveform as well as their power spectrums.

VIs from the LabVIEW VI library for generation of carrier as well as different types of message signals are used as sub-VIs in the block diagram. The four different types of pattern generators are placed in four different cases of a case structure. Wiring the terminal corresponding to the pattern selector control to the case selector allows the selection of a particular pattern from the front panel. Similarly the terminals corresponding to the carrier frequency control are wired to the carrier generator sub-VI so that it can be operated from the front panel. A LPF VI is selected from the LabVIEW VI library and the terminal corresponding to the cut-off frequency control is wired to it. Product operation between the message signal and the carrier as well as between the modulated carrier and the reinserted carrier are performed by using product nodes. The results are displayed on the front panel by wiring the terminals corresponding to the indicators through the array-to-graph conversion functions as shown in Figure 3.3. For obtaining the spectra, the power spectrum VI is used from the library. The recovered waveform is slightly distorted due to the transient characteristics of the digital filter used for demodulation.

3.2. AMPLITUDE MODULATION (AM) SYSTEM

Amplitude modulation results when a dc bias A, is added to the message signal, $m(t)$ prior to the modulation process (Figure 3.4). Thus for AM

$$x_c(t) = [A + m(t)]A_c cos(\omega_c t) \qquad (3.4)$$

Non-coherent demodulation such as envelope detection can be used with AM signals. However, the performance of a coherent demodulator in noise is better than a non-coherent demodulator. Coherent demodulation is performed in the same way as discussed in the case of a DSB-SC system.

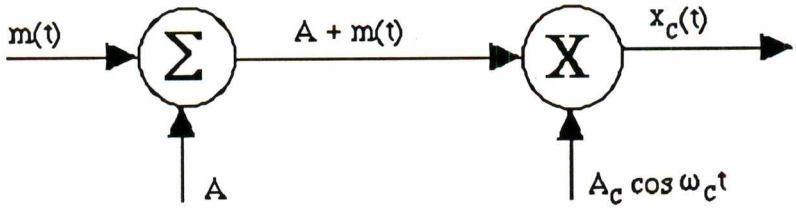


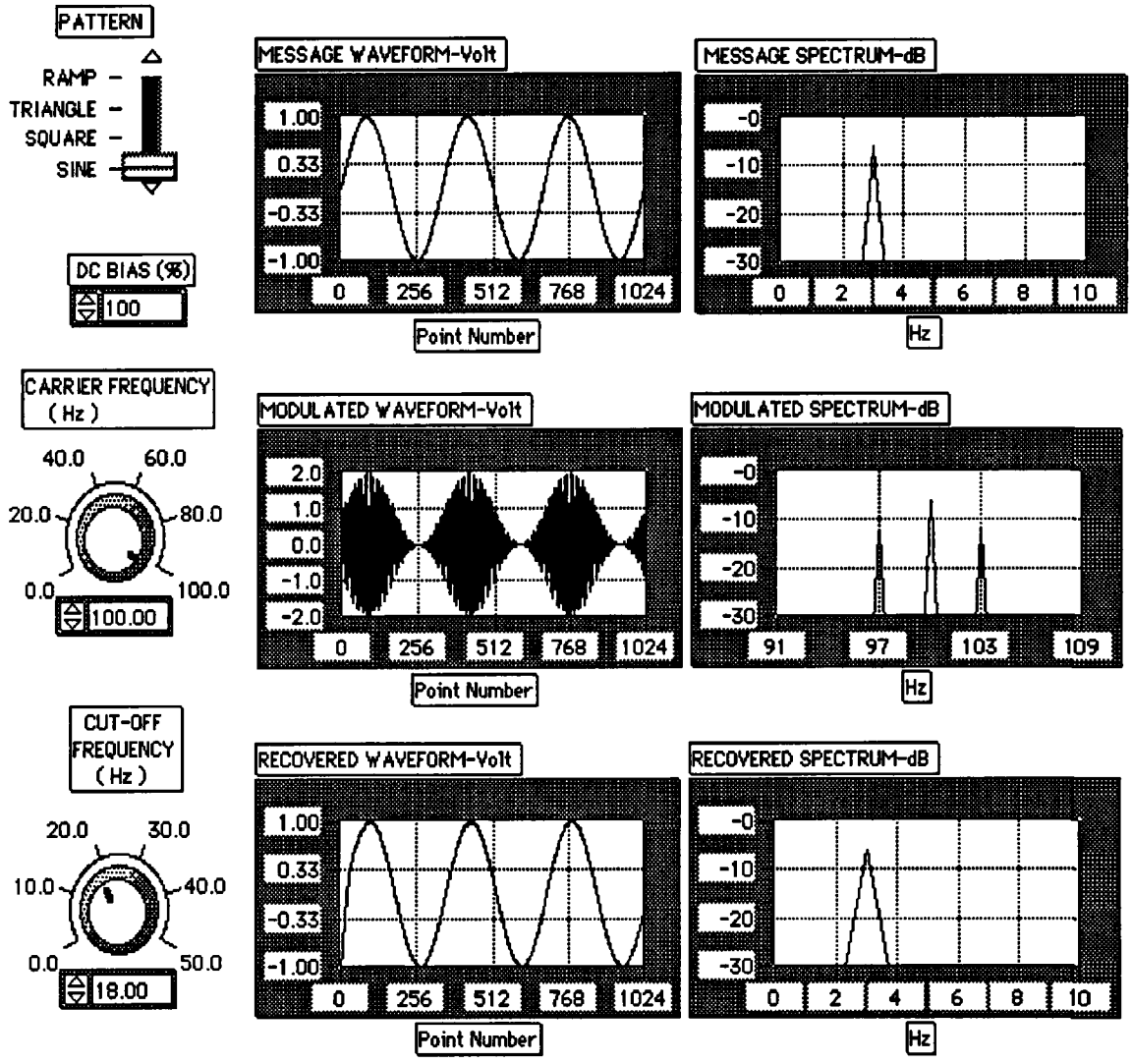
Figure 3.4 Schematic representation of Amplitude Modulation

In the front panel of the Amplitude Modulated(AM) System (Figure 3.5) the controls and indicators are almost the same as in case of DSB-SC system except for the dc offset.

In the block diagram of Figure 3.6 the dc offset in percent is first added to the message signal generated by the different pattern generators depending upon the pattern selected. The carrier wave generated by the sinusoidal wave generator is modulated by the signal with the offset in the multiply node. The waveform of the original message and the modulated carrier as well as their power spectra are displayed in the front panel by sending the signals through the array-to-graph conversion VIs. The modulated waveform is further multiplied by the reinserted carrier and passed through a LPF sub-VI obtained from the LabVIEW VI library in order to achieve the demodulated waveform. Finally, the dc offset is removed from the demodulated waveform at the subtract node. Slight distortion is noticed in the recovered waveform due to transient characteristics of the digital filter.

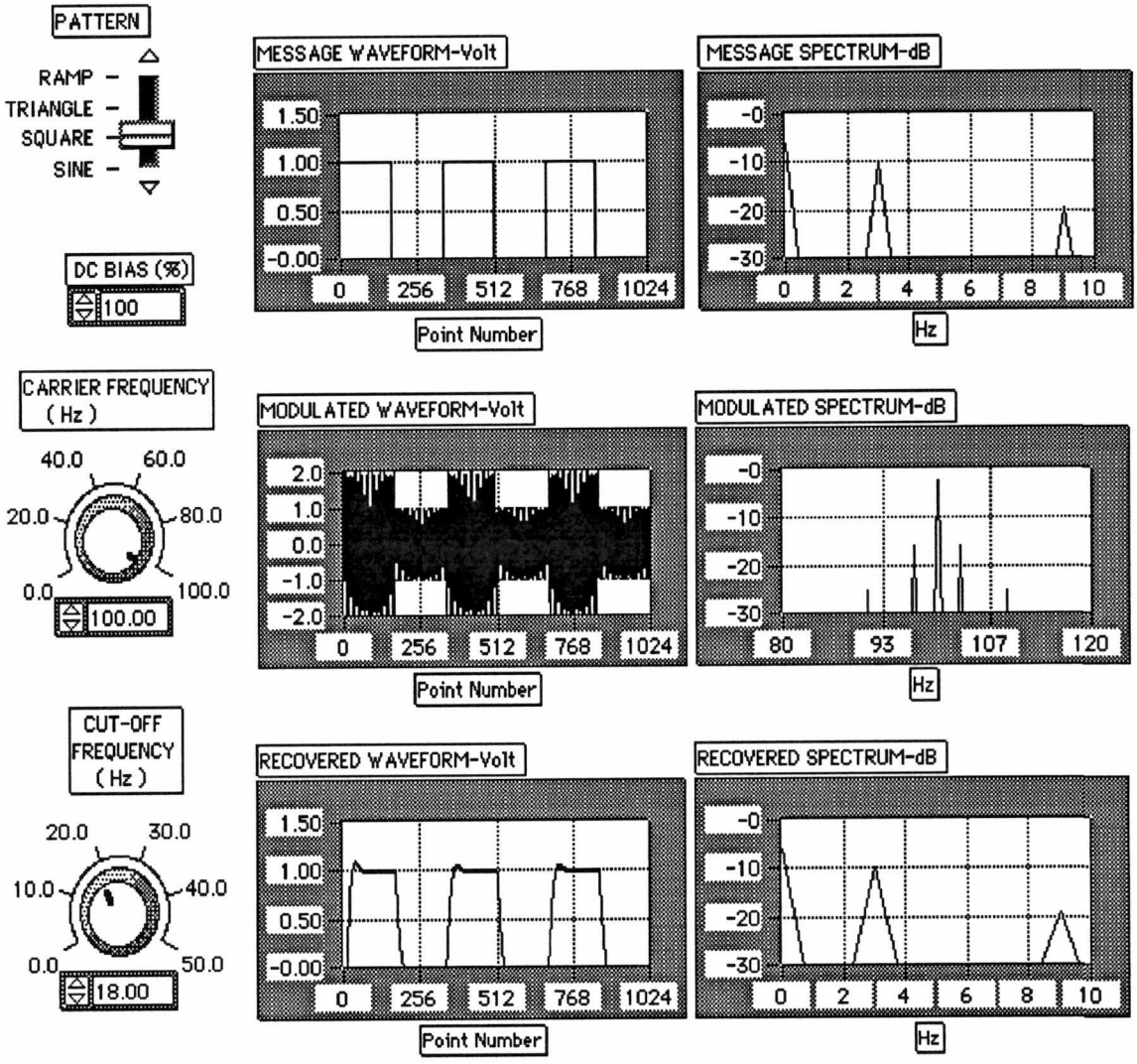
3.3 SINGLE SIDEBAND (SSB) SYSTEM

The upper sideband and the lower sideband of a DSB signal have even amplitude and odd phase symmetry about the carrier frequency. Hence, either sideband carries



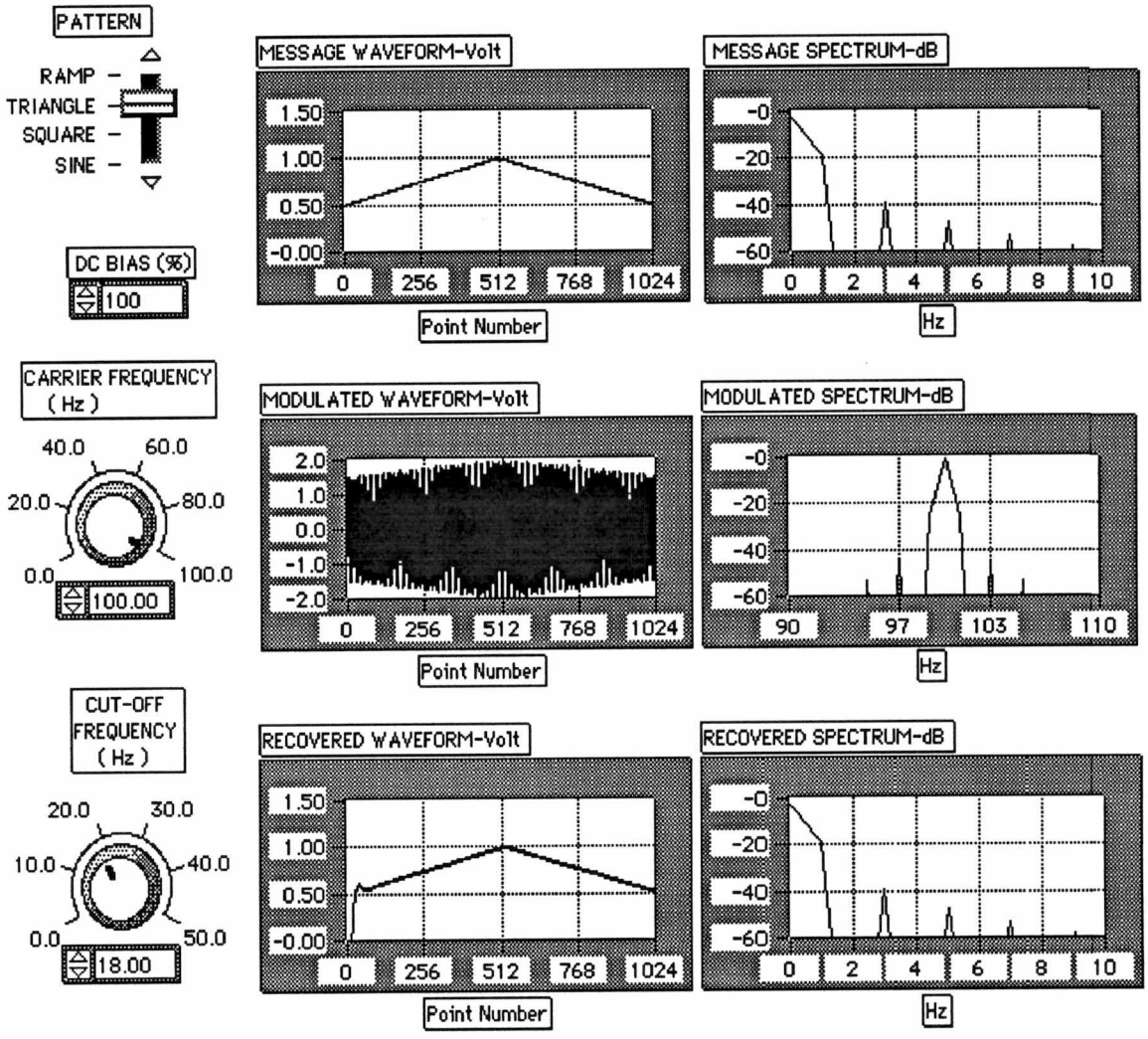
(a) Sine Pattern

Figure 3.5 Front Panel of AM System



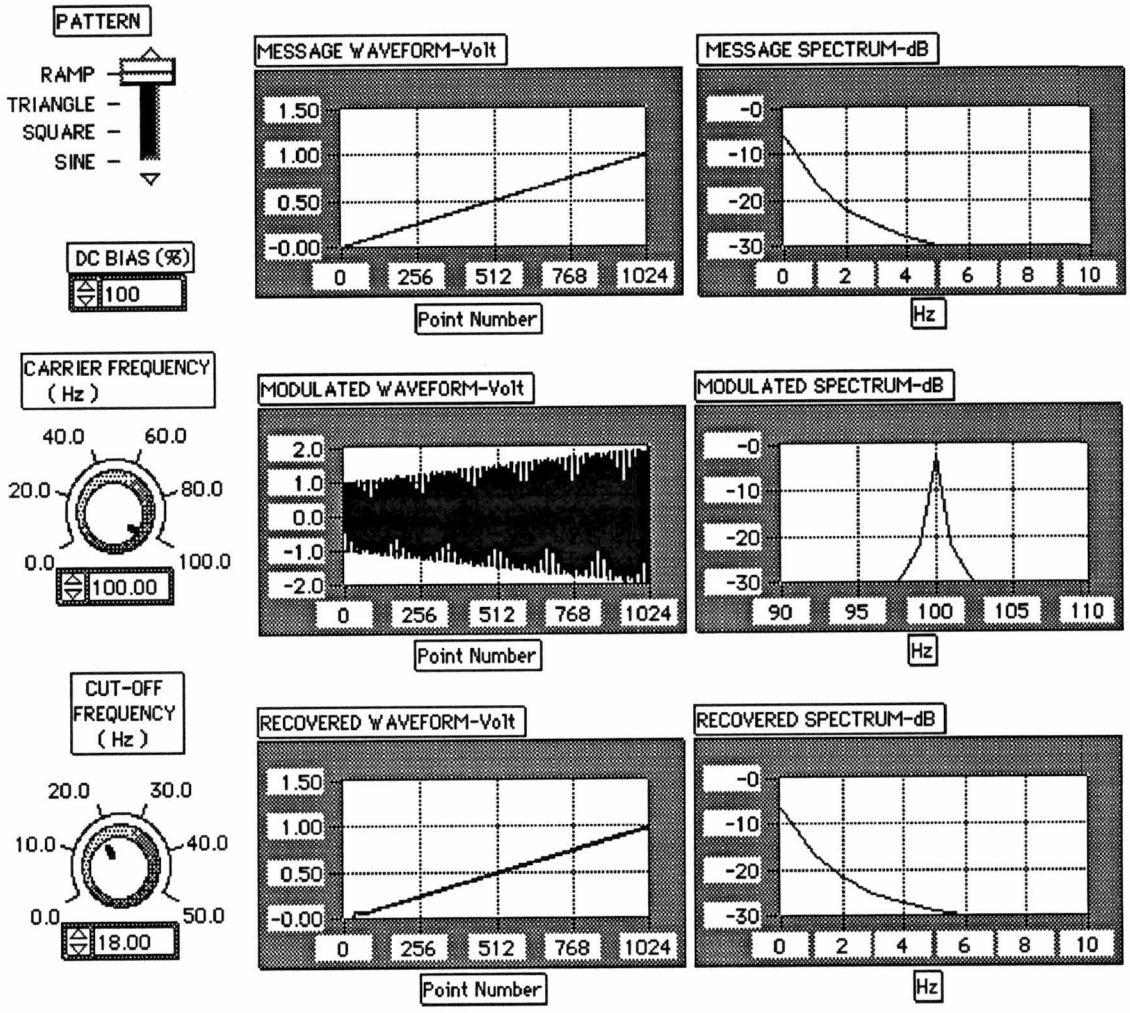
(b) Square Pattern

Figure 3.5 (Continued)



(c) Triangle Pattern

Figure 3.5 (Continued)



(d) Ramp Pattern

Figure 3.5 (Continued)

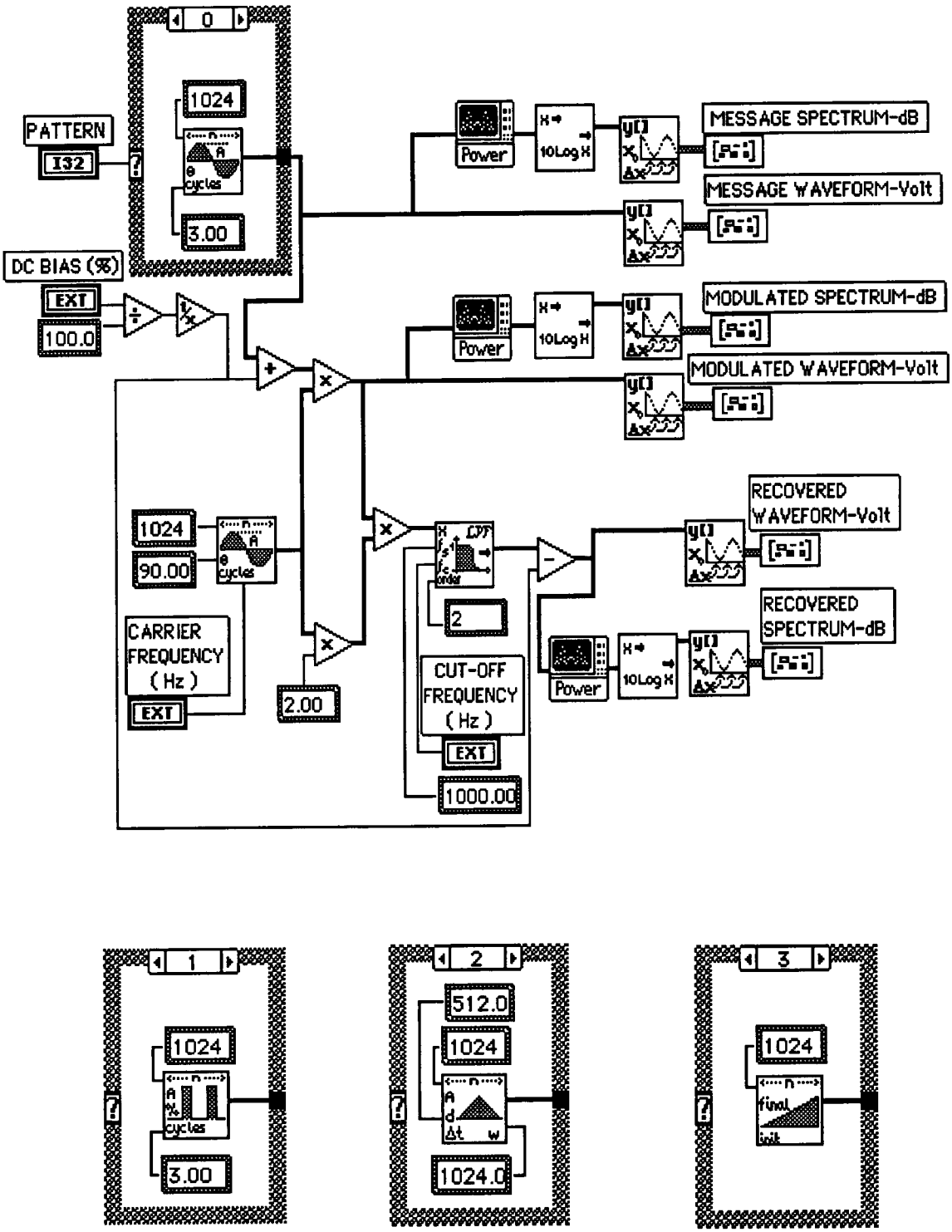


Figure 3.6 Block Diagram of AM System

sufficient information to reconstruct the message signal. Elimination of one sideband results in a SSB signal which reduces the bandwidth of the modulator output at the cost of system complexity.

SSB can be generated by sideband filtering, but this process demands very nearly ideal characteristics for the filter. As such, a much more practical method is to use phase shift modulation (Figure 3.7). This system is in fact term-by-term realization of the expression

$$x_c(t) = \frac{1}{2} A_c m(t) cos(\omega_c t) \pm A_c \hat{m}(t)\, sin(\omega_c t) \qquad (3.5)$$

where the positive sign gives the LSB and the negative sign gives the USB; $\hat{m}(t)$ represents the Hilbert Transform of $m(t)$ which corresponds to a -90° phase shift of all the signal's positive frequency components.

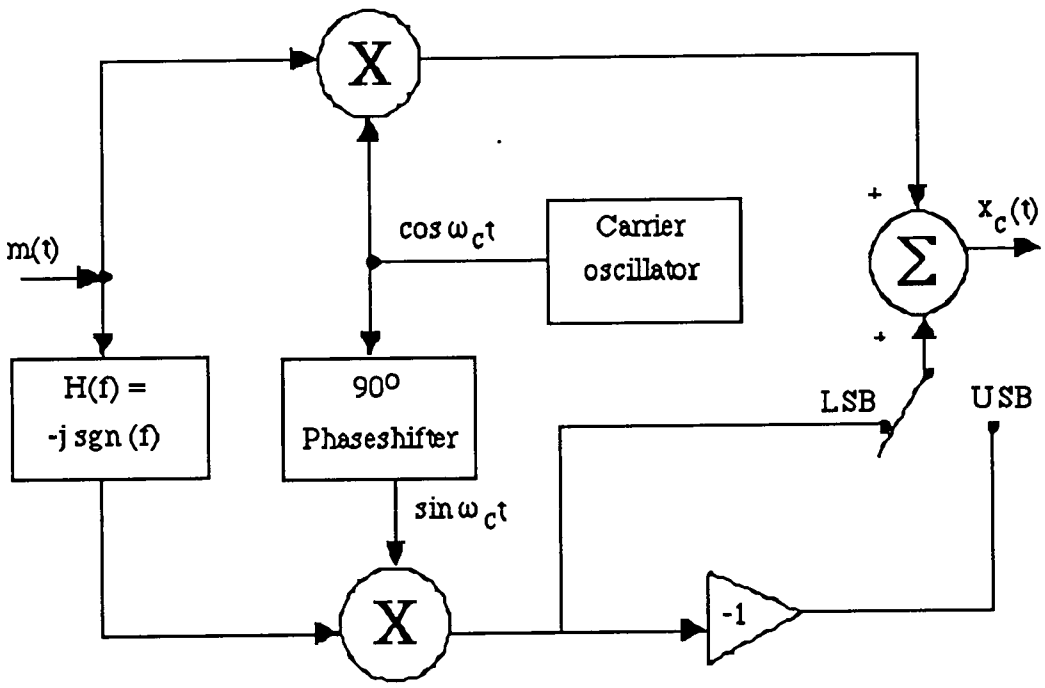


Figure 3.7 Schematic representation of Phase Shift Modulation

In the front panel of the VI (Figure 3.8) for the simulation of the phase-shift modulation scheme, a " modulation type control " is provided. The operator can choose between Lower SideBand(LSB) and Upper SideBand(USB) options. Indicators display the message waveform, modulated waveform, and demodulated waveform as in case of DSB-SC or AM systems.

The carrier is generated at a specified frequency by the sinusoidal pattern generator VI obtained from the VI library (Figure 3.9). Two simultaneous multiplications are done at two different nodes: one between the message signal and the carrier, and the other between the Hilbert Transformed version of the message and the ninety degree phase shifted version of the carrier. The results of the two cases are added in case of LSB and subtracted in case of USB. For demodulation a procedure similar to the DSB-SC case has been used.

3.4 ANGLE MODULATION SYSTEMS

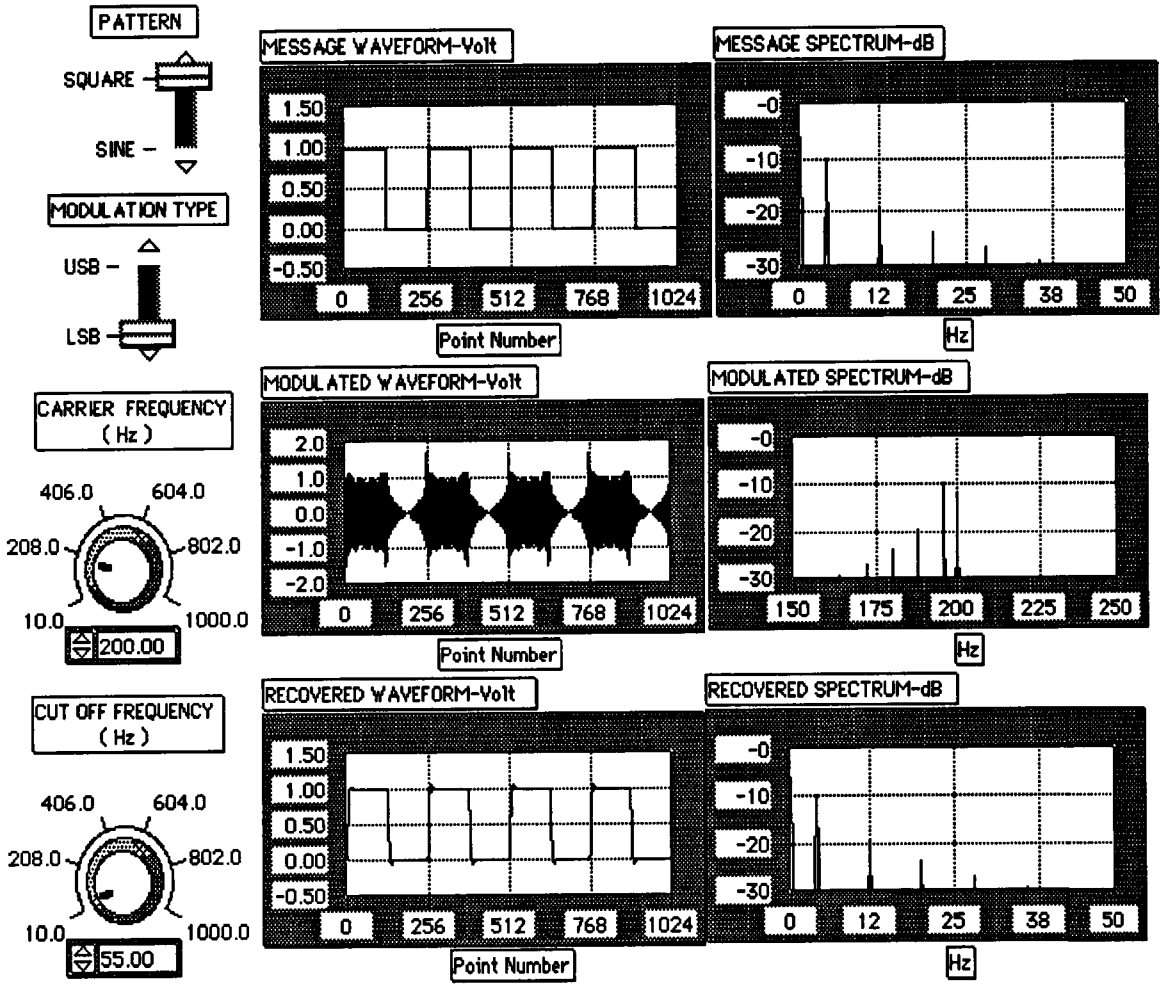
When the amplitude is kept constant and the phase or the time derivative of the phase of the carrier is varied linearly with the message signal, $m(t)$ angle modulation is obtained. Hence, the general angle modulated signal can be represented by

$$x_c(t) = A_c cos\ [\omega_c t + \phi(t)\ ] \qquad (3.6)$$

where A_c and ω_c are constants but the phase angle $\phi(t)$ is a function of the message signal

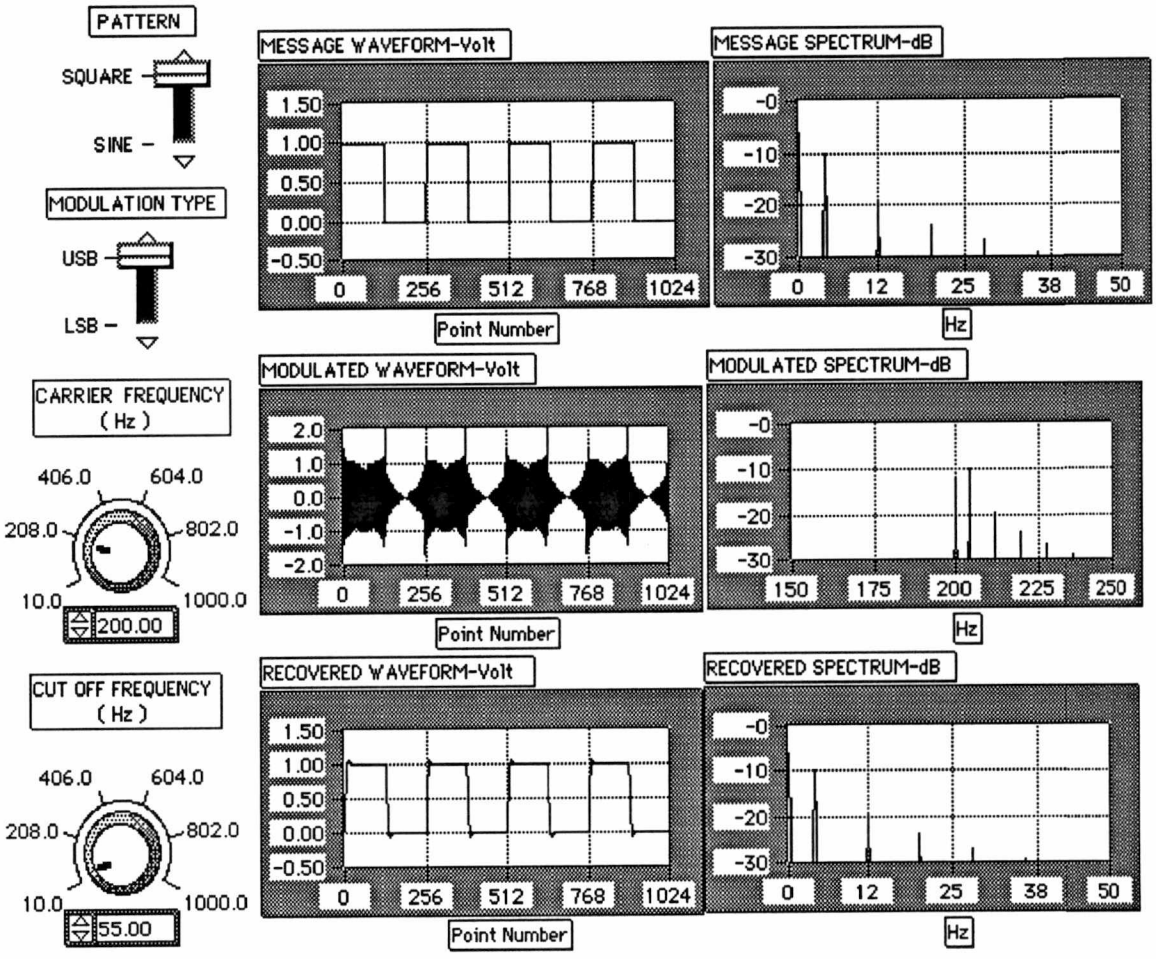
Angle modulation can be generated in two different ways. When the phase deviation of the carrier is proportional to the message signal, i.e., $\phi(t)=k_p m(t)$ where k_p is a constant, the general expression is obtained

$$x_c(t) = A_c cos\ [\omega_c t + k_p m(t)\ ] \qquad (3.7)$$



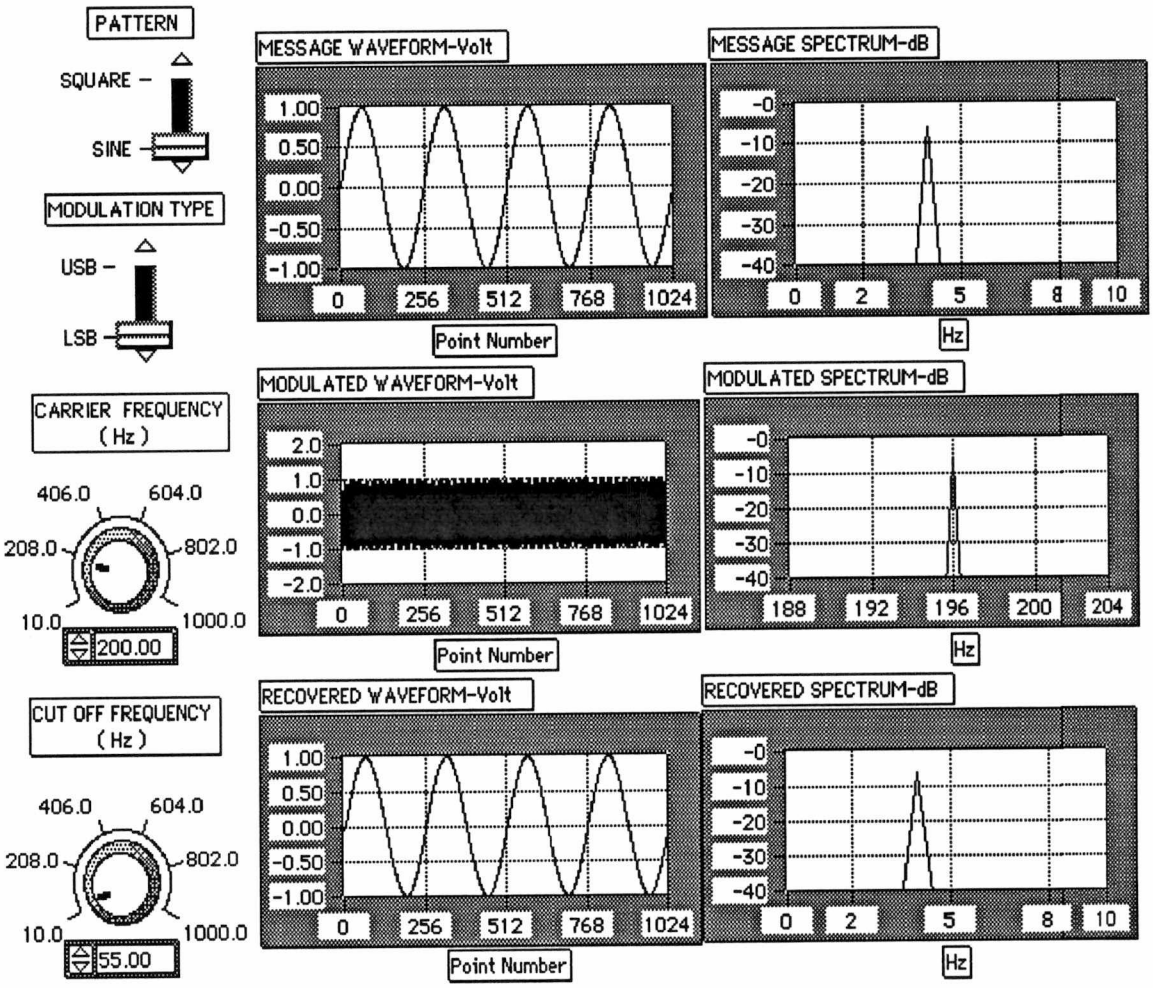
(a) LSB-Square Pattern

Figure 3.8 Front Panel of SSB System



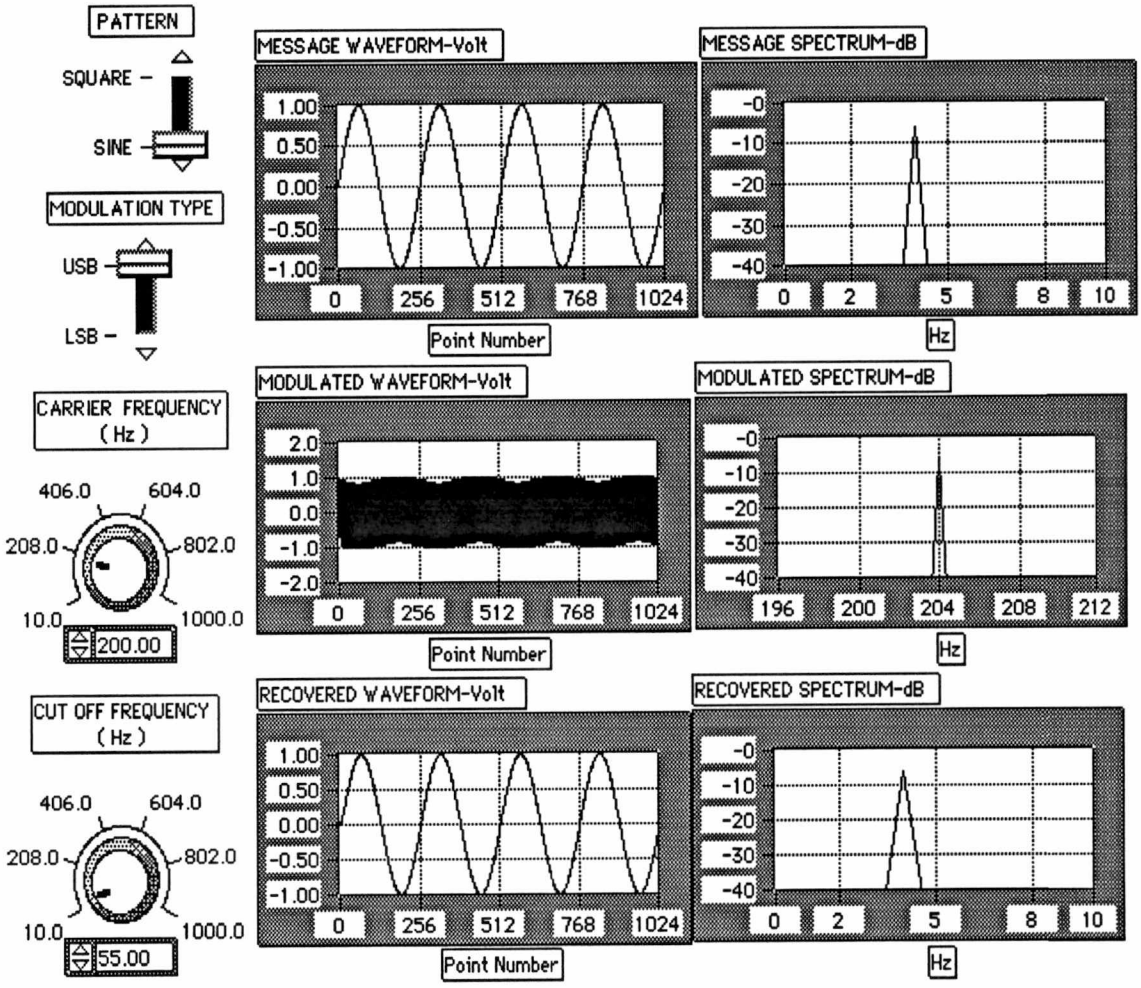
(b) USB-Square Pattern

Figure 3.8 (Continued)



(c) LSB-Sine Pattern

Figure 3.8 (Continued)



(d) USB-Sine Pattern

Figure 3.8 (Continued)

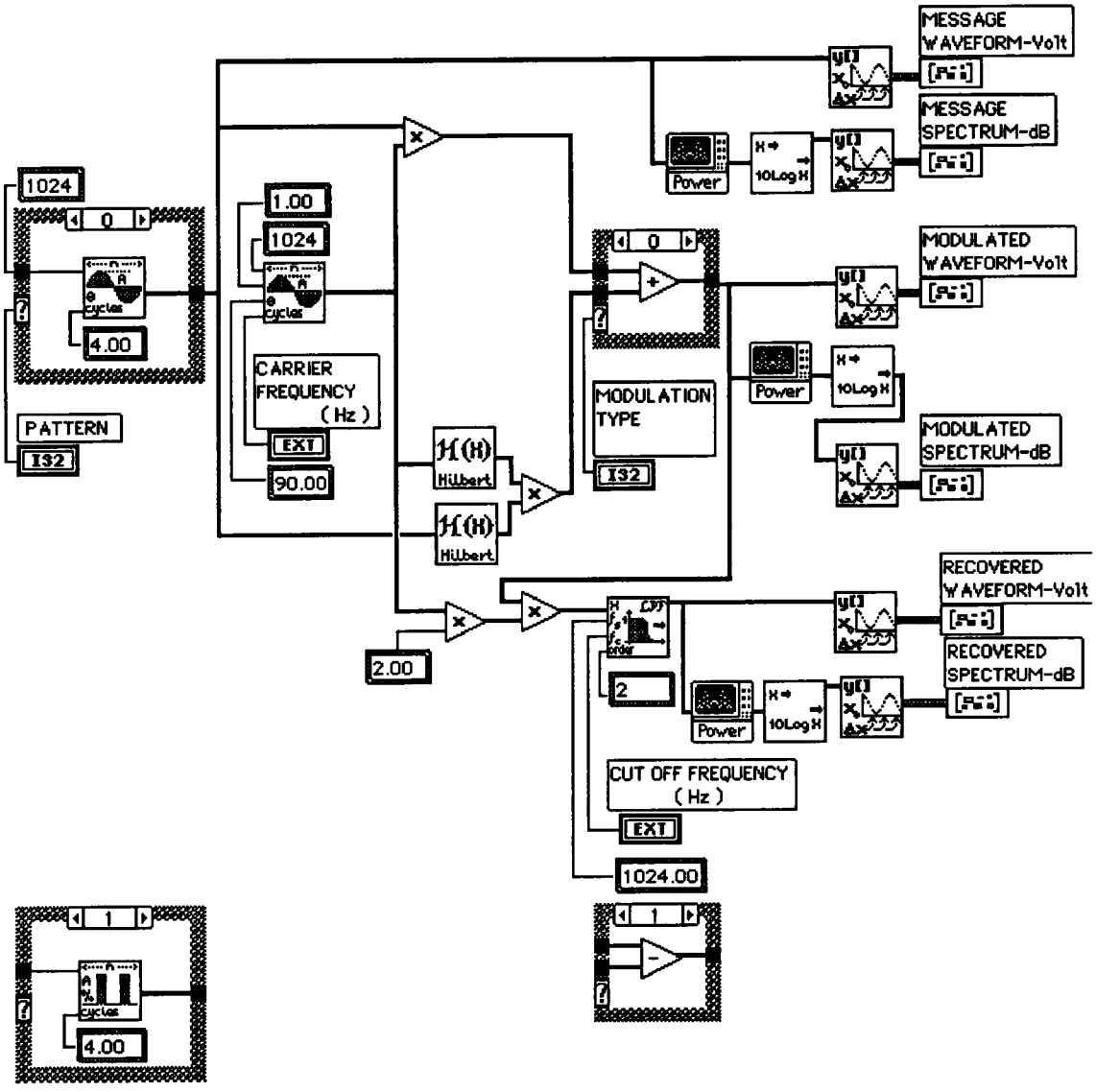


Figure 3.9 Block Diagram of SSB System

Again, when the time derivative of the phase (the frequency) is proportional to the message signal, i.e., $d\phi/dt=k_f m(t)$ where k_f is a constant, the general expression is

$$x_c(t) = A_c cos\ [\omega_c t + 2\pi f_d \int^t m(\alpha)d\alpha\ ] \qquad (3.8)$$

These two different modulation schemes are called Phase Modulation (PM) and Frequency Modulation (FM), respectively. Systems for the generation of FM and PM as shown in Figure 3.10 can be developed by the step by step realization of Equations 3.7 and 3.8 respectively. Demodulation is effected by using a simple discriminator in case of FM and a discriminator followed by an integrator in the case of PM.

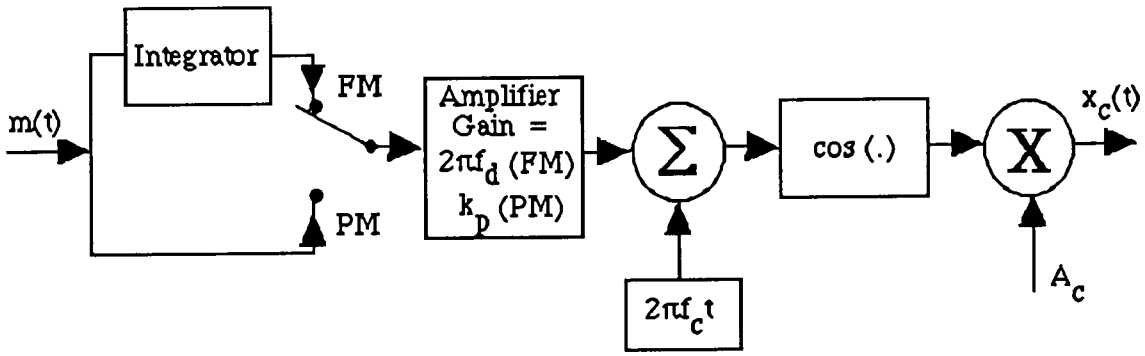


Figure 3.10 Schematic representation of Angle Modulation

For LabVIEW implementation of the FM system the message signal is generated by a sine wave generator whose characteristics are determined by the values of amplitude and frequency selected in the front panel (Figure 3.11). The message signal thus generated is integrated by passing it through the integration VI and then multiplied by a constant whose value can be changed by controlling the "Deviation Constant" in the front panel. The carrier is generated by taking the cosine of each element coming out of the "for loop"

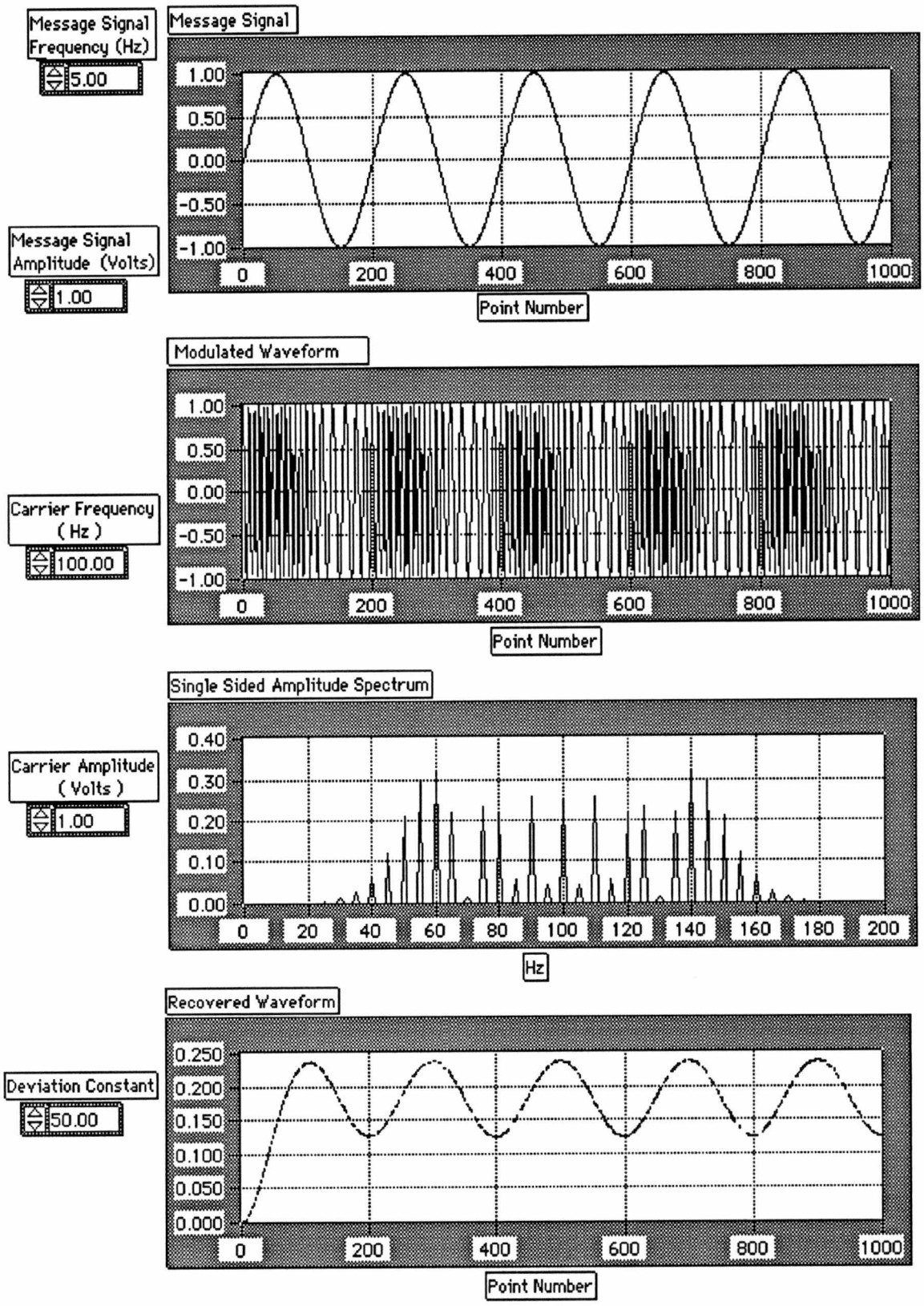


Figure 3.11 Front Panel of FM System

structure which actually generates a product of carrier frequency and continous time as shown in the block diagram (Figure 3.12). The amplitude of the carrier is determined by the value selected in the front panel. The message pattern as well as the modulated waveform are displayed in the front panel by passing the arrays representing them through the array to graph converter. The single sided amplitude spectrum is obtained by scaling the output of the power spectrum VI.

Demodulation is achieved by using a discriminator consisting of a differentiator and an envelope detector as shown in Figure 3.12. A differentiator is obtained from the LabVIEW VI library. The envelope detector is realized by using a clipper for the diode detector and a LPF as shown in the block diagram of the FM system. The recovered waveform displays the message pattern with a little distortion in the first cycle because of the transient characteristics of the digital filter. The DC offset is caused due to envelope detection.

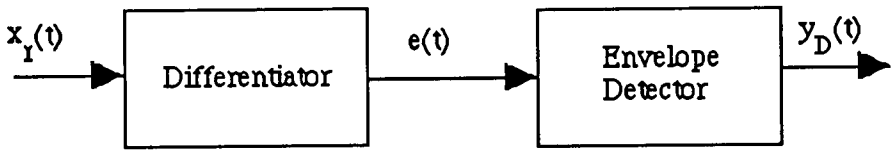


Figure 3.12 Schematic representation of Discriminator

In the case of a phase modulation system similar arrangements are made except for the elimination of the message signal integrator as shown in Figures 3.13 and 3.14. The recovered waveform shows a 90^o phase shift from the message signal. Hence, an integrator followed by the discriminator is necessary in obtaining the desired waveform.

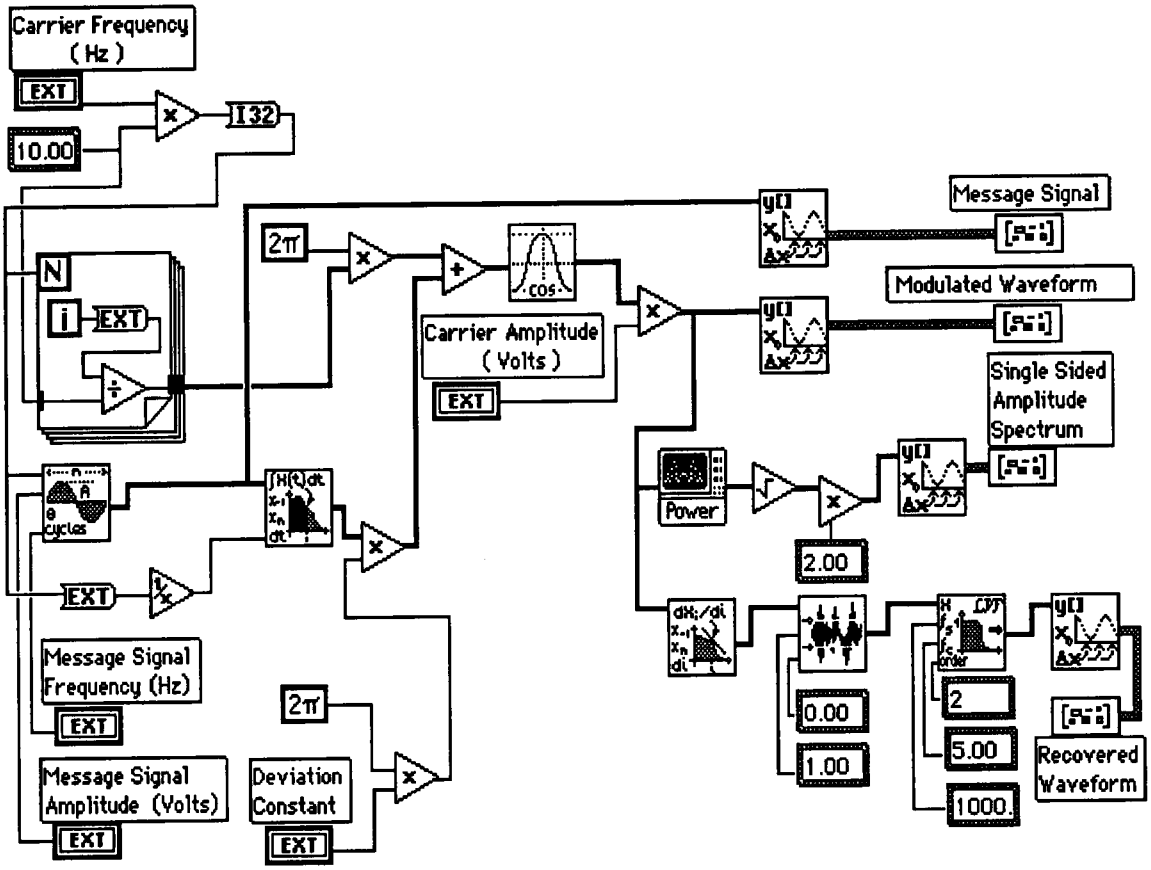


Figure 3.13 Block Diagram of FM System

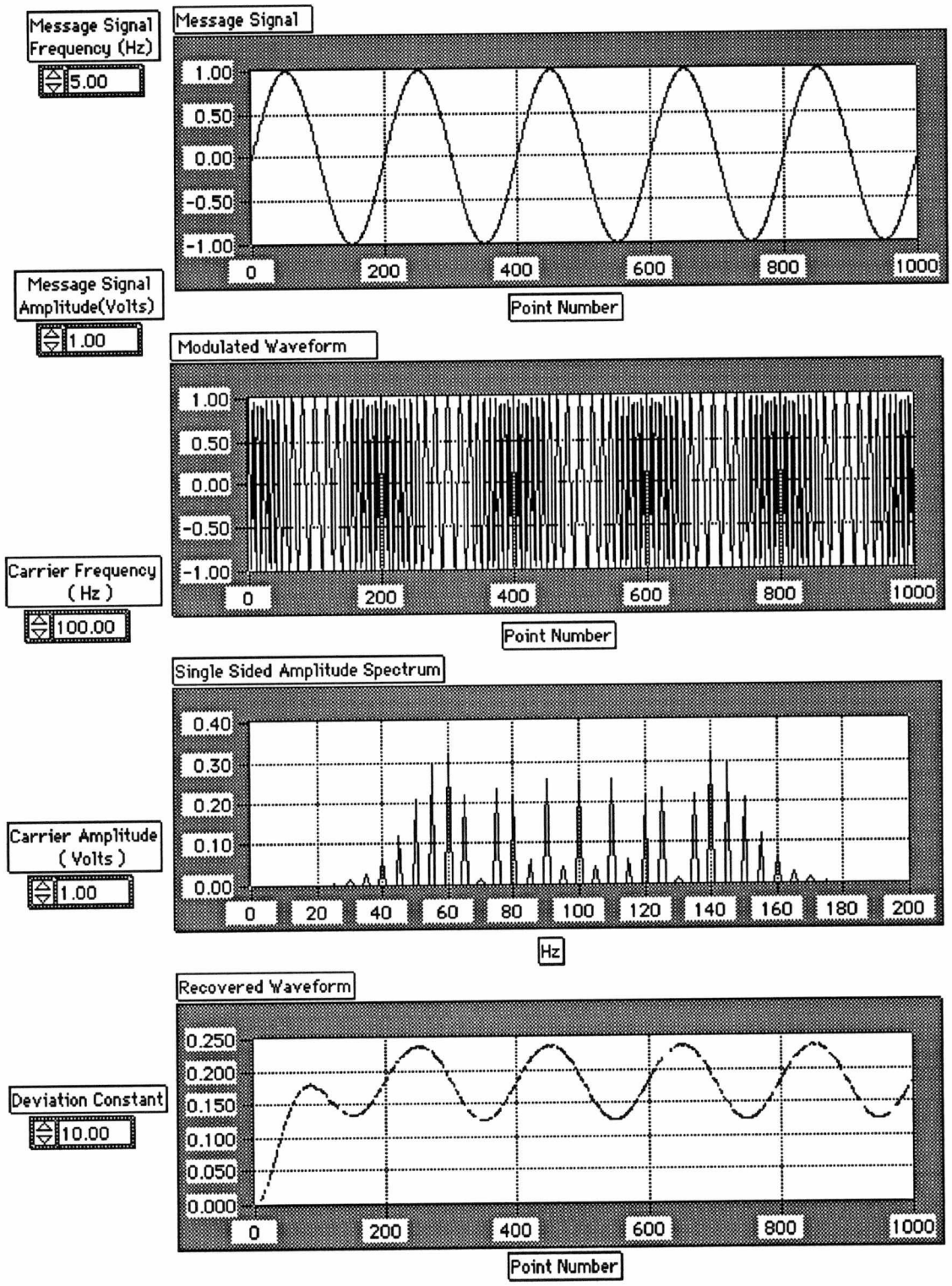


Figure 3.14 Front Panel of PM System

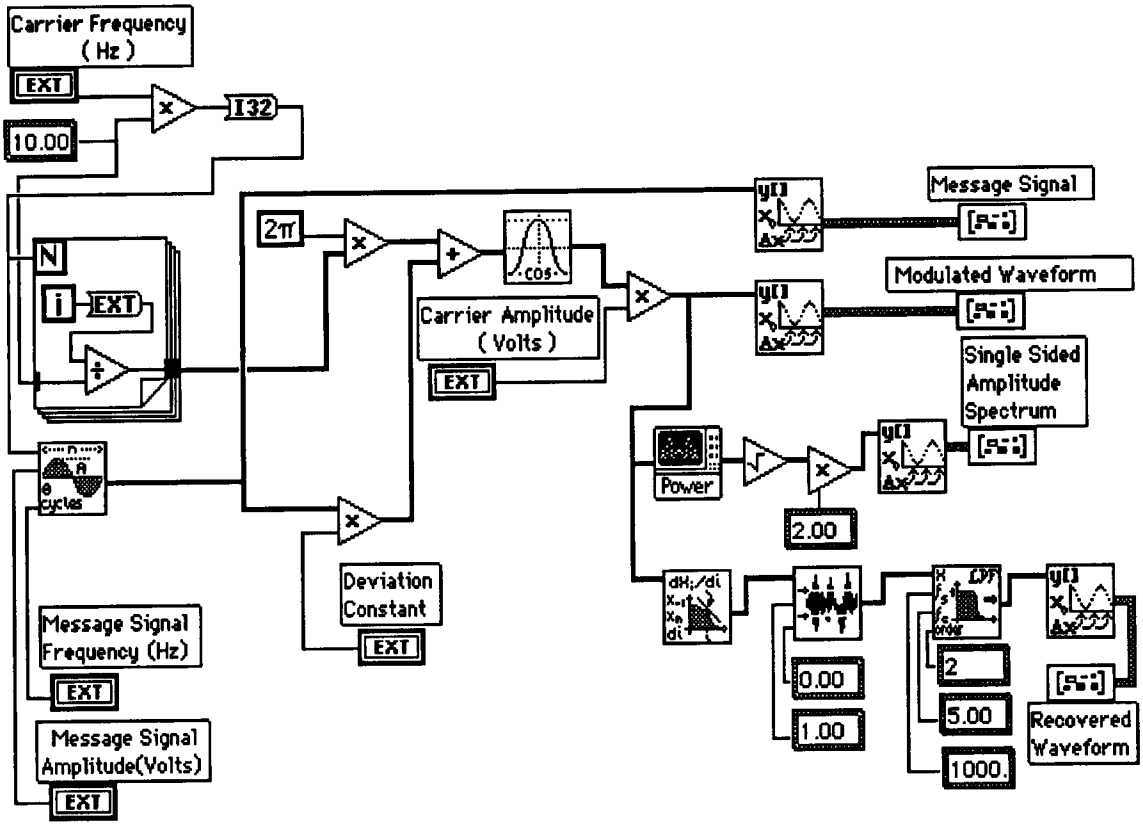


Figure 3.15 Block Diagram of PM System

CHAPTER 4

DIGITAL SYSTEM SIMULATION

A digital communication system may transmit data in baseband or passband mode. For baseband systems digital symbols are transformed into pulse waveforms, but in the case of bandpass systems the digital symbols are allowed to modulate a sinusoidal carrier whereby the amplitude, phase, or frequency of a sinusoidal carrier are varied in a discrete manner. In the following pages baseband as well as bandpass systems with phase and frequency modulation techniques have been discussed along with the LabVIEW programs for their simulation.

4.1 BASEBAND SYSTEM

For the simulation of digital baseband communications, a system consisting of a waveform encoder, transmitter, channel, receiver, and waveform detector is considered (Figure 4.1). The waveform encoder outputs a sequence of pulses (nonreturn-to-zero-

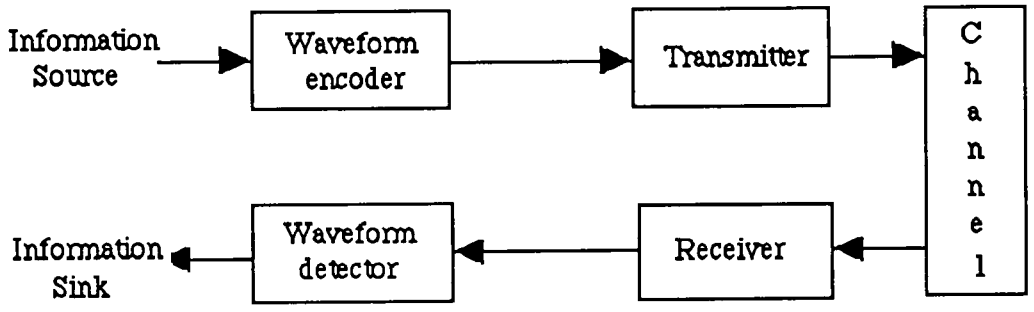


Figure 4.1 Schematic representation of a Baseband System

level) with the characteristics that correspond to the binary digits being sent. After transmission through the channel, the received waveform is detected to produce an estimate of the transmitted digits. The detection process is performed in two steps. Initially the received waveform is reduced to a number and then the the number is compared to a threshold level in order to estimate whether an one or a zero was sent. For the antipodal waveform considered here, the threshold is set to zero. The extent to which noise affects the performance of the system is measured by the probability of bit error. Based on the relative frequency definition of probability the value of bit error probability is obtained by dividing the error count by the number of bits considered. Comparison of the received bits with the transmitted bits gives this error count. The theoretical value of the $P_b(e)$ is given by

$$P_b(e) = Q(\sqrt{\frac{2E_b}{N_o}}) \qquad (4.1)$$

where E_b is the average energy per bit and N_O is the noise power spectral density. A comparison of the theoretical and simulated values of $P_b(e)$ for different values of E_b/N_O is presented later in graphical form.

For the LabVIEW implementation of the above system, two controls and two indicators are selected on the front panel (Figure 4.2). The controls allow the variation of the number of bits to be transmitted and the E_b/N_O values to be considered. One of the indicators in the form of an array presents the values of the bit error probabilities against different values of E_b/N_O selected in the E_b/N_O control. The other indicator in the form of a multiplot graph yields the plots of $P_b(e)$ versus E_b/N_O for theoretical and simulated results.

In the block diagram (Figure 4.3) the arrangement inside the innermost loop yields the error count. The middle loop is used to obtain an average of ten different values of the error count in order to get a better estimation of this random number. Finally, the output of

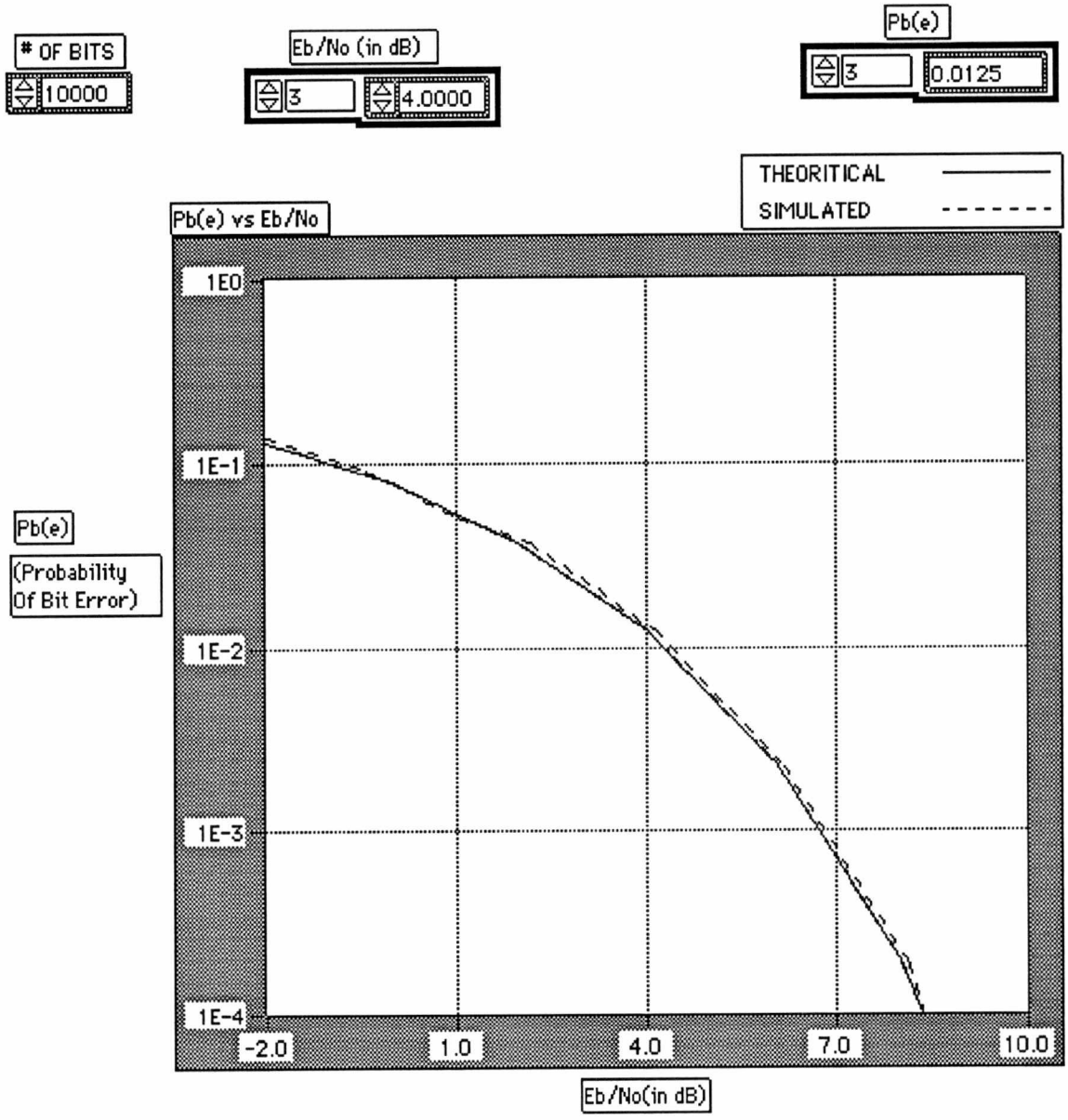


Figure 4.2 Front Panel of Baseband System

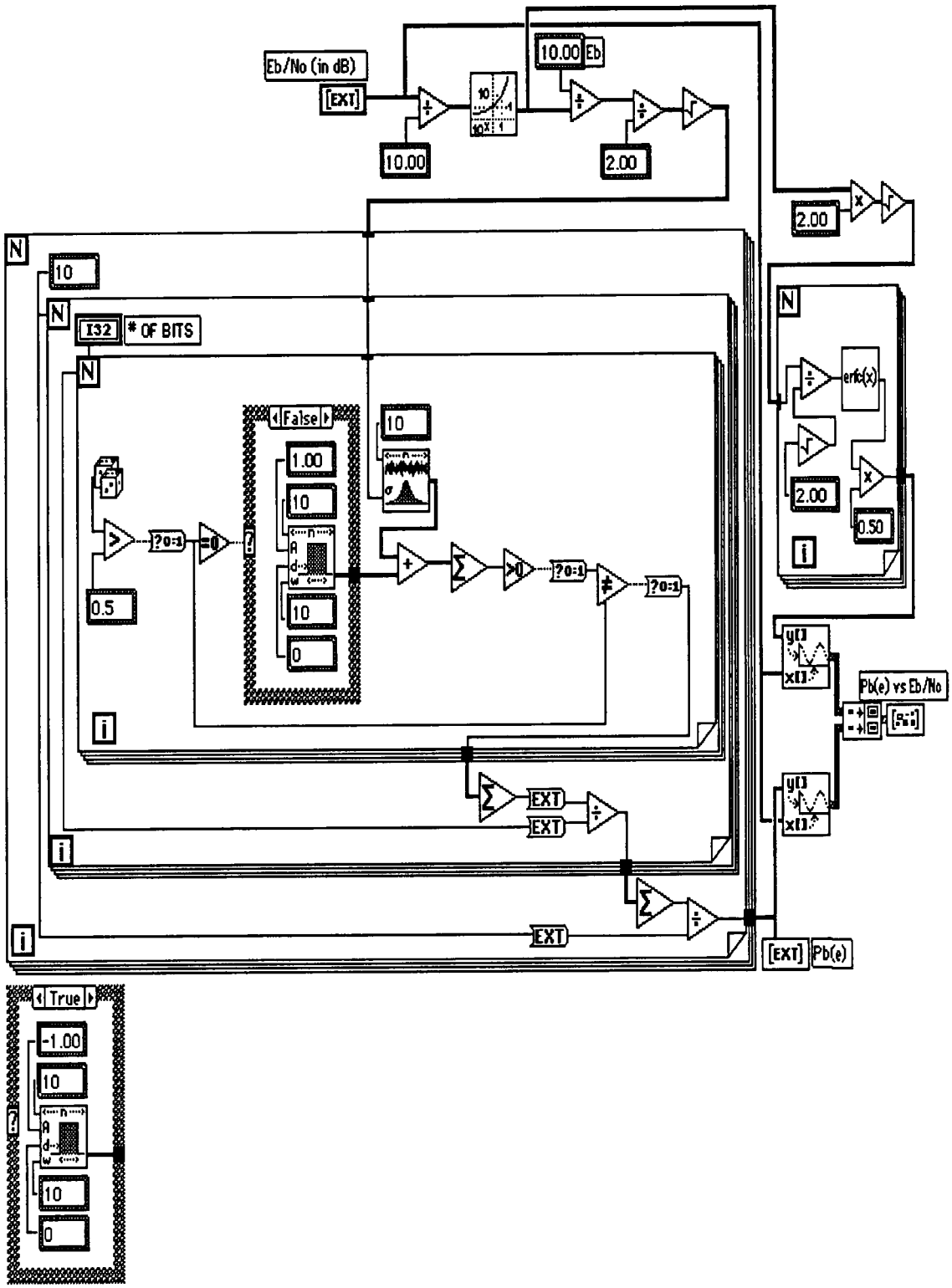


Figure 4.3 Block Diagram of Baseband System

the outermost loop yields the value of $P_b(e)$ for each selected value of E_b/N_o , based on relative frequency definition of probability. By varying the value of E_b/N_o , which in fact varies the standard deviation of the noise pattern, an array of $P_b(e)$ values is obtained against an array of E_b/N_o values. Also an array of theoretical values of $P_b(e)$ is obtained by using the complementary error function VI from the library. The theoretical as well as simulated values of $P_b(e)$ are plotted against the E_b/N_o values by connecting the terminal corresponding to the multiplot graph indicator to the source of these arrays through the array-to-graph conversion function.

The simulation starts by randomly generating a sequence of binary digits. A random number generator in LabVIEW generates a number between zero and one for each run. In the block diagram the random number generator inside the "for" loop is made to iterate N times (equal to the number of bits to be transmitted) and, therefore generates N different numbers between zero and one. Each of these uniformly distributed numbers is compared to a fixed number (0.5). If the random number is greater than the fixed number a one is generated, otherwise a zero using the boolean-to-binary conversion function of LabVIEW. The sequence of bits thus obtained generates a pulse of amplitude +1 if the bit generated is a one. Otherwise (if the bit generated is a zero), a pulse with amplitude -1 is generated. Thus an antipodal waveform is supplied to the transmitter.

The channel is represented by an Additive White Gaussian Noise (AWGN) source. At the receiving end the pulses are processed after being corrupted by noise. In order to reduce the received waveforms to numbers all the elements of the array representing each received pulse are summed up and compared to a threshold (zero in this case). For cases where the summation is greater than zero a one is obtained, otherwise a zero. This received sequence is then compared to the transmitted sequence and a one is obtained for a

mismatch. Summation of these mismatches gives the number of errors. Finally, the error count is divided by the number of bits by using the divide node in order to obtain $P_b(e)$.

4.2 BINARY PHASE SHIFT KEYING (BPSK) SYSTEM

In a BPSK system a pair of signals $s_1(t)$ and $s_2(t)$ are used to represent the binary symbols one and zero, respectively (Figure 4.4); they are defined by

$$s_1(t) = \sqrt{\frac{2E_b}{T}} cos\ (\omega_C t) \tag{4.2}$$

$$s_2(t) = \sqrt{\frac{2E_b}{T}} cos\ (\omega_C t + \pi) \tag{4.3}$$

where $0 \leq t \leq T$ and E_b is transmitted signal energy per bit.

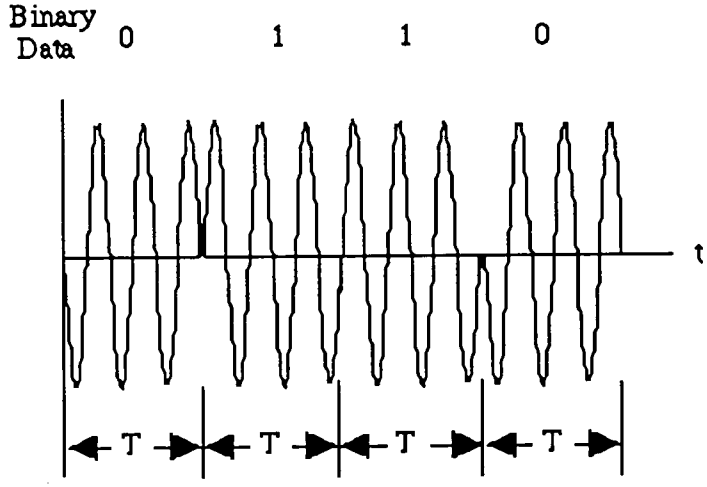


Figure 4.4 Signaling Waveforms for BPSK system

Figure 4.5 shows a schematic for a correlator receiver, while Figures 4.6 and 4.7 illustrates the front panel and the block diagram for the BPSK system. The modulating data signal shifts the phase of the carrier waveform to one of two phase states, either zero or 180 degrees (Figure 4.7). The carrier is a sinusoidal waveform generated by a sine wave generator whose frequency is controlled by the carrier frequency control in the front panel (Figure 4.6). Here the phase for both cases has been kept at zero and the amplitude has been taken as +1 or -1. Depending upon the bit to be transmitted the change in amplitude gives an effect equivalent to a change in phase. The channel is represented by an AWGN source.

The detection process is performed using a correlation detector (Figure 4.5). The reference signal is taken to be the difference of the two transmitted signals. The received signal, $r(t)$ is multiplied by the reference signal and all the elements of the resultant array are summed up in order to get a single number, $z(t)$. The number thus obtained is compared to a threshold value γ (zero for the antipodal case). If the number is found to be greater than zero the boolean to binary converter gives a one; otherwise a zero. This output in fact represents the detected symbols, $\hat{s}_i(t)$. Comparison of the transmitted and received symbols gives the number of errors and then $P_b(e)$ is found by dividing the number of

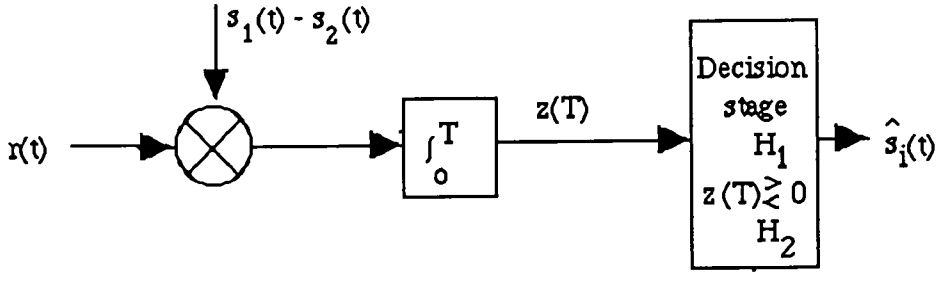


Figure 4.5 Schematic representation of Binary Correlator Receiver

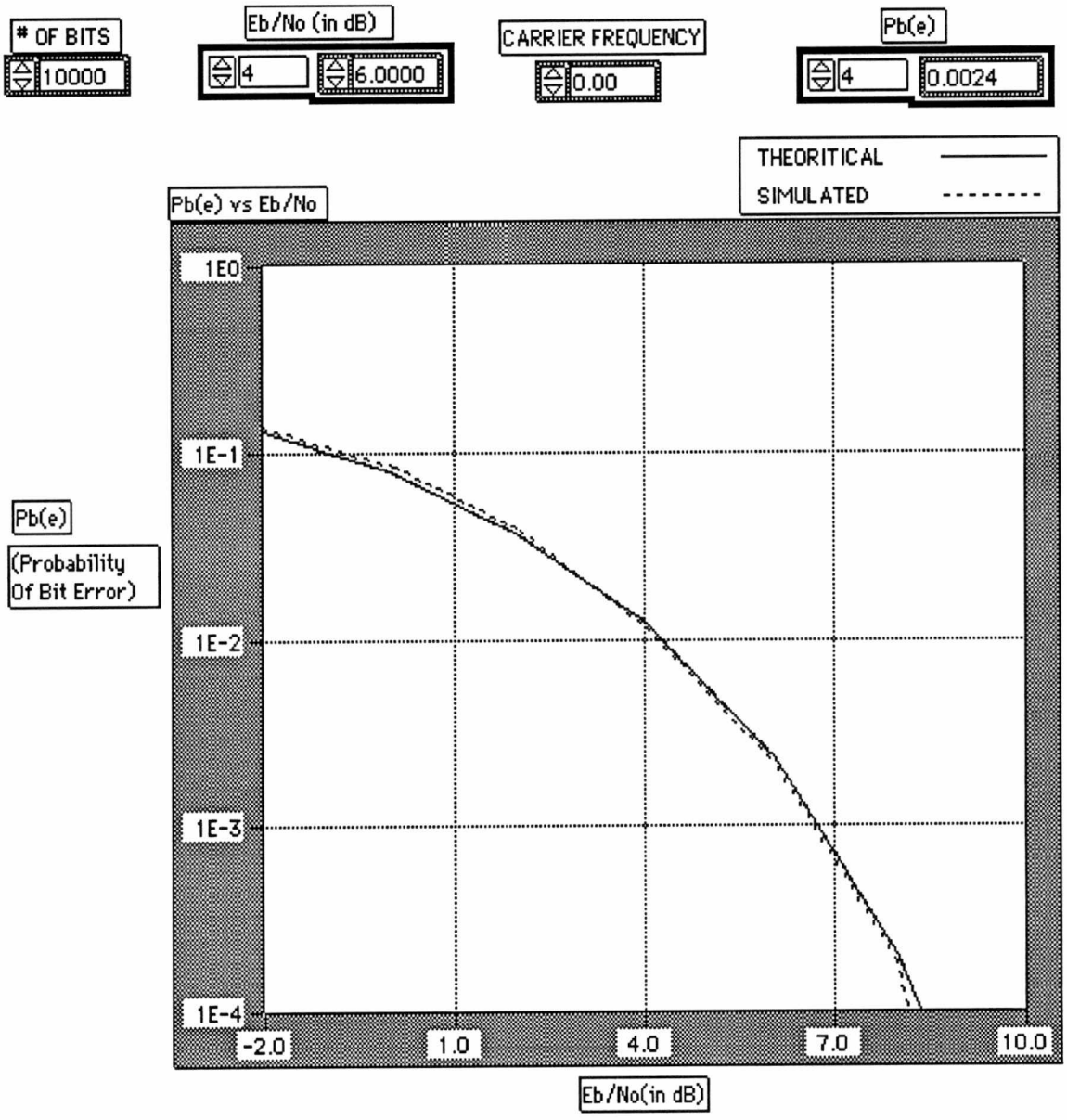


Figure 4.6 Front Panel of BPSK System

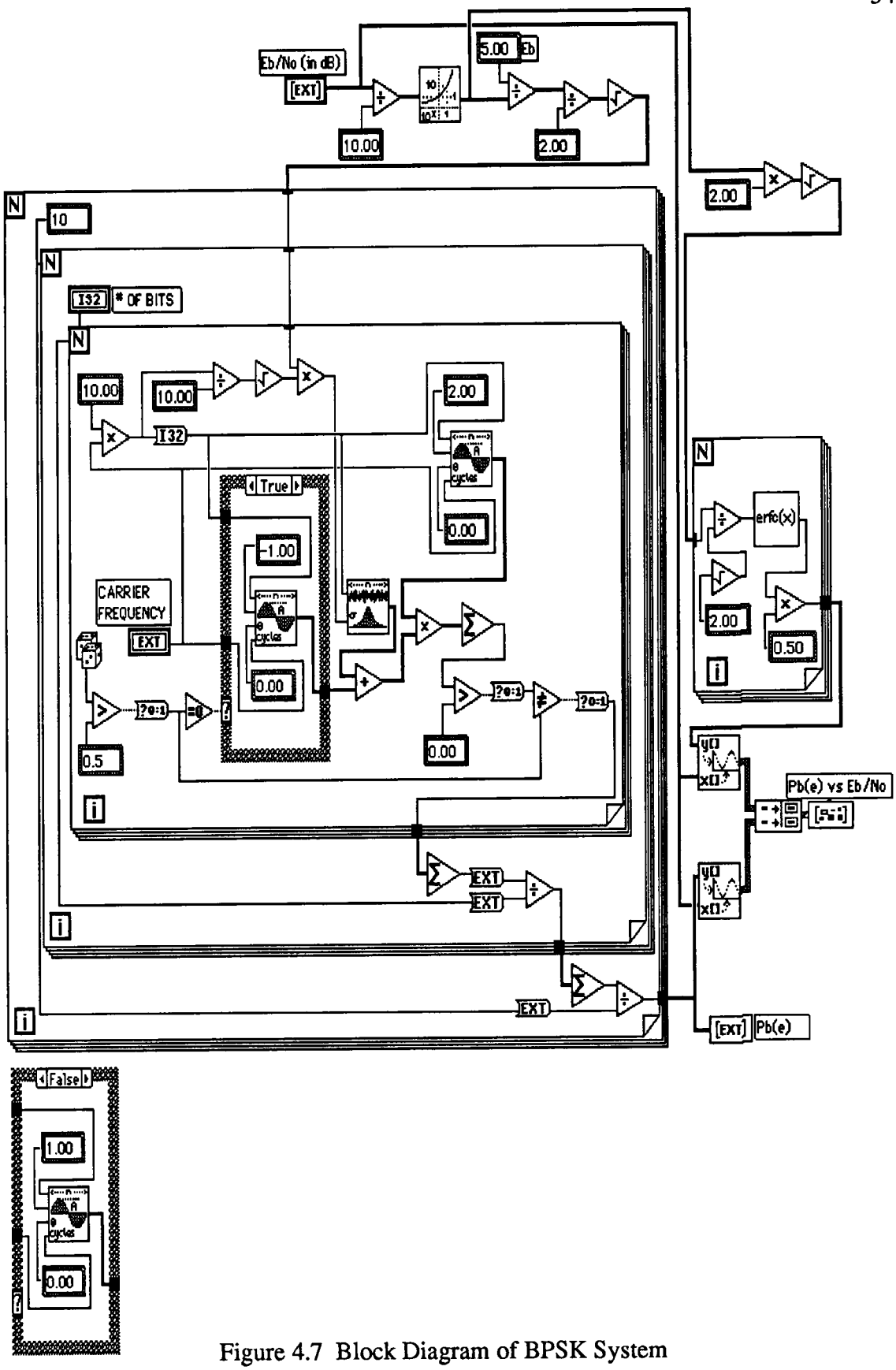


Figure 4.7 Block Diagram of BPSK System

errors by the number of bits considered. The arrangement outside the "for" loop is for obtaining the theoretical values of $P_b(e)$ by realizing the expression

$$P_b(e) = Q(\sqrt{\frac{2E_b}{N_o}}) \qquad (4.4)$$

The number of samples for the carrier, noise, and reference signals is made ten times the value of the carrier frequency so that each cycle is represented by at least ten samples and as such a better approximation of the signal or noise is obtained. An arrangement has been made to set all these values by setting a value for the carrier frequency control in the panel. Increasing the number of samples increases the energy per bit. Therefore, an arrangement has been made to set the corresponding value of the standard deviation in the AWGN generator VI to correspond to the change in bit energy.

4.3 BINARY FREQUENCY SHIFT KEYING (BFSK) SYSTEM

In BFSK system, the symbols one and zero are distinguished from one another by the transmission of one of two sinusoidal waves that differ in frequency by a fixed amount. A typical pair of sinusoidal waves is described by

$$s_i(t) = \sqrt{\frac{2E_b}{T}} cos\,(\omega_i t) \qquad (4.5)$$

where ω_i equals one of two possible values: ω_1 and ω_2. Transmission of frequency ω_1 represents symbol one, and transmission of frequency ω_2 represents symbol zero as shown in Figure 4.8.

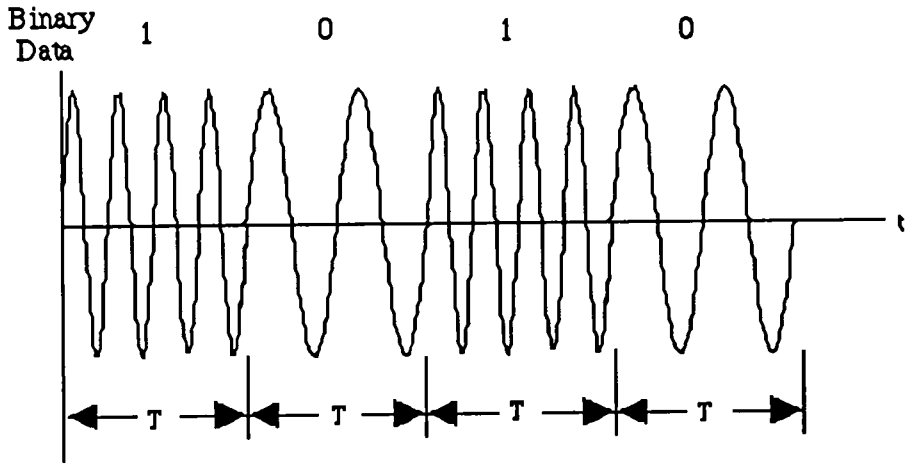


Figure 4.8 Signaling Waveform for BFSK system

FSK signaling may be coherent or non-coherent i.e., the system may or may not use the phase of the received signal for detection purposes. Correspondingly the hardware configuration changes for the receiver. Figure 4.9 shows the schematic diagram of a non-coherent receiver; a receiver for coherent detection is the same as one used for the BPSK system. For the transmitter there is not much difference. However in order for the signal set to be orthogonal the minimum required tone spacing (multiple of 1/T hertz) should be maintained. Both schemes have been simulated as shown by their front panels (Figures 4.10 and 4.12) and block diagrams (Figure 4.11 and 4.13).

For the coherent case the block diagram does not differ much from the BPSK case. The two carrier waveforms differ in frequency, whereas in the case of BPSK system they differed in phase. The reference signal in the correlator receiver is again the difference of

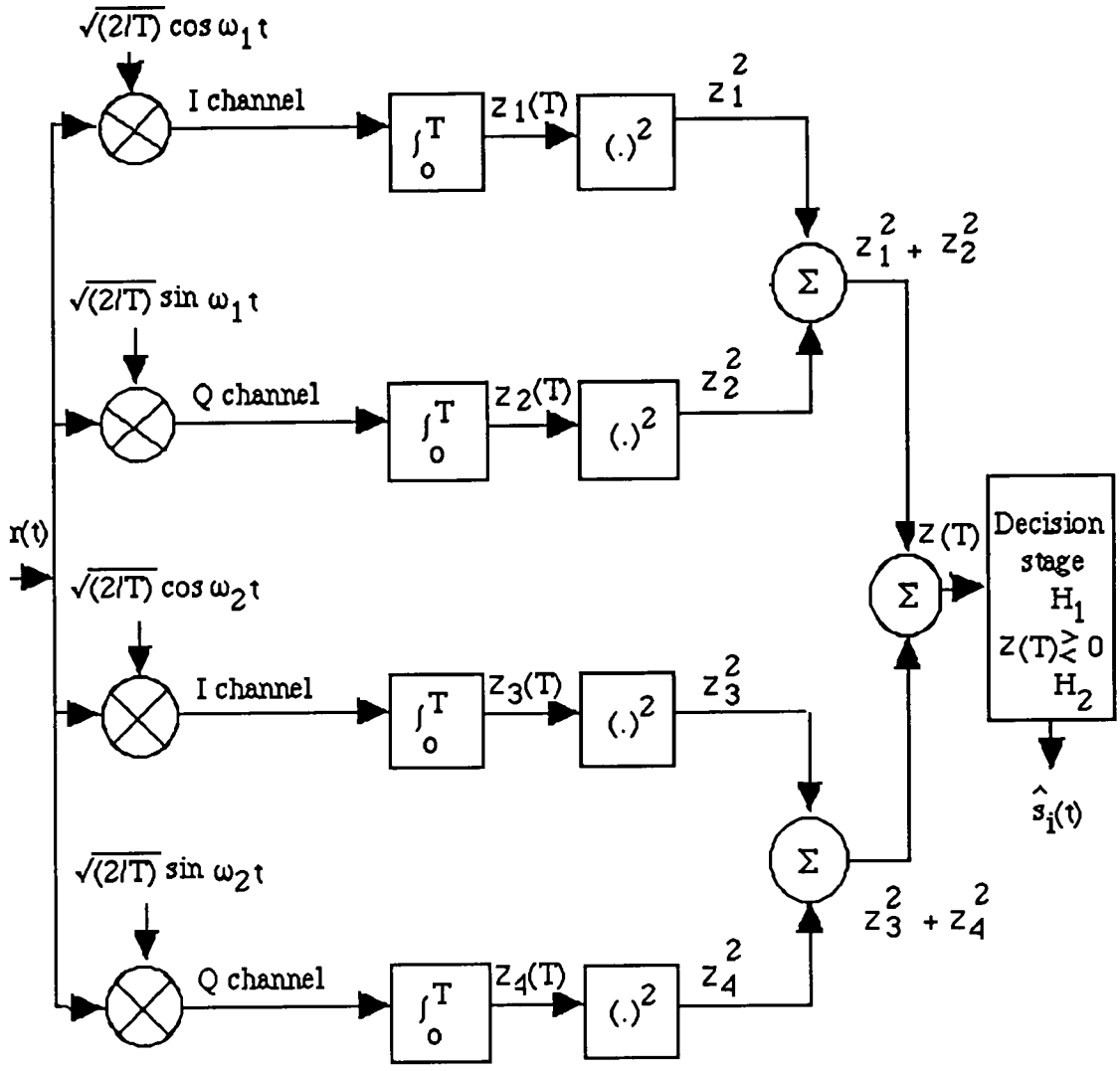


Figure 4.9 Schematic representation of Quadrature Receiver

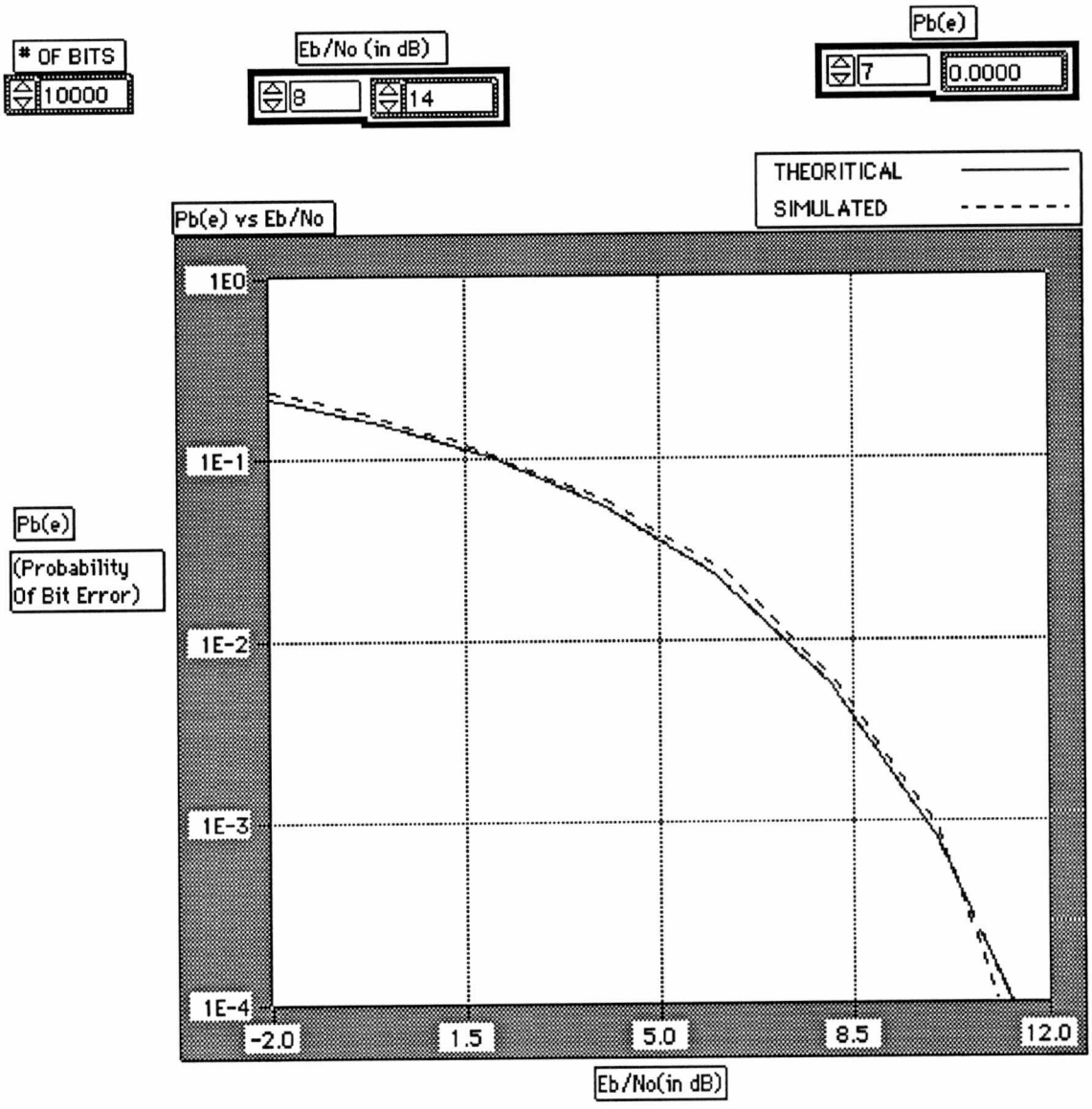


Figure 4.10 Front Panel of BFSK (Coherent) System

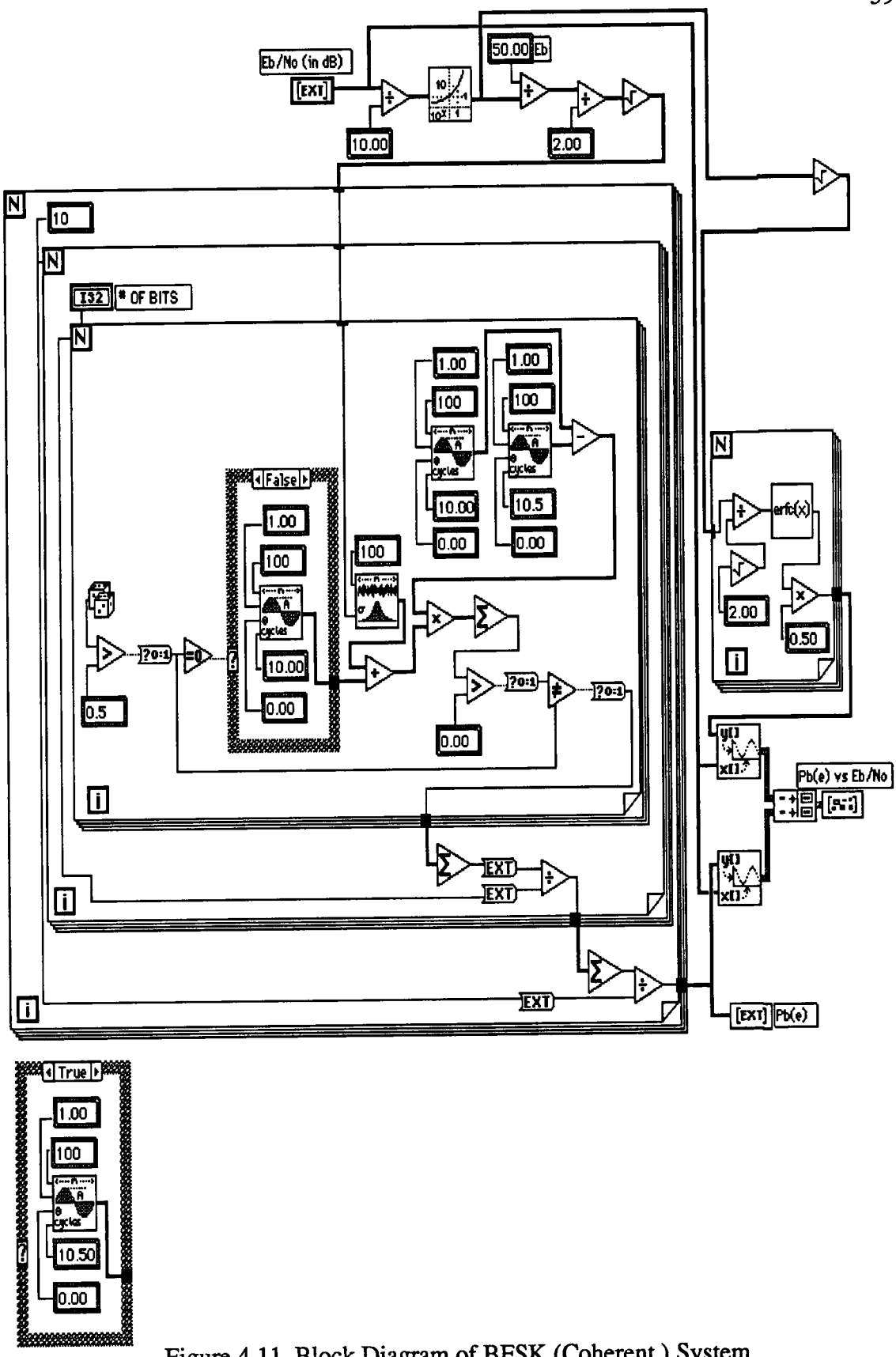


Figure 4.11 Block Diagram of BFSK (Coherent) System

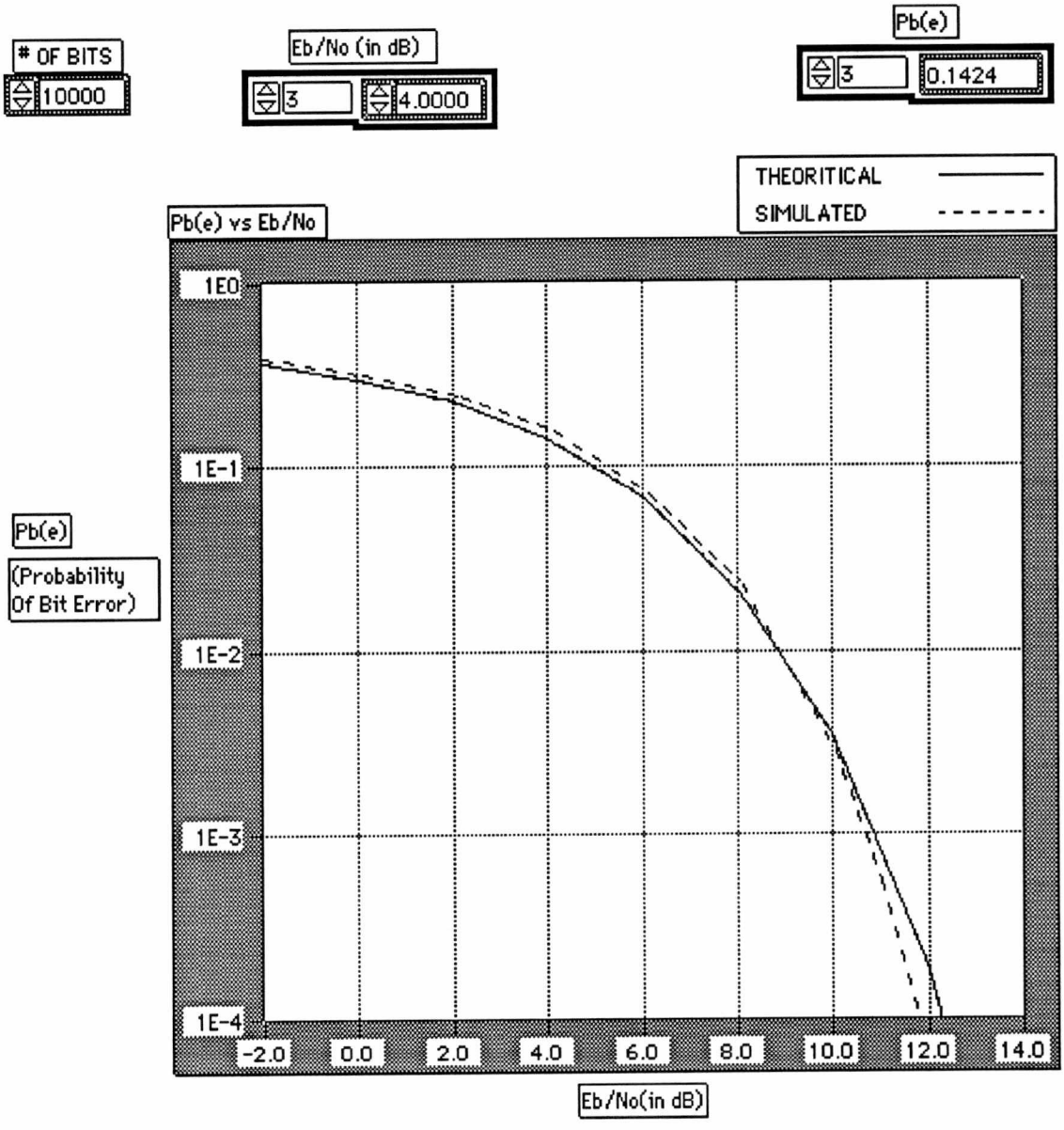


Figure 4.12 Front Panel of BFSK (Non-Coherent) System

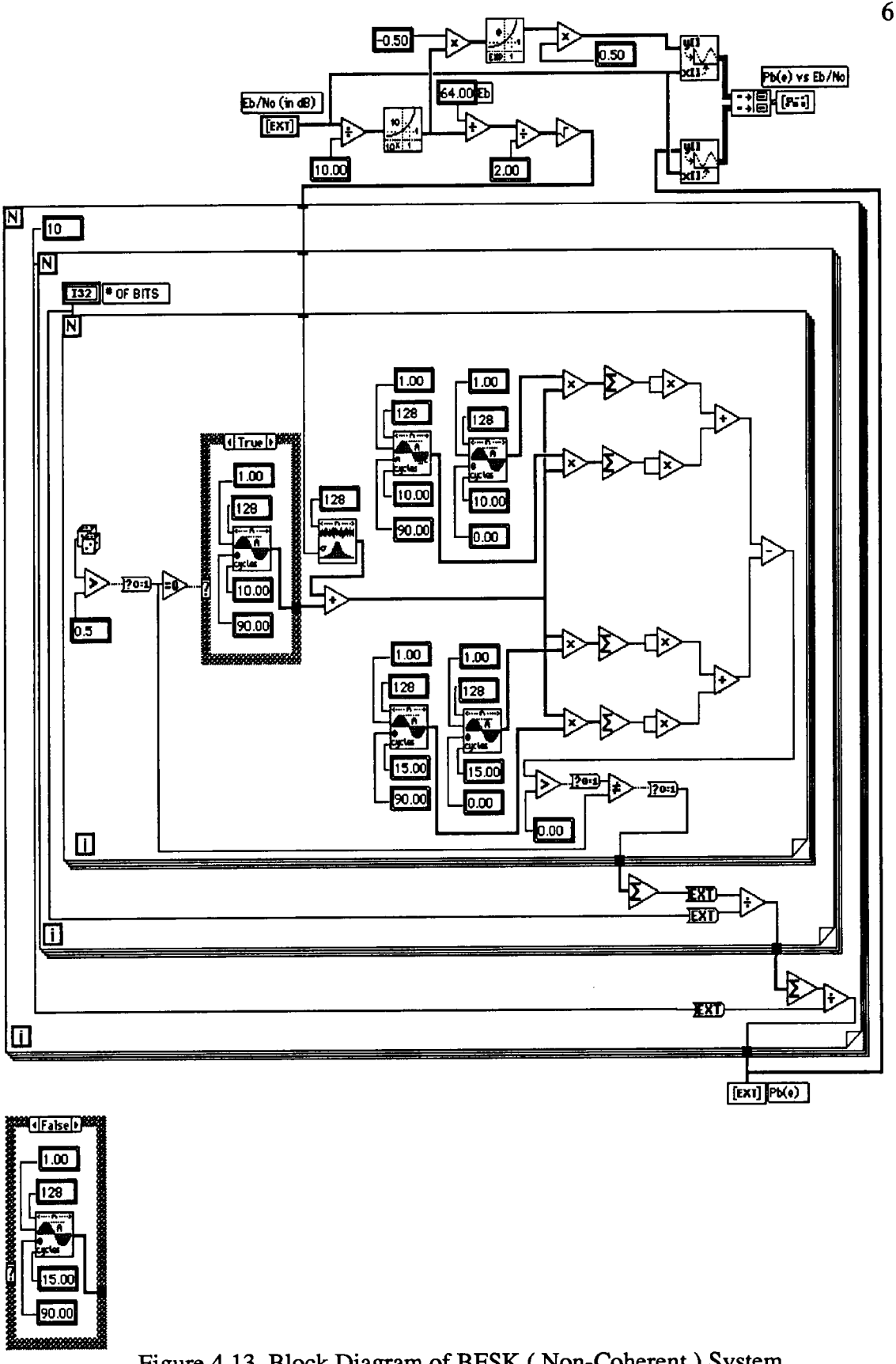


Figure 4.13 Block Diagram of BFSK (Non-Coherent) System

the two signals. The arrangement for obtaining the theoretical values has been kept the same except for thr elimination of a multiplication node in order to realize the expression

$$P_b(e) = Q(\sqrt{\frac{E_b}{N_o}}) \qquad (4.6)$$

For non-coherent detection the configuration implements a quadrature receiver. The in-phase and quadrature reference signals for both frequencies are multiplied by the received signal. All the elements from the product operation are summed up in order to perform the integration operation. Then squaring is performed by a product node. The results of the four branches are summed up and compared to the threshold value. Depending upon the number obtained from the threshold test, ones or zeros are obtained from the boolean-to-binary converter. Comparison of these received bits with the transmitted bits gives the number of errors and hence the $P_b(e)$. The arrangement outside the "for" loops is used to obtain the theoretical values of $P_b(e)$ given by

$$P_b(e) = \frac{1}{2} exp\left(-\frac{1}{2}\frac{E_b}{N_0}\right) \qquad (4.7)$$

4.4 CODED SYSTEM

Channel coding improves communication performance by enabling the transmitted signal to better withstand the effects of various channel impairments, such as noise, fading and jamming. Channel coding is partitioned into two study areas: waveform coding and structured sequences (or structured redundancy). Waveform coding deals with transforming waveforms into better waveforms to make the detection process less subject to errors. Structured sequence deals with transforming data sequences into "better sequences", having structured redundancy (redundancy bits). The redundant bits can then be used for the detection and correction of errors [12]. A baseband system coded with

structured sequence has been considered. Channel impairment in this case is due to AWGN. Single error correction is performed by the decoder.

The encoder uses a shift register (Figure 4.14) to transforms a block of four message digits (a message vector) into a larger block of seven codeword digits (a code vector) constructed from a given alphabet of elements (zero and one in this case). In the LabVIEW implementation the front panel (Figure 4.15) for the encoder is provided with a message vector control and a code vector indicator. In the block diagram (Figure 4.16) a "while" loop with three shift registers is chosen. A shift register has a pair of terminals opposite each other on the vertical sides of the loop border: the right terminal stores the data at the completion of an iteration; the data is shifted at the end of the iteration to the left terminal in time for the next iteration. Each bit of the message vector is modulo-2 added

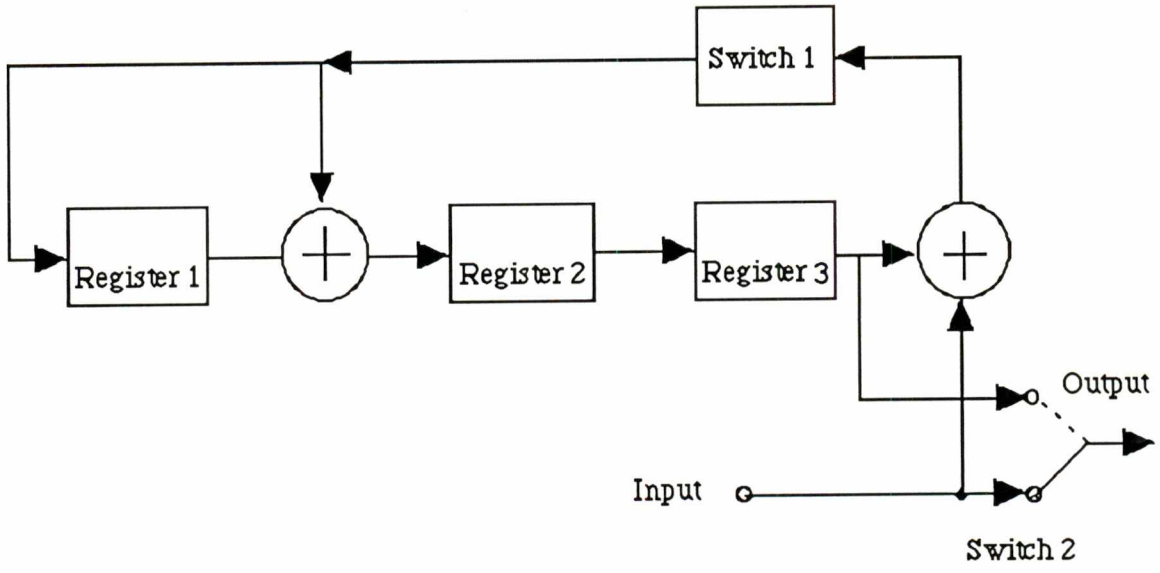


Figure 4.14 Schematic representation of Shift Register for Encoding

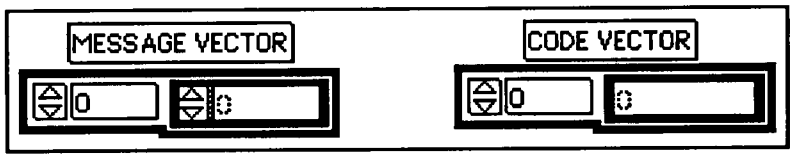


Figure 4.15 Front Panel of Encoder

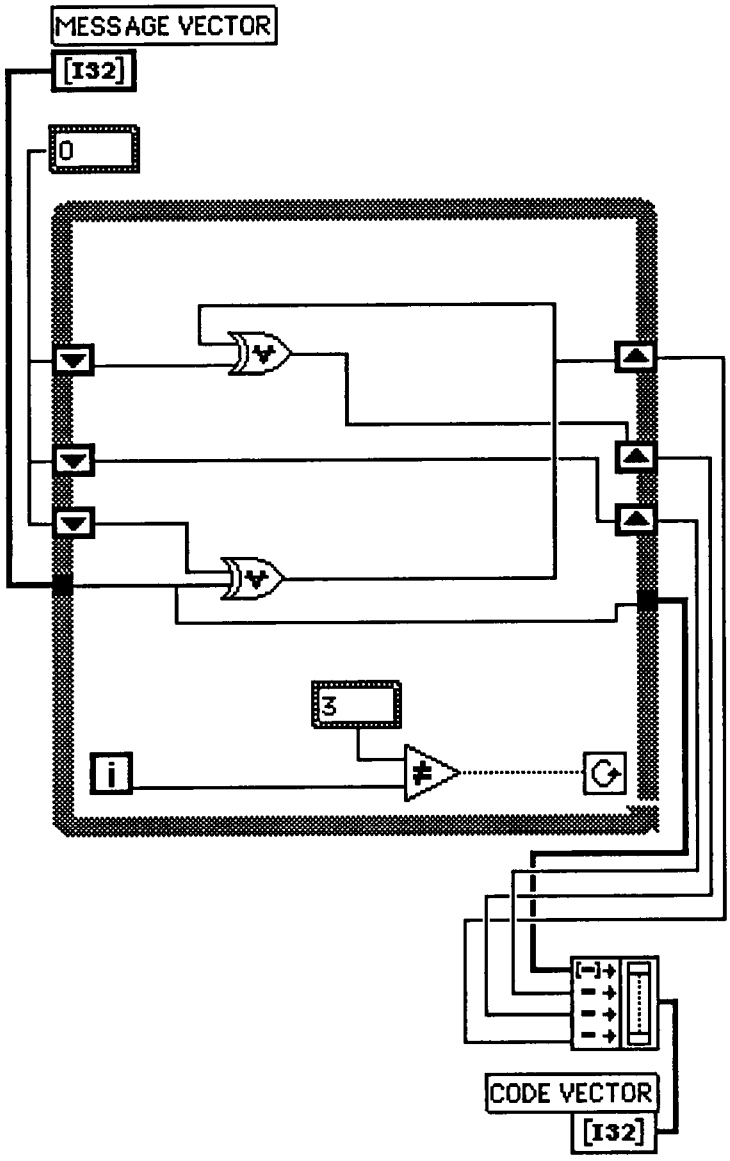


Figure 4.16 Block Diagram of Encoder

with the contents of the third shift register. The outcome is passed to the first register as well as modulo-2 added with the previous contents of the first register. The result of the second modulo-2 addition is passed to the second register which automatically passes it to the third register in each shift. Entry of a single bit from the message vector into the loop causes one shift in the registers. After four shifts the content of the registers represents the parity bits. When these parity bits are appended to the message vector the code vector is obtained.

For decoding, initially the syndrome is calculated using a shift register (Figure 4.17); then a "look up table method"is used for error detection and correction. The "look up table" method matches each value of the syndrome with a particular error pattern.LabVIEW

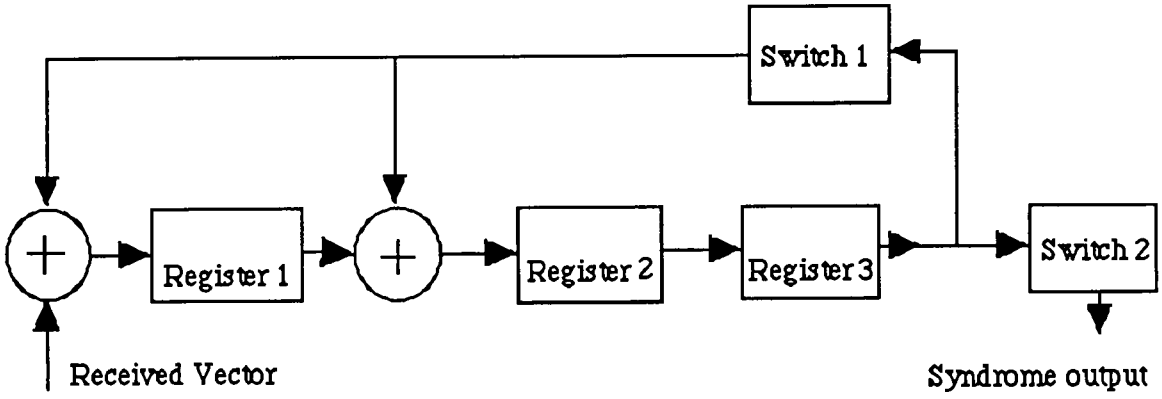


Figure 4.17 Schematic Representation of Shift Register for Syndrome Calculation

implementation requires a received vector control and corrected vector indicator in the front panel (Fig. 4.18). In the block diagram (Figure 4.19) a while loop with three shift registers is chosen as in case of the encoder. The received vector is modulo 2 added with the content

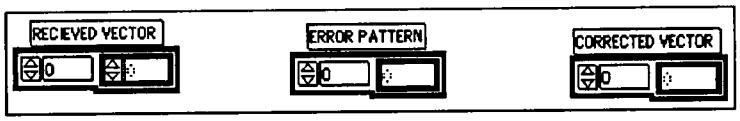


Figure 4.18 Front Panel of decoder

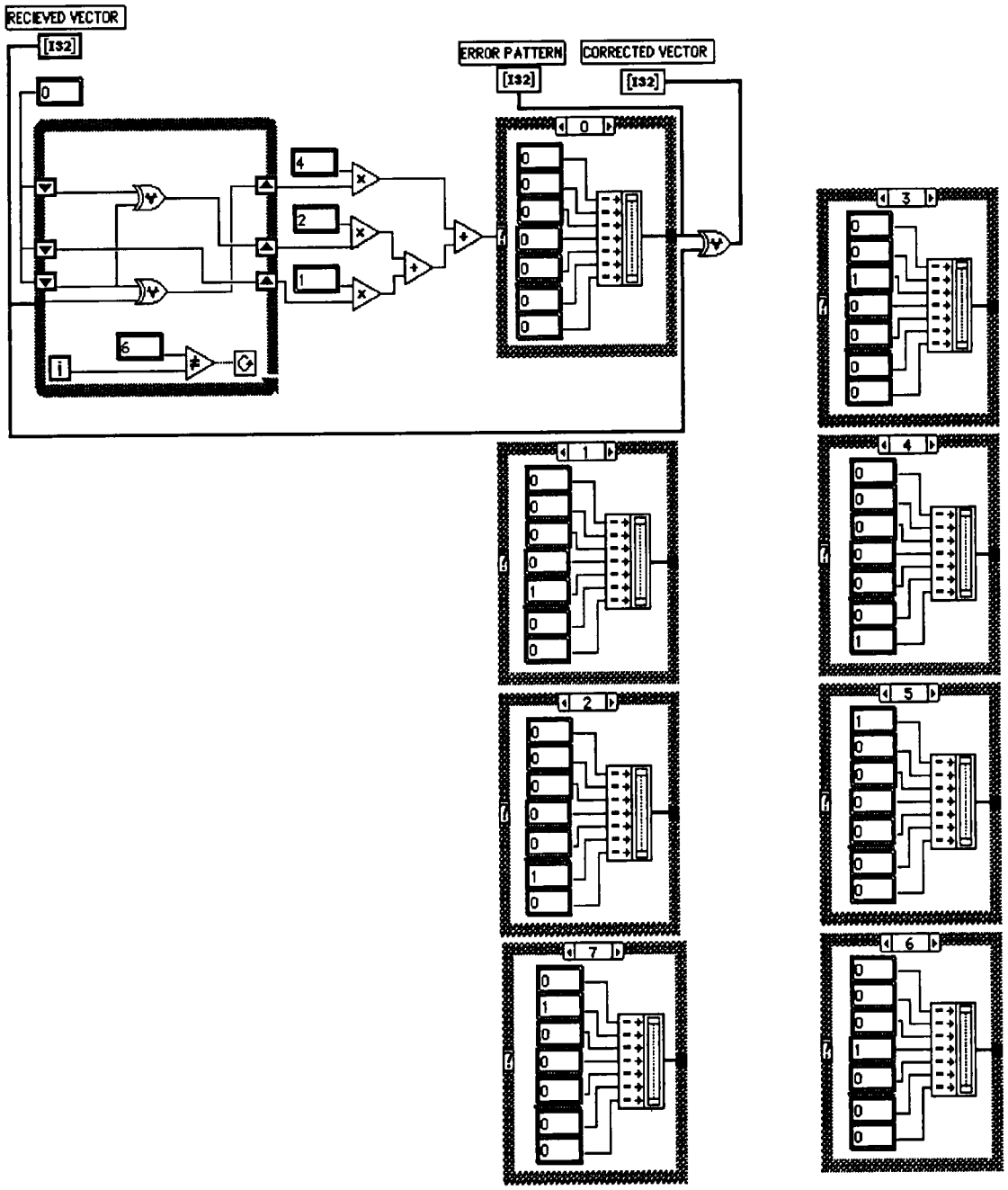


Figure 4.19 Block Diagram of Decoder

of the third shift register. Output of this modulo-2 operation is feed into the first shift register. Another modulo-2 addition is performed between the contents of the first and the third shift registers. The result of the second modulo-2 operation is passed into the second shift register which automatically is passed into the next register in the subsequent shift. The contents of the three registers after seven shifts gives the syndrome of the received vector. The syndrome is the result of a parity check performed on the received code vector to determine whether the code vector is a valid member of the codeword set. Binary representation of the syndrome value is converted into its octal equivalent. Thus for each received vector an octal number is received which corresponds to a particular case in the case structure with eight different cases representing different error patterns. Once the error pattern for a particular received vector is found it is added to the received vector and thus a codeword with single error correction is obtained.

The coded system shown in Figure 4.20 and 4.21 is basically the baseband system with coding and decoding sub-VIs incorporated. However, instead of generating bits to be transmitted in the stream, streams of words with four bits are generated in this case. After generating a message word of four bits it is passed through the encoder which converts it to codeword of seven bits. Depending upon the bit the waveform generator transmits a pulse of amplitude +1 or -1. Implementation of the channel and the receiver is similar to the original baseband case. The received codeword is passed through the (7,4) decoder and error corrected codewords are obtained. As the decoder is capable of correcting only single bit errors there may be word errors after decoding. Comparing the transmitted codeword with the decoded codeword, the number of errors is obtained which finally enables one to

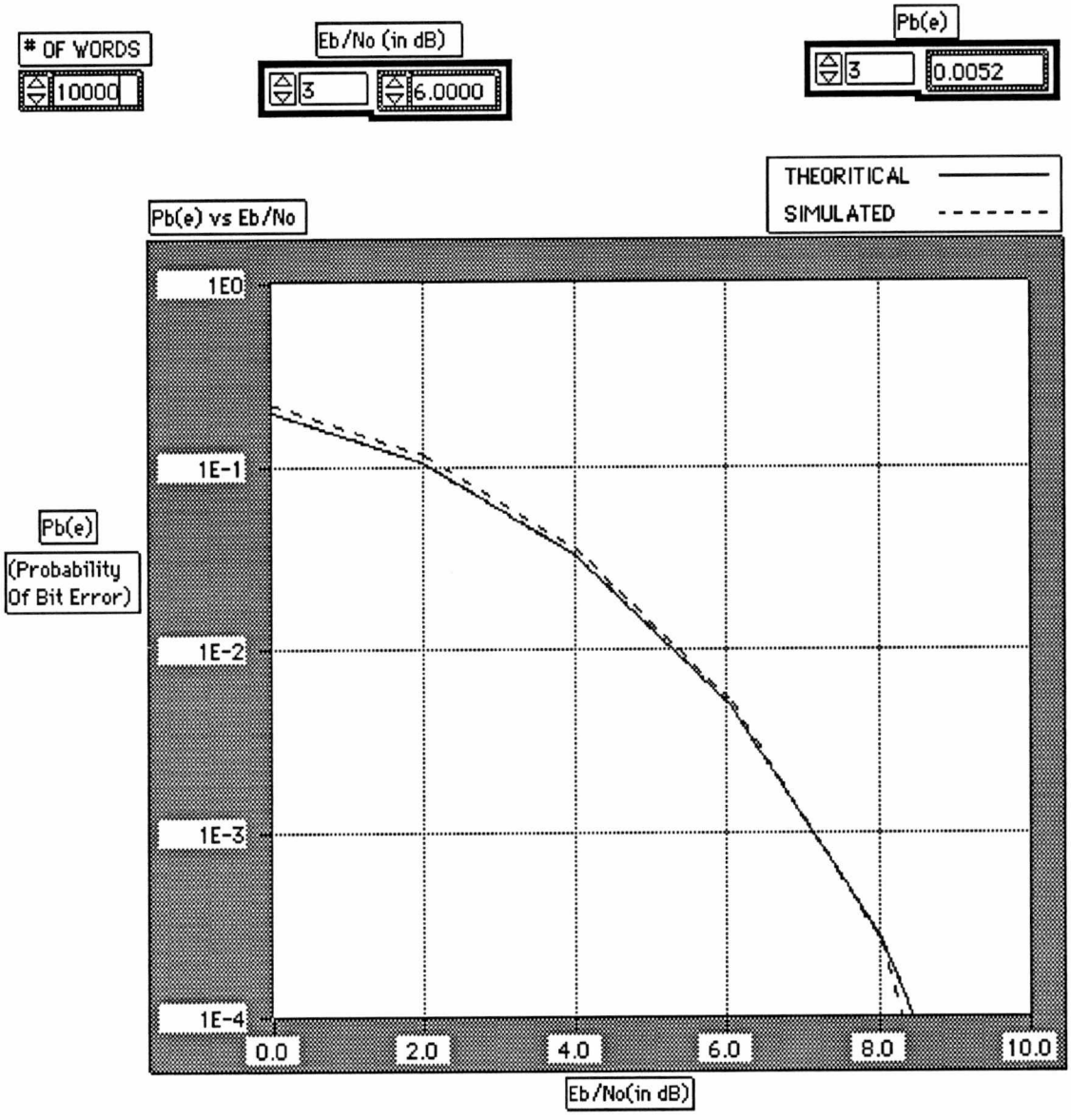


Figure 4.20 Front Panel of Baseband (Coded) System

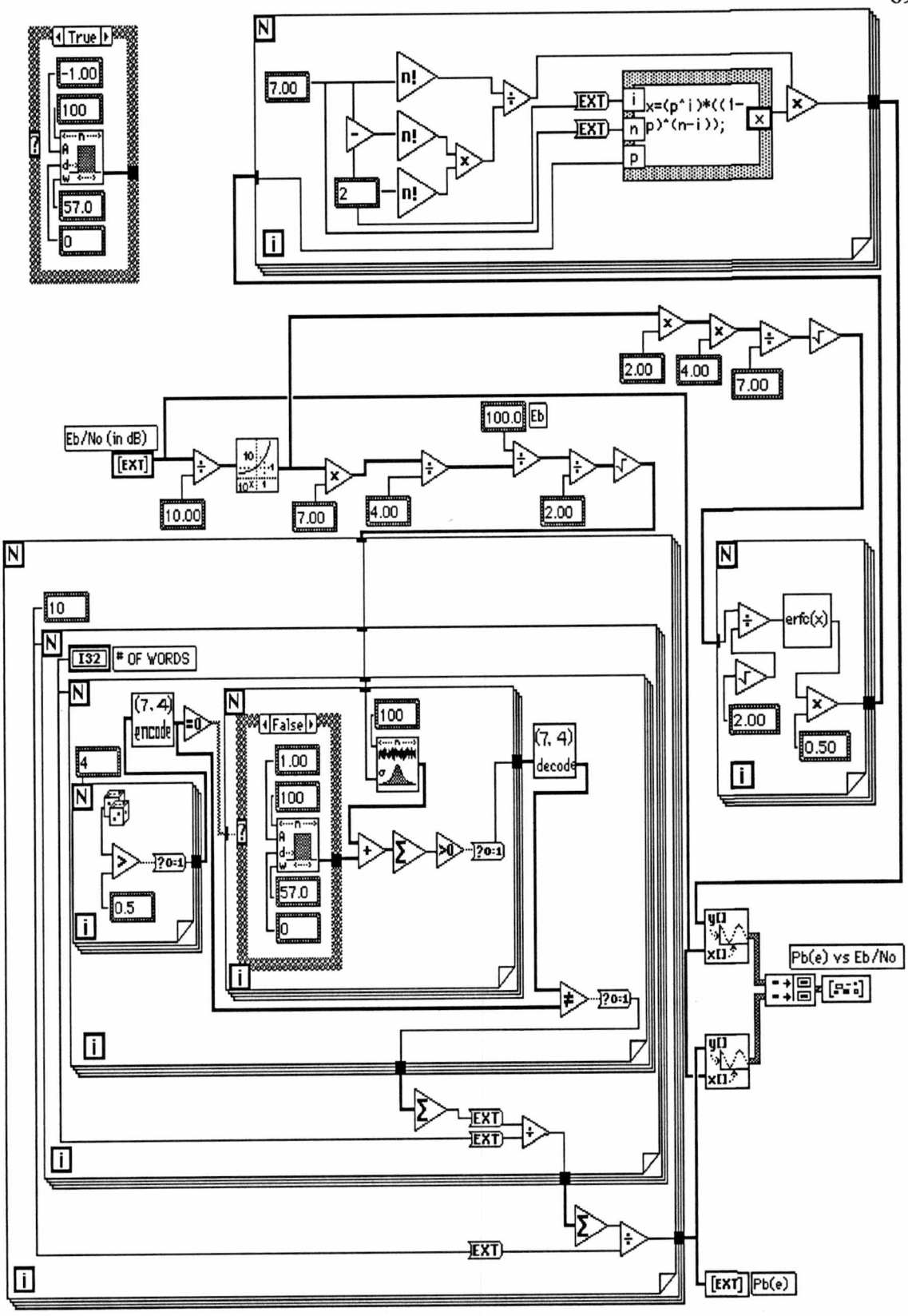


Figure 4.21 Block Diagram of Baseband (Coded) System

find the probability of word error. Theoretical values of probability of word error, P_M are obtained by direct implementation of the Equation 4.8 [12], where p is the probability that a

$$P_M = \sum_{j=t+1}^{n} \binom{n}{j} p^j (1-p)^{n-j} \qquad (4.8)$$

symbol is received in error,with the help of formula node as shown in the block diagram. Formula nodes give the value of the output defined depending upon the values of the inputs defined. The output and the inputs are related by the expression written inside the sizeable box using syntax similar to conventional programming languages.

4.5 SIMULATION TIME STUDY

Figures 4.22 and 4.23 shows the time taken by the digital baseband system of Figure 4.3 to complete a simulation run. An arrangement as shown in Figures 4.24 and 4.25 has been made to record the time taken for simulation of the communication systems. The VI representing the communication system is placed in the middle structure of a sequence of three structures. The "Tick Count" sub VI obtained from the LabVIEW VI library is placed in the first and third structures. This sub VI outputs the current value of the internal clock that ticks sixty times per second from the time the system is turned on. Since the execution is sequential the difference of the values from the first and third structures gives the time taken by the VI under consideration for a complete run.

Number of Bits considered	10	100	1000	10,000
Number of $P_b(e)$ values				
1	1.52	13.40	132.25	1,335.15
10	13.75	133.40	1,335.12	13,345.80

Figure 4.22 Time Taken (seconds) by the Baseband System

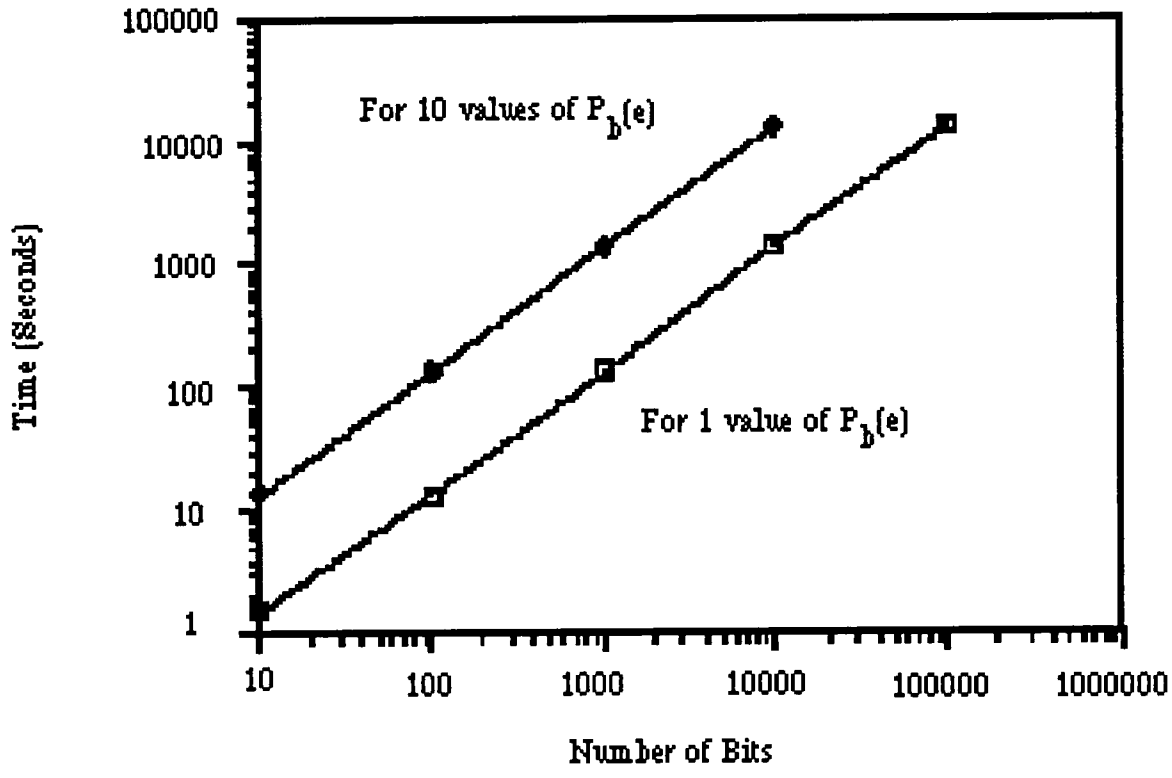


Figure 4.23 Plot of the Time Taken by Baseband System

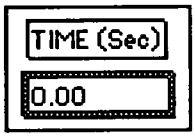


Figure 4.24 Front Panel of Timer

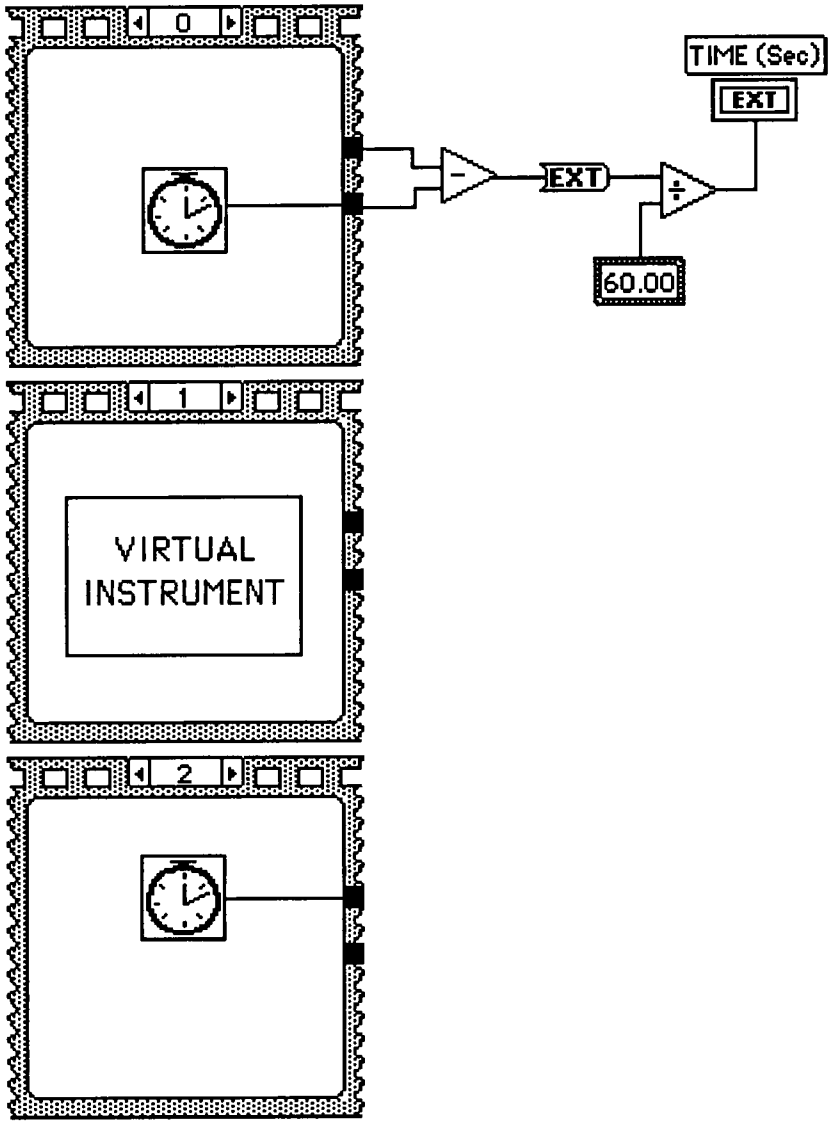


Figure 4.25 Block Diagram of Timer

CHAPTER 5

CONCLUSION

In the present work a general purpose software system, namely the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) has been used to simulate various communication systems: analog and digital. The computer simulation of communication systems has been found to be very useful and effective by designers and analysts. Various software packages providing a high level language for description of the system operations have been developed. A recent trend towards developing user friendly and intelligent software packages has resulted in some visual programming tools. A relatively new addition to the visual programming tools collection is LabVIEW developed by National Instruments. This general purpose tool has been found to be very attractive from the viewpoint of flexibility and versatility as it allows the simulation of many different systems and permits easy modification of the system parameters and structures. In LabVIEW system models are required to be defined by using graphical primitives. Use of multiple windows, pop-up menus, and mouse operation provides a very friendly environment. The hardware requirement for LabVIEW operation is easily accessible and not very costly. With respect to speed, the performance has been quite satisfactory as a simulation run of one hundred thousand samples could be completed in less than twenty three minutes. The results obtained from the simulation of communication systems have been found to be very much consistent with the theoretical results (within 90-100 % depending on the system and number of samples considered). Hence, it may be concluded that the LabVIEW Software System can be a very effective and interesting tool for simulation of communication systems.

LIST OF REFERENCES

[1] K. Sam Shanmugan, "An Update on Software Packages for Simulation of Communication System (Links)",IEEE Journal on Selected Areas of Communication, (JSAC) Vol. 6, No. 1, Jan 1988, pp 5-12.

[2] Michael Fashano and Andrew L. Strodtbeck, "Communication System Simulation and Analysis with SYSTID", IEEE JSAC, Vol.SAC-2, No.1, January 1984., pp.8-29

[3] Marco Ajmone Marsan, Sergio Benedetto, Ezio Biglieri, Valentino Castsllani, Michele Elia, Letizia LO Presti, and Mario Pent, "Digital Simulation Of Communication System with TOPSIM III", IEEE JSAC, Vol.SAC-2, No 1, Jan 1984, pp. 29-41

[4] James W. Modestino and Kurt R. Matis, "Interactive Simulation of Digital Communication System", IEEE JSAC, Vol. SAC-2, No.1 Jan.1984, pp.51-76

[5] Kurt R. Matis and James W. Modestino, "Interactive Communication System Simulation on a High Speed PC Workstation", IEEE JSAC, Vol.6, No. 1, Jan 1988, pp.24-33

[6] Carl Ryan and Don Jordan, " A Communication System Analysis Program for the IBM-PC", IEEE Conf. Rec. Phoenix Conf. Comput. and Commun., Mar. 1986.

[7] William H. Tranter and Carl R. Ryan,"Simulation of Communication System Using Personal Computers", IEEE JSAC, Vol.6,No. 1, Jan 1988, pp.13-23

[8] David G. Messerschmitt, "A Tool for Structured Functional Simulation", IEEE JSAC, Vol.SAC-2, No.1, Jan 1984,pp 137-147.

[9] Walter R. Braun and Teresa M. McKenzie, "CLASS:A Comprehensive Satellite Link Simulation Package", IEEE JSAC, Vol. SAC-2, No. 1, Jan 1984, pp 129-137.

[10] William D. Wade, Mary E. Mortara, Peter K. Leong, and Victor S. Frost, "Interactive Communication System Simulation Model-ICSSM", IEEE JSAC, Vol. SAC-2, No. 1, Jan 1984, pp.102-128.

[11] R.E Zeimer and W.H.Tranter (1985), "Principles of Communication", Houghton Mifflin Company, Boston.

[12] Bernard Sklar (1988), "Digital Communications Fundamentals and Applications", Prentice Hall, New Jersey.

VITA

Prabir K. Das was born in Guwahati, India in the year 1961. He did his schooling in the Don Bosco High School, Guwahati. After completing his Pre-University studies in the Cotton College, Guwahati he entered the Assam Engineering College, Guwahati. He graduated from the Gauhati University in July 1984 with a major in Electrical Engineering. He worked for the Government of Assam, India as Deputy Superintendant of Police (Communication) and later as Additional Superintendant of Police (Communication) before entering the University of Tennessee, Knoxville in August 1989. He completed his studies leading to the Master of Science degree with a major in Electrical Engineering in August 1991.