Interactive communication system simulator ICOSS.

Abdul-Wahab, A. S. M. S.

Award date:
1979

Awarding institution:
University of Bath

INTERACTIVE COMMUNICATION SYSTEM SIMULATOR ICOSS

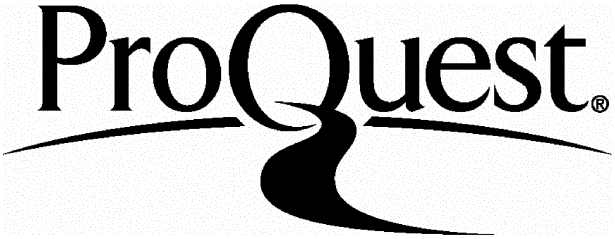
submitted by A S M S ABDUL-WAHAB

for the degree of PhD

of the University of Bath

1979

COPYRIGHT



SUMMARY

This work is concerned with the design of a general-purpose time-domain interactive communication-system simulator ICOSS. Signal processing modules may be interconnected in any order, and module control parameters, as well as the system control parameters can be varied while the simulation is running. Editing of the system structure (inserting or deleting modules) can be done on-line.

The ultimate objective is to set-up a communication system simulation working on an on-line basis, with an interactive capability providing the engineer in a research environment with a bench tool, which complements the hardware apparatus.

Once the prototype of ICOSS was developed, an investigation was made into one of the main areas of further development, namely, processing speed, in which a dedicated processor containing a signal processing module, comprising of microprocessor controlled unit, is coupled with the main computer, where ICOSS resides .

The simulator novelty in engineering and research was tested, by using problems involving feedback links, namely, interference in a phase-lock loop and performance of fast acquisition phase-lock loop. These provided confirmation and deeper understanding of experimental work as well as proving that ICOSS was working

correctly.

To implement the full system requires computing power and equipment which was unavailable. Therefore, only the prototype version was implemented.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 COMMUNICATION SYSTEMS

1.1.1 Definition and Characteristic [35]

Communication system design involves a complex interplay between the hazardous transmission environment and the skilful use of modulation techniques and electronic hardware. Many associated problems have successfully been solved by the traditional methods of experience and intuitive understanding, but it has always been clear that better judgements could be made and more realistic predictions of performance constructed if readily usable analytical techniques were available to the communication system designer. General analytical solutions are only available for the simplest of cases however, because of the general difficulties of the communication environment. Communication theory relates signalling speed, bandwidth and noise, which assumes well-behaved but non-realisable channel frequency characteristics; and the extension of these principles to real life is left to intuition. Non-linear circuit elements such as blankers and chippers, leave a certain amount of vagueness in receiver calculations, particularly when allied with a variety of modulation styles. Communication filters are generally of high order,

typically fifth or greater, which, if several are included in one analysis, lead to complicated equations. Filter characteristics are consequently chosen for the convenience of their analytical description, rather than their suitability for the job in hand, and although optimum filters are capable of being designed for specific applications, it is difficult to predict how the filter will behave when conditions have altered from those assumed for the optimisation.

1.1.2 Areas of problems in communication systems[9]

The areas of problems in communication systems are wide and ever increasing, but for the present discussion purposes, the following main areas of problems are defined as the major communication problems.

(a) Stochastic: long averaging times, or large ensembles involved, eg phase jitter and digit synchronisation studies, frequency stability studies.

(b) Signal analysis problems: long averaging times, long records, eg probability distribution generation, correlation function estimation, power and energy density spectra: P(f) and E(f) etc, for real signals, eg speech.

(c) Interactive or real time problems: long data record involved, eg optimisation studies, of filter bandwidth.

(d) Deterministic problems: short computation times

involved, eg transient studies such as "speed of lock" of a phase-lock receiver, distortion tests.

1.2 SIMULATION

1.2.1 System Studies

Analytical expertise in communication systems is both desirable and difficult to acquire. The alternative to exact analysis is to assemble the proposed system and investigate its behaviour experimentally; such investigations have been carried out since the science of electronics began. While providing some of the answers, this approach is expensive and also leaves many unsatisfied queries. What is really happening inside that mixer circuit? How would the system behave with slightly different filters? How sensitive is the system to circuit parameter changes? Computer simulation of communication may be the solution to these queries. But before introducing that, a brief introduction to the system simulation in general.

1.2.2 System Simulation [12]

The need for the simulation of a system may arise in several ways. An analytical solution or approximation solution to a problem may have been found and some form of corroboration be required. In some cases this is most easily obtained from a hardware model, but with more complex systems, computer simulation may be a more attractive

proposition from the points of view of time, accuracy and economy. In computer simulation parameters can be modified, variables monitored, curve fitting and regression analysis performed, and statistics of the system performance obtained.

A simulation may range from a pure numerical calculation from analytical results, to partial simulation where some parts of the system are modelled analytically and the rest simulated, to a full simulation in which every element in the system is modelled and simulated. In the interests of efficiency it is usually desirable to describe as many parts of the system as possible by analytical expressions.

A simulation may be run serially in time, where the response at each instant of time is computed recursively from the state of the system and the inputs at that time. Alternatively, it may be possible to compute the overall time response in parallel, usually by transform techniques. The parallel simulation can be faster, but is difficult to implement if there are feedback paths in the system and may require a large amount of storage to simultaneously store the values of variables at every instant of time.

The use of digital computers for the simulation of analogue system is a wide field, and over recent years a large amount of literature has been published. The literature concerning simulation systems tends to fall into several main areas[4] :

(*) Discrete simulation: concerning discrete change simulation; eg queueing problem.

(*) Simulation designed around a particular problem: eg FFT package.

(*) Continuous (analogue) simulation methods: the importance of these methods is in their historical development, their organisation and the structure of their basic elements.

(*) Existing methods for the simulation of communication system: these are standard packages, see later.

1.2.3 Communication system simulation

Simulation of communication system modules can take place either in terms of frequency or time domain. The difference between the two lies in the manner in which filters are described. A frequency-domain simulator specifies filters in terms of their frequency response, thus enabling ideal filters to be included. In a time-domain simulation, filters would be described through difference equations operating on a succession of signal time-samples. Both techniques suffer from the usual troubles introduced by signal sampling, limited frequency range and aliasing, but their time domain performance differs widely. The input to each transformation in the frequency-domain simulation is N time samples and the corresponding frequency scale has only N discrete points. The time scale is unlimited in the case of the time-domain

simulation, and the frequency scale is continuous up to the sampling frequency.

Communication system simulation can be classified into two types:

(a) Special-purpose, in which the simulation is specific to a particular problem: most simulation for laboratory and research work falls within this type. The engineer, who will write the special-purpose simulation, must be a skilled programmer and sufficiently familiar with the mechanics of simulation to describe adequately his problem in terms which allow accurate simulation.

(b) General-purpose: digital simulation allows a general approach to be made, where the programming effort is applied initially to construct in software a set of processing modules and some kind of data structure, which allows them to be strung together and signals and effective measuring instruments applied. Some familiarity with the basic concepts of programming will inevitably be required (although the tendency is to minimise them as much as possible), and also an understanding of the principles of the particular simulator. This type of simulation, the general purpose, will be adopted thereafter.

To simulate a given communication system, the block diagram of the system is constructed using processing

modules (eg amplifiers, filters etc , ie having different fundamental functions); and then a certain form of connection of the blocks in the diagram is written in terms of the functions of the blocks and certain parameters. If the block diagram and the expression of these connections are fed as the source program into the simulation, then the complier in the simulation system generates the program written in the computer language, say Fortran IV, for the simulation of the given system.

1.3 THESIS

1.3.1 Designing a new simulator, ICOSS

There are a large number of communication system simulation packages based on one technique or the other to perform a particular task or tackle one or more areas of communication system problems. A large number of papers and literature cover this subject as will be shown in the next chapter. However, communication system simulation does not stop at any particular boundary, but it progresses and becomes more and more ambitious. With the introduction of fast microprocessor and minicomputer systems, a new approach to communication system simulation is therefore needed in order to achieve the most ambitious requirements. A time domain simulator: the Interactive Communication System Simulator (ICOSS) is to be designed with this background in mind. Its main objectives are: being able to change the simulated model control parameters while in the running mode, being able to edit its module structure (block diagram), be portable, and operate

interactively for real-time simulation. Also to provide the communication engineer with a bench tool which will prove vital in research and development laboratories.

In the pursuit of designing a new communication system simulator, the following procedure is followed in this thesis. A survey of communication system simulators is made in order to present the current state of the subject. They are then classified into a number of areas according to their principle of operation, type of problem they are capable of solving, and system set-up (ie computer capability and supporting equipment available). A summary of the objectives in the design of the new simulator are formulated, which must satisfy the ambitious requirements of a typical user as much as possible. A number of options are available to implement those objectives: choice of programming language, generality of application, domain (frequency or time) etc. Finally, once the new simulator is implemented, its credibility is examined by testing its functions in solving the type of problems it is designed for.

1.3.2 Outlook and modifications

The intention of this work is not only to design and implement a new digital communication system simulator of unique characteristics, but to lay the foundation for simulation system with wider applications as outlined earlier. However, the inherent problem of real time simulation in

which fast processing is essential, can be overcome by either large or fast computer system, or alternatively processing parts of the simulation by fast dedicated processing working in conjunction with the mini or medium size computer. Although the objectives of the prototype version of ICOSS are achieved, the overall and final version of ICOSS system, running in real-time were not fully implemented, because of either the limitation of time or facilities. Therefore, there is still further work to be done, which will be outlined in Chapter 6.

1.3.3 Application

The credibility of any new simulator is measured by its fulfillment of the objectives laid in the design, as well as its power in solving the type of problems it is designed for. Since one of the main features of time-domain simulators is the ability to simulate communication systems containing feedback links, phase-lock loop performance and the problems associated with it are an ideal testing case, which will be fully utilised.

CHAPTER 2

SIMULATION TECHNIQUES AND SIMULATORS

2.1 INTRODUCTION

2.1.1 Historical Background

Simulation is the building of a model of a device or system[38]. Models are useful in that they provide a means of testing ideas and designs without the complication or expense of building a prototype system; and modelling is widely accepted in engineering. In some fields the models are physical devices, such as small scale versions of the actual system. When a system can be defined mathematically, modelling can proceed from a conceptual view point and the model need not be practically realisable. Such is the type of simulation that can be used for communication or signal processing systems.

Computer simulation of systems in general is a wide subject, each case influenced by the limits of computer capability and particular problem at hand, and there are numerous articles and books on the subject*.
Limiting the discussion to communication (or signal processing) system simulations, there are a number of approaches, depending on:

---

* Refs 9-12, 15,16,21,24-26,29,30,35,38,40,41,48

(*) The simulated system itself and the degree of accuracy required of the simulator, as well as the degree of similarity with the actual system.

(*) The computer at hand and its capabilities (memory size, execution time etc).

(*) The degree of generality required by the simulator, ie will the simulator be used in a different environment or will it only be used for one particular problem.

(*) Continuous time or discrete time systems.

Historically, signal processing system (SPS) simulations started with analogue computers [38], in which the system is modelled by differential equations, in which integrators, adders etc make up the system, and the problems can be solved quite easily many times over, using simple configurations. There are big advantages in analogue computer simulations, mainly:

(*) Fast speed of action, since there is only hardware involved, and the results are obtained immediately.

(*) Simple technique in solving certain difficult problems.

(*) Easily used by engineers.

However, there are disadvantages as well, mainly:

(*) Difficult to set up for large problems.

(*) Variables must be scaled.

(*) Non-linear operations: such as multipliers, function generators are possible but have limited dynamic range.

(*) Only continuous time system, whose theoretical analysis is based on the solution of differential equations, can be simulated by this method; ie discrete time systems cannot be simulated.

(*) Limited in order of system, by number of integrators, ...

With progress toward developing digital computers, the analogue computers started to lose their predominance in the field of SPS simulation. Although the digital computer is flexible and capable of being programmed for many problems, such as solving discrete-time systems, whose analysis is based on difference equations (see later), yet the execution time is increased considerably as compared with analogue computer simulation [16], vis 100:1 ratio.

In order to overcome the time limit, hybrid computers were introduced in to combine the two types (analogue/digital) and obtain the advantages of both. With the simulation techniques of communication systems becoming more and more ambitious, the hybrid computers have been replaced by "Giant" computers [16], but the latter solution has proved to be rather expensive for the vast majority of

applications.[16] The ultimate solution, with real-time on-line system simulations in mind, is to combine a digital computer of a medium size, with mini-computer network with storage and interfacing operated separately. In the coming sections a development to that end is described.

2.1.2 Simulation Techniques

In order to simulate a system, one must have a clear understanding of the overall content of the communication system itself, as well as what is required from the simulation. In signal processing system simulation, the system can be looked at as a group of sections and sub-sections interlinked together. These can be summarised briefly as follows, for the example system shown in Fig 2.1.1:

(a) Signal processing modules (M): eg signal generators, filters etc.

To simulate those modules, the following parameters have to be determined.

(i) Module internal and external coefficients, ie local and global (see Chapter 3).

(ii) Inputs/outputs node relationship of the module.

(b) The state variable of the system.

(c) The running mode (procedure).

(d) Output signal measurements and evaluations.

The common denominator in practically all simulators is to present the signal processing system as near to the physical system as possible, and then utilising it in situation which would be impractical or difficult in the real system. The emphasis, therefore, is how simple the simulator appears to the user, in terms of familiarity to him as a bench tool, rather than learning how the simulator has been constructed. This is in contrast to the earlier work of signal processing system simulation[26].

The simulation of the above signal processing system can take any of the following forms depending on the application and tests required; these are summarised as:

1. Frequency domain based simulation.
2. Time domain based simulation.
3. On-line operation.
4. Off-line operation.

Looking through the development of signal processing system simulators(*), which are mainly based on block diagram simulation (see below), one can see clearly the trend moving from being heavily dependent on programming technique, with specialist approach on the off-line basis, to that which is closer to the real physical situation,

---

* Refs 11,16,21,26,29,30,35,41,46

the on-line working basis; especially with the introduction of high speed computers.

2.1.3 The Block Diagram Technique

There are numerous simulators basing their analysis on the so-called block diagram technique(*). It is so called because the simulation is based on the ordinary block diagram representation of a system. For this reason, this technique is most suited to the simulation of communication systems, and shall be adopted for the present work. The main feature of this technique and method of implementation is briefly introduced at this stage, and a more rigorous examination will be given in later sections. The communication system is completely defined as cascaded blocks or units representing the signal processing modules (MO, Ml, etc Fig 2.1.1), with the necessary interconnections for the feedback or cross-over paths. Hence when simulating such a simple system, the following items have to be treated:

(a) The signal processing modules MO, Ml etc: The transfer function of each module must be clearly defined, and a subroutine is created in the usual way.

(b) The Local control parameters of those modules, such as the cut-off frequency of a digital filter, or the gain of an amplifier etc.

(c) Input/output nodes of each module (multiple levels).

(d) The inter-link between the modules (input/output nodes connections), which may take number of forms(+).

---

* Refs 10,11,16,21,25,26,35,46
\+ Refs 11,16,35,41

They have to be uniquely defined; but once adopted they have to be precisely followed.

(e) The "global" control parameters, ie the parameters which control more than one module, such as a sampling frequency of a digital communication simulation.

(f) The type of output device which evaluates the signal at any node within the system, such as the measurement of a mean value of a signal etc.

There are three distinct stages which constitute the simulating procedure of this block diagram technique, they are:

1. The construction stage: in which the communication system is defined and its parameters are supplied to the simulator. This is implemented in various ways depending on the designer(*). However, it should be implemented in as simple a way as possible for the reason is that the user may have limited or no programming knowledge or experience at all[15,35]. Appendix A shows some samples of implementation of a number of important simulators. During this stage also a number of tables, strings and stacks are generated in order to set the simulator for the following stage.

2. The running stage: in which the incoming signal is processed through the system in orderly manner. This requires efficient storage arrangement of various parameters

---

* Refs 11,16,35

needed for signal processing as well as searching procedure with minimum elapsed time.

3. The output stage: in which the output signal at any node of interest is examined in order to evaluate the behaviour of the simulated communication system.

2.1.4 In this Chapter

The signal processing system techniques are discussed in more detail and presented as the subject developed historically. The limitation and misgiving of the various techniques mentioned will be discussed, giving the lead to a new approach, which will be called ICOSS, the Intensive Communication System Simulator. The latter will be introduced only briefly, since it will be fully described in Chapter 3.

2.2 IMPLEMENTATION: USER SIDE:

One of the big problems in communication system simulations is how to present the simulated system in simple terms recognisable by a user with limited or no programming experience. Various approaches were made to overcome this problem, which can be found in literature(*). These can be grouped according to application, machines and period of implementations. However there are two main ways of presenting the simulated communication system to

---

* Refs 11,16,21,26,35,46

the signal processing simulators; they are:

(a) The block diagram method: which is commonly used and most suited for digital computer simulators.

(b) The differential equations method: which is implemented for analogue (operational)computer oriented simulators.

In the following, a brief description of both methods and their limitations as implemented by well-known simulators is given, and the ideal solution is forecast. Typical examples are presented in Appendix A.

2.2.1 Typical communication system

The object is to translate the representation of the typical communication system shown in Fig 2.2.1 to a set of computer instructions. The translation depends mainly on the type of simulation and computer in use. The discussion will be restricted to two types only, the analogue computer, and the digital computer, since any other type will be the combination of the two.

The digital-microcomputer network will be mentioned in later chapters.

(a) The analogue computer oriented simulators: The differential equation method.
The procedure is summarised as follows:

(i) The complete system is defined as a set of differential equations, and their elements (integrators, delay, adders etc), ie the conversion is from block diagram of the system to differential equations.

(ii) The simulation is implemented by an orderly connection of these elements.

Generally, a patching panel is needed for the analogue computer, which can be programmed in the case of hybrid computer.

(b) The digital computers oriented simulators: The block diagram method.
Historically BLODI[26] was one of the early simulators in which BLODI program accepts input program written in BLODI language. The latter corresponds closely to an engineer's block diagram of a circuit. The input code consists essentially of designating the connectivity of a number of boxes drawn from an alphabet of about 30 types.

The object program produced by BLODI consists of 3 parts

(i) the prefix which sets up the logic for the main loop

(ii) the main loop, which is executed once for each sample processed

(iii) the suffix, which causes the main loop to be repeated the proper number of times, empties output buffers, fills input buffers etc.

Example: for system shown in Figure 2.2.2

line 1 :	SUM	ADR	BUFF
line 2 :	BUFF	FLT	a_1, a_2...parameters of filters, DELAY
line 3 :	DELAY	DEL	....
etc			

In later years the BLODI was modified to become BLODIB.[21] The language of the latter is designed for programming sample-data system which may be represented either in block diagram form OR in the mathematical representation

of the Z-transform calculus. Therefore, although it is basically for sample-data system, it may be used for sample-data approximation to continuous (analogue) systems. The procedure in simulating a system is summarised as follows:

(i) the determination of an appropriate discrete representation for the system to be studied
(ii) the preparation and compilation of a BLODIB computer program which causes the computer to perform the same operations as would be performed by the actual system
(iii) the digitalisation of a real speech signal for processing by the computer,

There are 3 ways of representing the system

(i) by MACROS or SUPER: defines new type of block (for basic type available)
(ii) by SSUBR: which allows block of BLODIB coding to be used as an external "module" to a main BLODIB. But SSUBR are coded and compiled separately; then loaded for use by the main program.
(iii) by SUBR, which permits coding an entire simulation so that it can be controlled by a main or "executive" program.

Example:

```
                    Inputs      Control Parameters
ADB     MACRO     I1,I2,I3,I4     A1,AO,B1,B2
IN      MIP       1,DEF
          .
          .
          .
        END
```

Another type of presentation was made by another group of simulators, and WASP[11] is a typical example. In this a fixed structure is established for the analysis of waveforms and spectra in the communication systems and then building on this framework a library of electronic or electromechanical modules of the type needed for the specific application. The procedure for WASP simulation is as follows:

(i) Draw communication system as block diagram (each block must be one of the library modules)

(ii) Number each node and decide on the node for the output device connection

(iii) Input data

Example:

```
                module name as it appears in library
n       module 1,          module 2,  - - -      /  comment or remark
      - - - - - - - - ,    module n              /  ..................
1       module 1           Parameter  - - -      /  ..................
        - - - - - - - - - - - - - - - - - -
2       module 2           - - - -
        - - - - - -
              .
              .
              .
              .
              .
n       module n           - - - - - - - -
        - - - - - - - - - - - - - - - - - -
XY      Title                                    / X:module number, whose output is
                                                     required to be tested
                                                 / Y:type of output device required
                                                 / Title: title on graph or results
                                                          (if applicable)
```

Finally, yet another approach which is relevant to this discussion, is the one made by SYSTID [16] . The latter is a time domain simulator which relies on cards for the data input and each group of cards represents a communication system. Its language processor translates the simple English language user command supplied on cards, and links element descriptions and topology into the Fortran code necessary to establish a digital filter equivalent of each link. The data structure of this system is as shown in Fig 2.2.3.

Example: A model which squares a signal:

```
MODE 1      Ø  Example
INPUT       Ø  $ * $ Ø    OUTPUT
INPUT       Ø  SIN ($) Ø  NODE 1
END         Ø
```

2.2.2 Limitations

There are many ways of simulating communication systems other than the ones mentioned above, but they all have some limitation. Some of these limitations are summarised below:

(i) Batch operated rather than interactive, so that data representing the communication system block diagram is supplied to the computer separately for each run.

(ii) Modification of system structure once supplied to the simulator is difficult to achieve

(iii) Variation of signal processing modules parameters is not direct, if at all

(iv) The simulated system is not clearly related to the original communication system block diagram. Hence the user is not able to look upon the simulator as another bench tool identical to the simulated communication system.

2.2.3 Projection

In order to achieve as near an ideal solution of this problem as possible, the following must be fulfilled:

(i) Overcome the limitation of (2.2.2) above

(ii) Establish a peripheral device on which the simulated communication system is drawn on it directly, and the user could modify the system

readily on the peripheral device. This will eliminate completely the difficulty of transforming the electrical block diagram onto the set rules of the simulator.

It is hoped that ICOSS will fulfil these conditions when it is finally completed.

2.3 IMPLEMENTATION: COMPUTER SIDE

In the design of an efficient general purpose signal processing simulator, the designer has to aim for the following essential targets:

(*) Efficient utilisation of computer memory, speed of execution (cycle time), and computer facilities.

(*) Efficient programming without ambiguity and with minimum redundancy of occupied memory.

(*) Program the simulator to make it portable.

(*) Program routines to represent a complete set of general SPS modules.

Thence, in the design of SPS simulator on a digital computer, the following detailed points must be carefully considered:

1. Structured programming: in which the various sub-

routines which make up the simulator are interconnected without nesting and the overall program of the simulator can be modified by adding or taking away subroutines with minimum complication.

2. The use of simulation languages: presenting a simulator to a user in a language easy to understand, means creating a new computer "language". Since there exist nowadays communication system simulation language such as WASP[11] and CSSL[3], then it will be beneficial to utilise such languages, if they are general enough.

3. The choice of a program language: since there are a number of computer languages such as Fortran, Algol, etc, then the choice must be made for a suitable language, eg for engineering application Fortran IV is most suitable and provides portability. It might be desirable to use assembler language or machine code in some part of the simulator program, in which time is of prime importance (see Chapter 5).

4. Signal processing modules transfer functions: must represent a fundamental set, enabling all common communication systems to be simulated.

5. Output devices can take many forms:one form is to have a real hardware peripheral or oscilloscope in the case of real-time operation. Or the output device can be simulated, which is the case in the majority of situations; eg RMS meter etc.

6. Library construction: for signal processing modules, data base (parameters, state variables, pointers etc).

Efficient arrangement is of vital importance in order to utilise the limited computer memory economically. In the course of running the simulator, there will be samples, pointers, state variables, links etc which have to be stored in stacks and there will be a relationship between the contents of one stack with another, which will be consulted during the process of execution. Therefore an efficient method in either space, or speed, ie access time depending on the simulation whether real time, stochastic or off-line simulation, must be devised.

7. Search procedure of tables (library): Serial or Hash[39] techniques depending on the situation, has to be efficient, since choosing the right technique means minimising the execution time, which is of vital importance in the on-line and stochastic simulation, in which large averaging times or large ensembles are involved such as fading problems (error rates studies) multi-channel interference problems (see later).

8. Storage of samples: at some stage within the execution, it will be found necessary to store block of samples of the signal at any node of the system so as to process it further at a later time, internally or externally.

9. Memory allocations (real and image): it is always beneficial to plan and utilise computer memory capacity according to the simulator need (program, tables etc).

10. Efficient processing (arithmetic and logic): one of the chief functions of a digital computer is to perform arithmetic operation upon numbers and series of numbers can represent the instantaneous amplitude of a signal. Therefore, if the correct arithmetic operations are used, sequences of numbers could be converted into other sequences, in such a way that the overall operation models some element in a communications systems. Such is the basis of simulation by digital computer[38]. Therefore, the computer arithmetic capability is an important factor when considering the design of a simulator. This is more apparent for the on-line real time simulation case where the time saving is of vital importance, eg

    If one instruction takes 1 m second then for 1000 instructions would take 1 second, giving a sampling frequency = 1Hz which is highly unacceptable.

    It is preferred to have a high speed arithmetic unit in conjunction with computer system to solve this problem.

11. Multiprocessing attempts: Since communication simulation sometimes involves multiple processing and since time saving is an important factor in simulation then it is desirable to multiprocessing simulation but with additional problems in organisation. This can be achieved by software or hardware or both. The latter point will be elaborated in later Chapters.

12. Cross-reference of samples (nodes): as signal samples are flowing within the simulator passing through

multiple input/output nodes, it is essential that at multiple inputs, each sample has the same time registration. Otherwise the simulation is invalid. Special care must be taken regarding this problem.

14. Real-time simulation: this type of simulation has a number of criteria, namely:

    (a) Real-time clock: this is preferably an external clock, which can be varied depending on the sampling frequency. Its accuracy therefore has a major factor on the running of the simulation.

    (b) Analogue/Digital and Digital/Analogue converters: the number of levels and digits in either direction, ie quantisation error, will influence the accuracy of the simulation. Therefore, it must be taken into account and made as small as possible. Wide dynamic range logarithmic-converters may be needed.

15. Continuous signal as represented by a digital system: the correspondence between the digital model and the continuous system depends on:

    (a) Satisfying the sampling theorem

    (b) Correctly relating the z-transform of digital filter structures to the frequency response of the continuous prototype.

2.4 FREQUENCY DOMAIN SIMULATION

2.4.1 Introduction

Ever since the development of computer procedure for performing the Fast Fourier Transform (FFT), the communication engineer has been provided with a simple method of achieving frequency domain processing (wave filtering) without the need to acquire skills in the technique of digital filtering[9] . Most communication simulators today are based on this type of domain and are well covered by the literature (*).

2.4.2 Principle of Operation

This type of simulation is based on processing a block of signal samples at a time. When filtering is required, these samples will be transformed to the frequency domain by applying the Discrete Fourier Transform (DFT) to this block of samples, then transforming to time domain by applying Inverse Discrete Fourier Transform (IDFT) for further non-linear processing. The number of samples in a block must be large enough to give a good resolution. However if there is a very long signal record to be processed, then successive sets of blocks are processed independently, and the processed blocks are joined up together so as to produce the continuation of signal record as before. However, this joining up process is complicated by the fact that this filtering performs a circular convolution instead of an aperiodic one[49] , so that a

---

* Refs 11,35,41

certain proportion of each segment must be discarded.

2.4.3 Execution procedure

(a) Obtain the filter response from either a pole-zero description, or a numerical description of a measured response curve.
(b) Take N points of input signals.
(c) Transform into the frequency domain using DFT.
(d) Multiply each sample (point) by the corresponding filter response.
(e) Transform resulting spectrum back to time domain using IDFT for further processing.

2.4.4 System Analysis

A prototype system containing the basic elements of a communication system, ie a filter, nonlinear module and a linear module, Fig 2.4.4(a), has a corresponding structure for frequency domain simulation, as shown in Fig 2.4.4(b). The signal analysis (signal calculations at nodes concerned), for an input signal X(t) having a bandwidth ω Hz is as follows:

* At node (1): analogue to digital conversion

$$x(t) = \sum_{-\infty}^{+\infty} x(nT)\ \text{sinc}\ (\frac{t}{T} - n) \qquad (1)$$

where T = the sampling frequency.

There is a major restriction which must be observed:

$T << \frac{1}{2B}$ where B is the highest frequency component of the input signal (= ω in this case). This is the limitation imposed by the sampling theorem[8], so that folding back into the fundamental band, thus disturbing the representation, does not take place, also to overcome aliasing (or overlap or high frequency impersonation[6]).

* At node (2): Conversion to frequency domain

The conversion is performed using discrete fourier transform (DFT):

$$X(K\delta) = \frac{1}{N} \sum_{n=0}^{N-1} x(nT) \exp - j \frac{2\pi Kn}{N}$$

$K = 0,1,2,....,N-1$

* At node (3): Filtering action

The filter coefficients are either stored (storing N points), or generated (N points), using the zeros and poles of the filter. The output signal $Y(K\delta)$ is determined by multiplying signal component by the filter spectrum

$$\dot{Y}(K\delta) = H(K\delta) \cdot X(K\delta)$$

* At node (4): Conversion back to time domain

The conversion is performed using the inverse discrete fourier transform (IDFT)

$$y_1(nT) = \sum_{K=O}^{N-1} Y(K\delta) \quad \exp j \frac{2\pi Kn}{N}$$

$n = 0,1,2,....,N-1$

* At node (5): Time-invariant non-linear module

$$y_2(nT) = g(y_1(n_1T))$$

* At node (6): Time-invariant linear module

$$y_3(nT) = z(nT).y_2(nT)$$

This is a simple multiplication.

2.4.5 Characteristics

(a) Systems with feedback links cannot be handled since data is processed by the FFT in segments.
(b) Slow, hence it cannot be used for stochastic simulation, unless hardware FFT machine is employed.
(c) Resolution is limited by fixed number of points (samples).
(d) Easier to handle complex signals than time domain

based simulators.

(e) Any type of computer can be utilised, but as an off-line oriented simulator.

2.4.6 Utilisation

Wide range of problems are being solved using this technique and could be found in literature, and some languages have been developed such as WASP [11], MODSIM [41] and SIGSIM [35]

2.4.7 Scaling

For time scale of signal block consisting of N samples at spacing of T seconds, and frequency scale of signal block consisting of N samples at spacing of δ frequency,

then $NT\delta = 1$

Thus, for a given number of points $T \propto \frac{1}{\delta}$ (reciprocal relationship).

But frequency scale is limited to $\frac{1}{2}$ the sampling frequency

then $F_{max} = \frac{1}{2T} = \frac{N\delta}{2}$

2.5 TIME DOMAIN SIMULATION

2.5.1 Basic principles and execution procedure

The conversion from time to frequency domain for the purpose of filtering signal samples becomes unnecessary

in time domain simulation, because the filter itself is simulated. This is done by correctly relating the z-transform of digital filter structure to the frequency response of the continuous prototype[9], and using the difference equation version of the filter response. However, digital filter cannot model the corresponding analogue filter in both impulse and frequency response, but it is possible to make the correspondence close with careful design[49]. There is no limitation to the number of points and for this reason it is suitable for on-line real-time working simulation. It is possible to simulate systems with feedback links. However, state variables of the system have to be clearly defined.

2.5.2 System state variables

The state variables of a discrete-time system are the minimum set of variables which define the overall state of any system (linear or non-linear) at instant of time[9]. They are useful for the following reasons:

(*) It may be necessary to examine the behaviour of all relevant signals in a system.

(*) The need for a more general system description to treat multiple inputs and outputs.

(*) The need for a more compact system description in the study of complex systems.

(*) Often in the study of systems, only a general (qualitative) description of system behaviour is required.

The general mathematical formulation of the state variables concept is as follows[17] :-

Assume a system, whose outputs represented by (y) and inputs by u, is defined by

$$y(k) + b_1\, y(k-1) + \ldots + b_n\, y(k-n) = a_o\, u(k) \qquad (1)$$

for $K \geq 0$

Defining the state variables as: $x_i(K)$, $i = 1,2,\ldots,n$

thus:
$$x_1(k) = y(k-n)$$
$$x_2(k) = y(k-n+1)$$
$$\vdots$$
$$x_n(k) = y(k-1)$$

Hence by substitution:
$$x_1(k+1) = x_2(k)$$
$$x_2(k+1) = x_3(k)$$
$$\vdots$$
$$x_{n-1}(k+1) = x_n(k)$$

and $$x_n(k) = y(k) = a_o\, u(k) - b_n x_1(k) \ldots - b_1 x_n(k)$$

This equation is represented by block diagram shown in Fig 2.5.1.

Hence, using vectors $$[x(k+1)] = [A][x(k)] + [B] \cdot u(k) \qquad (2)$$

where:

$$[x(k)] = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} , \quad [A] = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ & & & & \\ 0 & 0 & 0 & \ldots & 1 \\ -b_n & -b_{n-1} & & & -b_1 \end{bmatrix} , \quad [B] = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ a_o \end{bmatrix}$$

NOTE

A system is linear if and only if superposition and homogeneity hold[17]: ie

$$\alpha\, x_1(t) + \beta\, x_2(t) \rightarrow \alpha\, y_1(t) + \beta\, y_2(t)$$

In general for $y(t) = H[\,x(t)\,]$,
a system is linear if and only if H is a linear transformation: ie

$$H[\alpha\, x_1(t) + \beta\, x_2(t)] = \alpha\, H[x_1(t)] + \beta\, H[x_2(t)]$$

The output in terms of state variables:

$$y(k) = [C] . [x(k)] + [D]\ u(k) \qquad (3)$$

where $[C] = \left[-b_n \quad -b_{n-1} \ldots . -b_1\right]$ and $[D] = a_o$

Rewriting in state variable representation:

The general equations for linear and non-linear systems

$$[x(k+1)] = [A] . [x(k)] + [B] . u(k)$$
$$y(k) = [C] . [x(k)] + [D] . u(k)$$

The above formulations can be utilised in the simulation of digital filters, eg a second-order Butterworth low-pass digital filter with a cut-off frequency ω_c = 0.3249 rad/sec, has a transfer function given by [1] :

$$H(z^{-1}) = 0.0676 \left\{ \frac{1 + 2z^{-1} + z^{-2}}{1 - 1.142\ z^{-1} + 0.412\ z^{-2}} \right\}$$

Applying difference equation method :

$$y(k) = x(k) + 2\ x(k-1) + x(k-2) + 1.142\ y(k-1) - 0.412\ y(k-2)$$

This is represented by a block diagram, Fig 2.5.2.

Using the above formulation for state variables:

$$\left.\begin{aligned} v_1(k) &= y(k-2) \\ v_2(k) &= y(k-1) \end{aligned}\right\} \text{ for section (a) of Fig 2.5.2}$$

$$\left.\begin{aligned} \omega_1(k) &= x(k-2) \\ \omega_2(k) &= x(k-1) \end{aligned}\right\} \text{ for section (b) of Fig 2.5.2}$$

Using equation (1), for section (a) of the filter: the state variables

$$[v(k+1)] = [A] \cdot [v(k)] + [B] \cdot u(k)$$

where $[v(k)] = \begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix}$, $[A] = \begin{bmatrix} 0 & 1 \\ 1.142 & -0.412 \end{bmatrix}$,

state variables $[\omega]$ $[B] = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Using equation (3), the output y(k) in term of state variables for section (a) of the filter is:

$$y(k) = [C] \; [v(k)] + [D] \cdot u(k)$$

where $[C] = \begin{bmatrix} 1.142 & -0.412 \end{bmatrix}$, $[D] = 1$

Repeating calculations for section (b) of the filter and by inspection:

State variables:

$$[w(k+1)] = [A] \; [w(k)] + [B] \cdot x(k)$$

$$[w(k)] = \begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix}, \quad [A] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad [B] = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The output in terms of state variables:

$$u(k) = [C] \cdot [w(k)] + [D] \cdot x(k)$$

$$[C] = [2 \quad 1] \quad , \quad [D] = 1$$

Solution of the general equations (1) and (2):

If the vector $x(k_o)$ for some k_o is known, then $x(k)$ can be computed, and hence the output $y(k)$, for any $k \geq k_o$ in terms of the input sequence $u(k_o)$, $u(k_o+1),\ldots,u(k)$. Thus the vector x satisfies the definition for the state of a system. This n-dimensional vector is equivalent to the (n) initial conditions needed to solve the difference equation (1) in terms of input u. Accordingly the following definitions are obtained:

X state vector of the system

A state or system matrix: nxn matrix which relate the state at index k+1 to the state at k.

Therefore there are two areas to consider when applying the above formulations to the problems of time domain, communication system simulation:

(i) Modules containing delay elements, such as filters, differentiators etc.

(ii) The multiple input/output modules, with feedback links.

These two items will be discussed in detail in Chapter 3, as part of the construction and running procedures of a simulated communication system, in the newly designed simulator ICOSS.

2.5.3 Characteristics

1. More suitable for stochastic simulation than frequency domain simulation.
2. On-line real time working is possible.
3. Settling time is long, unless time scaling is introduced, see later.
4. Non-linear system elements response are treated in an accurate straightforward way.

2.5.4 General comments

Time domain simulation was classically applied to analogue computations for control systems. With the introduction of fast digital computers economic considerations favour the digital computer simulation opposed to analogue simulation [16]. Comparison between analogue and digital computers is as follows:

Analogue computers

(a) Significant set-up and check-out time for initialisation.
(b) Additional time for modification of original situation.
(c) Extremely low unit run cost, even for wide bandwidth systems.

(d) Degradation of electronic elements may create large solution errors.

Digital computers

(a) Assuming program developed and debugged, then

(i) Minimum set-up time and very limited initial checking.

(ii) Negligible additional time for parameters and topological variations.

(b) High hourly cost.

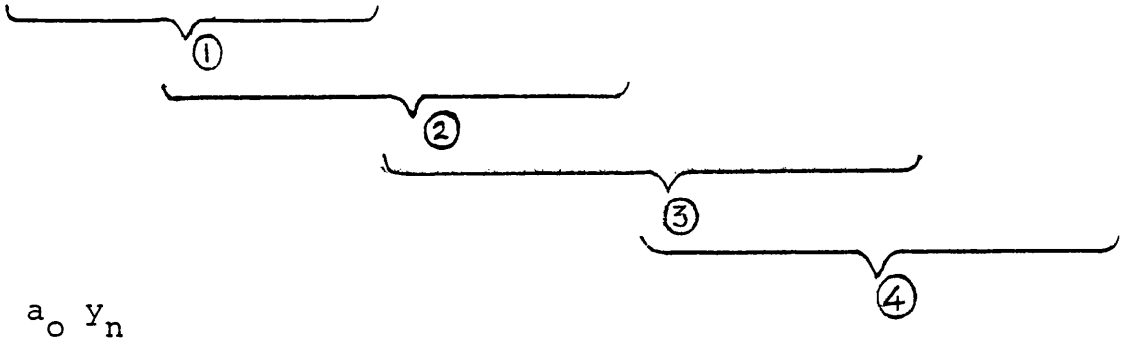
2.5.5 Execution Time

Assuming a digital filter made up of n second order segments, Fig 2.5.3, in which the transfer function of one segment h_i is given by[20]

$$h_i = \frac{1 + a_i Z^{-1} + b_i Z^{-2}}{1 + c_i Z^{-1} + d_i Z^{-2}}$$

giving a difference equation

$$y_i(k) = x_i(k) + a_i x_i(k-1) + b_i x_i(k-2) - c_i y_i(k-1) - d_i y_i(k-2)$$



$$y_{out} = a_o y_n$$

Defining an operation as one multiply/add action, then there are 4 operations (second order segment). For n segments filter the total number of operations = 4·n+1 (the 1 for the constant a_o of the final result). However, additional time is required for the state variable manipulations as mentioned earlier (2.5.2).

2.6 R.F. SIGNAL SIMULATION

In the digital simulation of continuous RF signals, the problem is in having a modulating signal as a small ratio to the carrier. The carrier which contains no useful information will predominate the sampling rate at the expense of the useful modulating signal. One method[49] of simulating RF signal is by decomposing the RF signal to:

(a) Useful modulating signal at baseband: m(t).

(b) Carrier.

For RF signal: $x_m(t) = m(t) \cos \omega_c(t)$ (1)

Therefore the simulation will concentrate only on m(t), and the sampling frequency will be as ratio to the

carrier frequency.

For various types of modulations, the modulating signal is given by

$$\begin{aligned}
&\text{(i)} \quad \text{DSBSC} && m(t) = x(t) \\
&\text{(ii)} \quad \text{DSB-AM} && m(t) = 1 + a\,x(t) \\
&\text{(iii)} \quad \text{Phase modulation} && m(t) = 1 \exp jx(t) \\
&\text{(iv)} \quad \text{Frequency modulation} && m(t) = 1 \exp j\{\int x(t)\,dt\} \\
&\text{(v)} \quad \text{SSB} && m(t) = x(t) + j\hat{x}(t)
\end{aligned} \qquad (2)$$

where $\hat{x}$ is the Hilbert transform of $x(t)$.

Equations (2) can be rewritten as 2 conjugate parts:

$$x_m(t) = \frac{1}{2}\{m(t) \exp j\omega_c t + m^*(t) \exp(-j\omega_c t) \qquad (3)$$

Equation (3) is used when the negative and positive frequency regions, resulting from the presence of small angle modulation, is to be included in the simulation. If this angle modulation is discarded, then only amplitude is modulated and the resulting spectrum is symmetrical about the centre and equations (2) are used.

In some frequency domain simulations , the spectrum of RF signals are deduced from the series expansion:

$$S(t) = \frac{a_o}{2} \cos \omega_c t + \sum_{k=0}^{N} \{a_k \cos(\omega_c + k\omega_o)t + b_k \sin(\omega_c + k\omega_o)t\} + \frac{c_o}{2} \cos \omega_c t$$

$$+ \sum_{k=0}^{N} \{c_k \cos(\omega_c - k\omega_o)t + d_k \sin(\omega_c - k\omega_o)t\}$$

where ω_c angular carrier frequency

a_k, b_k coefficients of the k^{th} upper sideband

c_k, d_k " " " " lower sideband

2.7 OFF-LINE WORKING

2.7.1 Principle of Operation

The idea behind the off-line operation is to run the simulation independently of time. Both frequency domain simulation and time domain simulation can be performed in this way, but with every module of the system simulated or made independent of time. The latter point especially applied to a real signal stored on a magnetic tape or disc and run in scale time.

2.7.2 Characteristics

(a) All computers of any size or type can be utilised, regardless to speed or language. The only limitation is the size of the computer memory.

(b) Both frequency and time domain simulation can be performed.

(c) Simulation strategy easily implemented.

(d) The real signal must be simulated unless specially treated as explained earlier.

2.8 ON-LINE WORKING

2.8.1 Principle of Operation

The analogue input signal is converted to a digital signal by the A/D converter, Fig 2.8.1, ready for processing by the time domain simulator. The resulting digital signal is converted back to analogue signal by the D/A converter, whereas the digital signal could be obtained directly from the simulator if so desired. However, the seemingly easy procedure has a number of limitations and problems which must be carefully treated for good simulation. These are summarised as follows:

(a) The time allowed for processing an incoming signal sample by the simulator is the sampling time T_s seconds where $T_s = 1/f_s$, f_s being the sampling period.
From the definition of sampling theorem[8], the sampling frequency must not be less than 2 x highest frequency component of the incoming analogue signal.

(b) The limitation introduced in (a) for the processing time means that only computers with fast computational capability will be suitable for this type of simulation, unless a different computer strategy is adopted.

(c) A computer system with high speed arithmetic unit, together with minicomputer or microprocessor

controlled peripherals will overcome the time limitation economically, as will be discussed in a later chapter.

(d) Scaling already mentioned earlier, is essential and on-line simulation is only possible at base-band.

(e) The accuracy in the analysis depends, amongst other things, on the word length (number of bits) of the incoming signal samples produced by the A/D converter. Therefore there is inherent error due to the conversion action, and an acceptable error is allowed and taken into consideration in the simulation process.

2.8.2 Characteristics

(a) Output signal may be seen on an oscilloscope directly during the running mode.

(b) Other real-time systems can be used in conjunction with the original system, eg

(c) Off-line working is possible.

2.9 LIMITATIONS AND OBJECTIVES

Having presented the general layout and various aspects of communication system simulation, it will be beneficial to list the limitations in the subject in order to present the objectives in the design of a signal processing system simulator.

2.9.1 Limitations

These can be divided into 2 groups as follows:

(a) Simulation facilities:

(i) Real-time operations are limited to a few simulators (*) in environments having fairly large and expensive computers.

(ii) Stochastic simulations are not readily available.

(iii) Each simulator has its own simulation language.

(iv) The construction stage of the simulation involves procedures which must be made familiar to the user before a simulation is attempted.

(v) A pre-constructed system cannot readily be loaded into the simulator.

(vi) Running mode facilities are limited to the execution of a predetermined set of measurements. No change of control parameters of the simulated system are possible.

---

* Refs 14,15,47

(vii) No editing is possible, ie adding or deleting modules to or from an already existing system, as well as adding or deleting display routines (RMS, Mean value etc) during the running mode.

(viii) Running large simulations or multiple-run applications are limited by computer power.

(x) Portability of simulators is limited, ie it is difficult to use a simulator on more than one computer system.

(b) Computer power or system available: this limits the type of simulation problems which could economically be solved, such as:

(i) Real-time or stochastic simulation. Fast processing is necessary, hence a hybrid computer or multiprocessing computer system is needed.

(ii) Most communication simulations can be performed off-line in relatively slow machines, hence digital computer (serial processing) can be used.

2.9.2 Objectives

When designing a new communication system simulator, the following objectives must be met:

(a) Utilise all the points mentioned in (2.2) and (2.3) earlier.

(b) Transform the simulator into another research bench

tool, in which the user through the computer console can perform the following:

(i) During the construction mode:

* Loading the system block diagram interactively into the simulator, with the simulator taking the load and the user responding. The target is to have a peripheral unit on which the user draws the system block diagram directly, hence reducing the simulation language problem.
* The user should be able to add or delete modules to the already loaded block diagram without starting the simulation again. This facility will make the simulator similar to the equivalent "hardware" circuit, where the hardware modules are added or removed readily.
* The user should be able to load a preconstructed block diagram stored on a disc or tape to an already constructed block diagram in the simulator, and by some minor editing the final block diagram is produced. This facility will save valuable computer time and eliminate construction errors.

(ii) During running mode:

* In the same style, ie making the simulation as near to the actual physical situation as possible, the user should be able to change parameters of modules while the system is in the RUN mode, ie during signal processing. The new parameter value could be supplied through,

eg a buffer, and the parameter modification is performed during the signal processing cycle. This has the advantage of testing the behaviour of the simulated system by varying its module control parameters without pre-set changes.

* The user should be able to vary the global control parameters, ie a parameter which is used by more than one module such as sampling frequency, while the signal is processed. Again the system behaviour is tested readily.
* Display routine (RMS, FFT, etc) calculations should be made during the running mode, ie concurrently with the signal processing, and the result displayed at any instant of time the user wishes. The combination of this point together with the other two points make possible the behaviour of a system which is completely under the user control as well as being near to physical reality.

(c) On-line real time/off-line working must be possible in a practical and economical way.

(d) Capable of handling stochastic simulation.

(e) Designed on a modular form, in which routines are added or removed, or processed by another processing unit (minicomputer chip) with minimum difficulty.

(f) The simulator package to be made portable.

(g) The simulation language is made as general and easy to understand as possible.

2.10 ICOSS

The objectives outlined in (2.9.2) for the design of a communication system simulator were used as a guide for the implementation of a new signal processing system simulator to be called ICOSS: "Interactive COmmunication System Simulator", whose principle of operation is based on having complex interrupts controlled by the sampling rate. Each interrupt represents a function within the simulator, eg signal processing, construction, etc. The main features of ICOSS are summarised as follows:

(a) It is time domain based simulator oriented mainly on real-time on-line simulation, but off-line simulations are possible. Systems with feedback links can be simulated. However, the real time on-line process can be scaled (simulated) in order to operate ICOSS, off-line, for the same purpose. It is therefore possible to test ICOSS operation without the necessary equipments for actual real-time on-line operation.

(b) It makes use of all the points mentioned in sections (2.2) and (2.3).

(c) The objectives a-d, g of (2.9.2) are implemented and arrangements are made for the implementation of e and f.

(d) It can make use of microprocessor controlled modules in which some parts of the simulator, eg filter or display routines, or any other interrupt

of the simulator system, are brought outside the main computer and operated on in parallel, reducing execution time.

In Chapter 3, the complete design will be fully explained.

2.11 SUMMARY

In this chapter a survey was made in the use and design of signal processing system (SPS) simulations. The main theme was to present the various aspects of SPS simulation. Those aspects included: the listing and description of the requirements of both user-side and computer-side for the design of a new SPS simulator, the limitation of existing SPS simulations, and the main objectives which have to be met for the design of a new simulator. Those discussions were made in order also to state precisely what would be required for the design of a new time domain real (scaled) time oriented simulator, called ICOSS. The main features of ICOSS were listed, and in the following chapter 3 this new approach to SPS simulation and the complete description of it will be made.

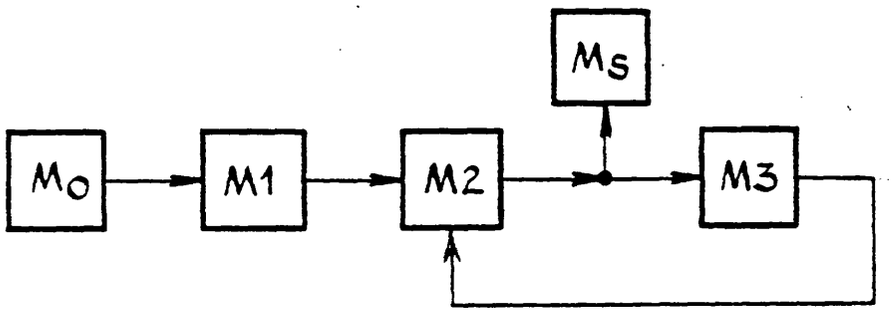
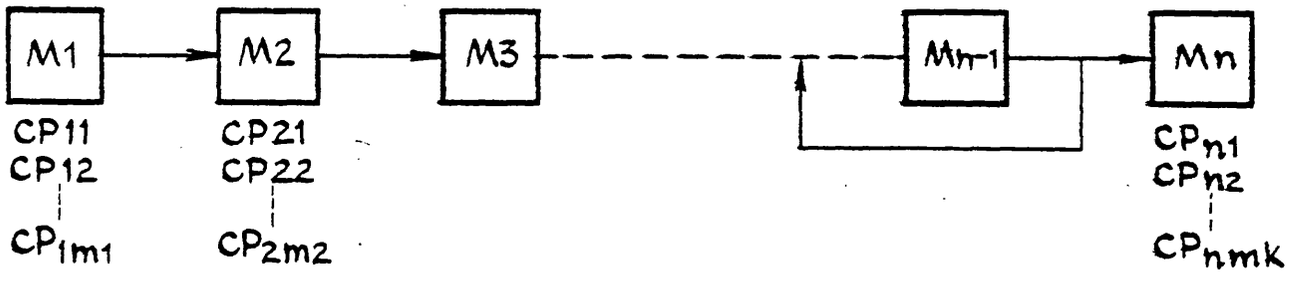


Fig. 211 Simple SP System



Where : M : Signal Processing Modules.
CP : Module Control Parameter.

Additional Factors :

* Measurement Parameter (or Graphical Output).
* Global Control Parameter.

Fig. 221 Typical Communication System

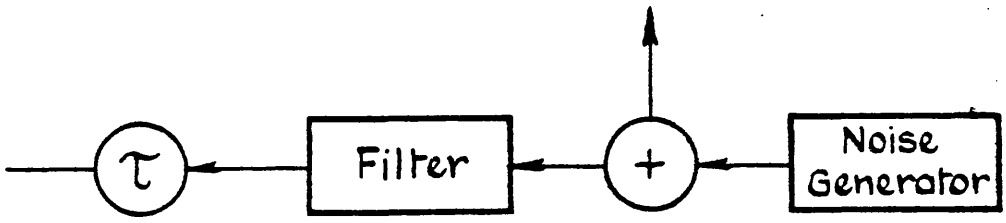


Fig. 222 Simple System for BLOBI Simulation

Left Node Field (Input)	Expression Field (Action)	Right Node Field (Output)	Tap Field Axialiary Output (or) Input to the Unit

Fig. 223 Systid Data Structure

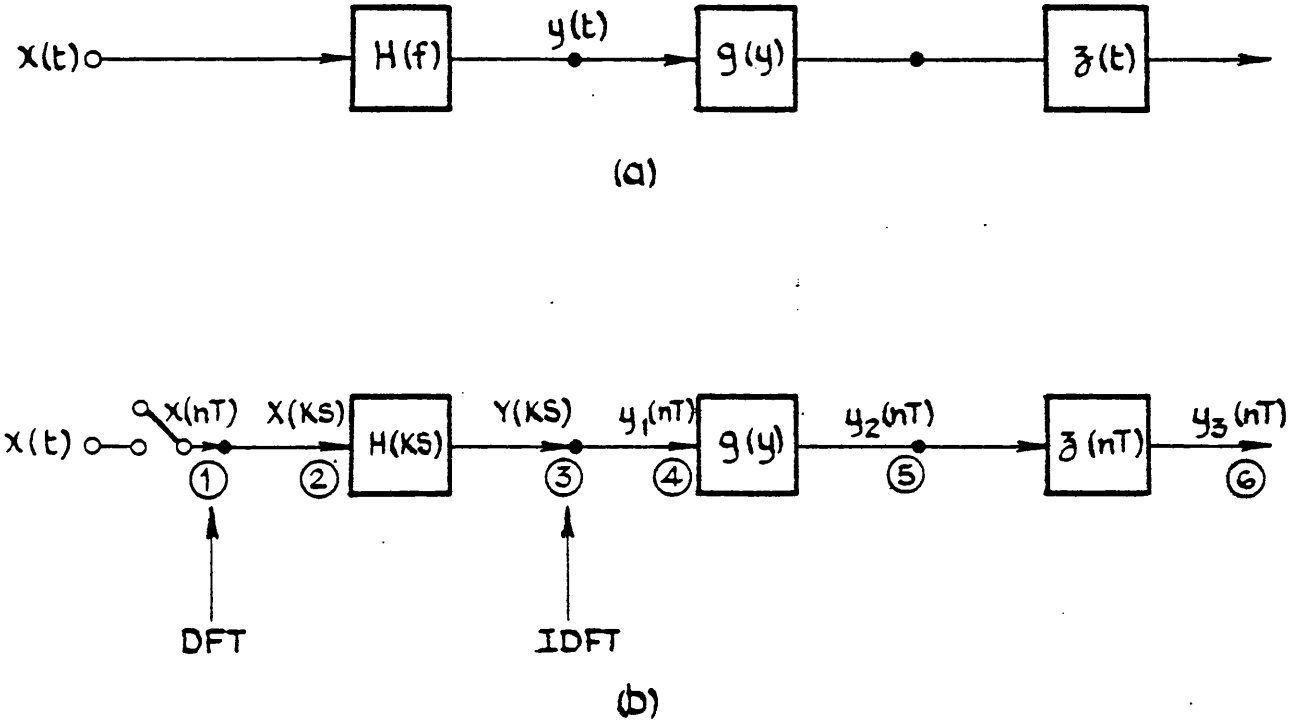


Fig. 244 Prototype System Containing Basic Elements of Communication System.

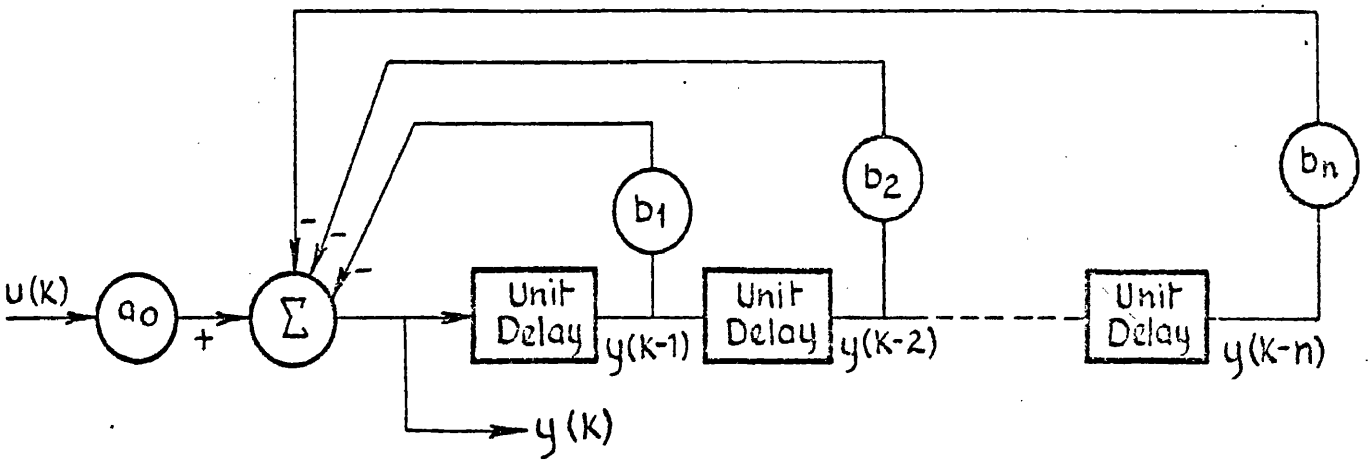


Fig. 251 Block Diagram Representation of a Discrete Time System

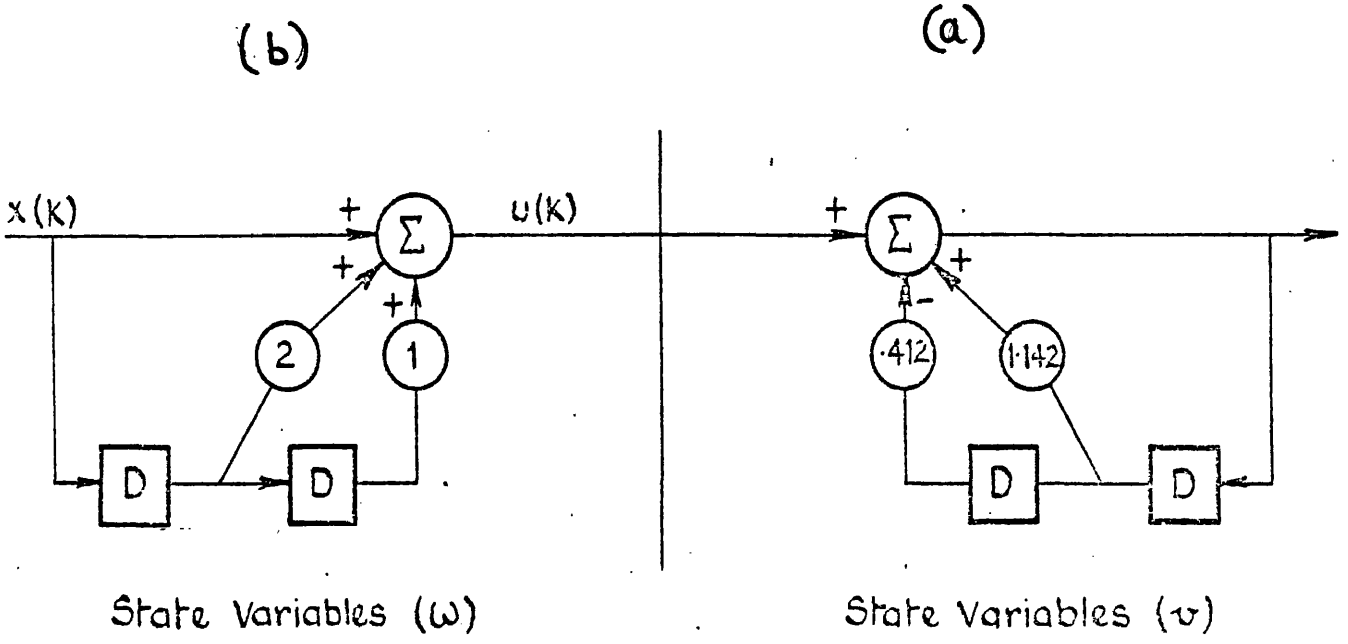


Fig 252 Block Diagram Representation of the Difference Equation of a Second Order Butterworth Low-Pass Digital Filter

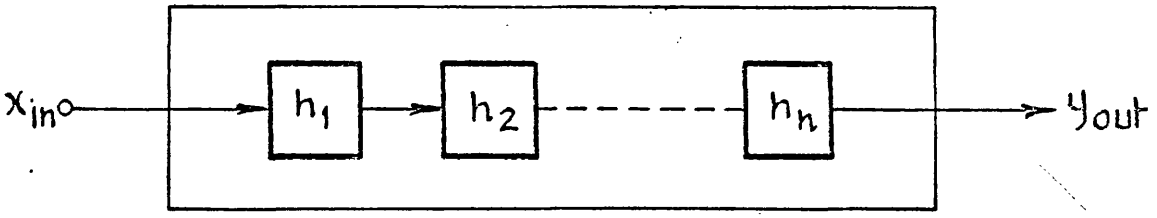


Fig. 253 A Digital Filter of n Second-order Segments

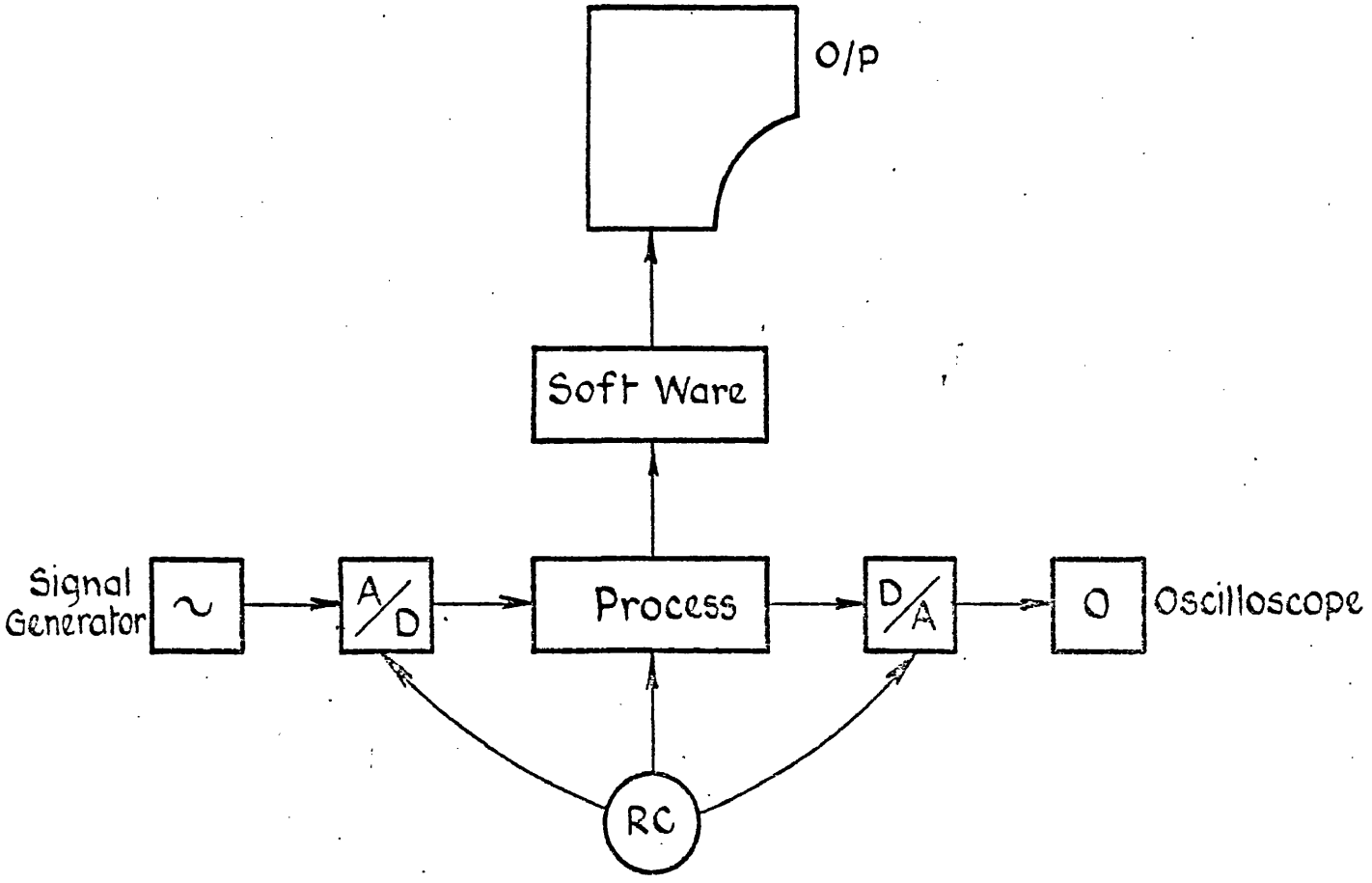


Fig. 281 On Line Working System.

CHAPTER 3

INTERACTIVE COMMUNICATION SYSTEM SIMULATOR ICOSS

3.1 INTRODUCTION

In the previous chapter, the general layout and various aspects of communication system simulation were given. Also the limitations in the subject of simulation, as well as the objectives in the design of a communication system simulator were given. These objectives and other relevant points already mentioned have been utilised in the design of the new simulator ICOSS, whose main features are summarised again as follows:

(*) Time domain real (or scaled) time on-line/ off-line operated system capable of handling stochastic simulation.

(*) Time independent mode of ICOSS: includes interactive block diagram construction, system editing (adding or deleting modules), etc.

(*) Time dependent mode of ICOSS: includes change of module parameters, output routine calculations etc.

(*) Software transportable where possible.

(*) Hardware microprocessor controlled units may replace part of ICOSS in order to improve the

time dependent operation, leading to faster execution time and better on-line operation.

ICOSS is basically a multiprocess software package, designed for signal processing in real (or scaled) time. The multi-processing operation is a combination of complex interrupts within the system, controlled by a real-time clock (sampling rate), as well as the priority of the interrupts. The ICOSS system does not only treat the signal processing problem interactively, it also provides the user, who requires limited or no computer programming knowledge, with the ability to access the system interactively and set up his communication system by means of a high level language on the terminal. The language chosen is Fortran IV for reasons of familiarity and usefulness in engineering and scientific applications, as well as its universality.

The ICOSS structure is made up of three groups of interrupts, Fig 3.1.1, namely:

(a) The Teletype group of interrupts: for the time independent actions such as block diagram constructions, editing etc.
(b) The Loop group of interrupts: for the time dependent actions in which signal processing takes place.
(c) The Flag group of interrupts: again for the time dependent actions in which output routine calculation (RMS, MEAN,FFT, etc) takes place.

The computer terminal allows the user to input lines of

data and use these lines for correction (editing) and changes. The system is made to generate enough messages to direct the user when constructing his system, and hence minimise errors. Once the system is set and becomes ready to run, a number of commands are made available in order to control the running of the system. New control parameters can be entered through a buffer on the terminal, and then during the running cycle the change takes place of the particular signal processing module's control parameter, without interfering with the actual running of the system itself, as shown in Fig 3.1.2. The user can then observe the effect of the changes at any node within the system (the system state can be accessed at any time). Adding a signal processing module or deleting one can also be done after changing to pausing mode, as another teletype interrupt. However, the output routines (RMS, MEAN, etc) are performed as priority interrupts within the running mode of the system, each routine representing an interrupt in its own and any result can be accessed at once (after becoming ready), by forcing a teletype interrupt command. The complexity of the problem and the diverse requirements that are expected from it, raise a number of points and problems in the design of ICOSS;

(a) Structured programming has to be adopted[50]
(b) The various parameters and links (pointers) have to be stored and accessed efficiently.
(c) The storage of signal values and their subsequent manipulation has to be performed independently of various interrupts.

(d) The system state variables have to be handled carefully for the overall system in order to maintain the correct signal relationships at various nodes, especially for feedback paths.

(e) The various signal processing modules have to execute at high speed.

The points raised in 2.2 and 2.3 as well as 2.9.2 of the last chapter were carefully utilised.

Development of ICOSS structure contained a library of modules sufficient for phase lock loop (PLL) type of problems. The reasons for this choice are the facts that it contains a feedback link, basic signal processing modules, and because the application problems which will be discussed in Chapter 5 , are based on phase lock loop.

3.2 PROGRAM STRUCTURE

The complexity in the design of ICOSS makes the adoption of structured programming[50] absolutely necessary, avoiding any "nesting" in the program. The software module within ICOSS has to be easily defineable and replaceable, ie ICOSS structure is based on a modular form system, each separate function of ICOSS is represented by a distinct software module. Each software module has only one entry point and one exit point and control is transferred from one software module to the next without any ambiguity. Program debugging, addition or deletion

of a subroutine within ICOSS can be performed with noticeable ease; see appendix 6. However, adopting this technique requires careful distinction between the program branches, and accurate classification of subroutines, thus increasing the time spent in building up the system initially.

The basic structure of ICOSS consists of three groups of complex interrupts, as has been mentioned earlier. These interrupts are divided into two categories:

(a) Time dependent interrupts.
(b) Time independent interrupts.

This division, which is more clearly defined in Fig 3.2.1, is due to the type of operation each individual interrupt has to perform, and the way ICOSS operates in general. The principle of ICOSS operation is as follows:

(i) With a (RUN) command from the teletype, the time dependent operations will be initiated by triggering the "real" time clock. During the clocking period T_{CL},

where $T_{CL} = \frac{1}{f_{CL}}$ and f_{CL} > sampling frequency, (Fig 3.2.2),

time dependent interrupts start to execute according to their order of priority, with the LOOP group of interrupts having the highest priority, and always in a ready situation as

soon as the system block diagram has been correctly constructed.

(ii) The time allocation within T_{CL} is so arranged that all the interrupts are catered for, (see Section 3.3).

(iii) A teletype command for a teletype group of interrupts will pause the system, and the time independent operation will override it.

The system software structure which is constructed in software blocks as mentioned earlier, are arranged so that the program execution is performed in a closed loop fashion, Fig 3.2.3, in which the controlling program (ICOSS1) rotates round a loop in the RUN mode of the system as follows:

LOOP → TTYFG → FLAG → LOOP etc

The highest priority group will interrupt this rotation, and the highest priority interrupt within the group is indicated by a vector. If time allows, the next level of priority interrupt is captured as soon as the first priority is serviced, and so on. The priority decisions are predetermined with regard to most interrupts, but some of the output routines (RMS, MEAN, etc) of the FLAG group of interrupts can be adjusted and made to give the facility to the user, ie the user labels the various flag interrupts himself.

The various operations mentioned above will be fully described during the course of discussion.

3.3 TIME ALLOCATIONS

The time allocation problem is mainly concerned with the time-dependent operations of ICOSS. Therefore, the time-independent operations of ICOSS may be allowed as much time as necessary for their operations, but remembering that when the time-independent interrupts are in action, they are in fact using valuable computer time, and must be minimised as much as possible.

Looking closely into the time allocation within ICOSS operations, ie during the sampling period T_s (Fig 3.3.1.), there are two regions within this period:

(*) A_L : which is devoted to the execution of the LOOP group of interrupts

(*) A_O : which is devoted to other time-dependent interrupts execution.

These two are related to the sampling period by:

$$T_S = T_L + T_O \qquad (1)$$

But T_S is related to the highest components of the input signal by the relations[8]

$$T_S \leq \frac{1}{2f_{sig}} \qquad (2)$$

where f_{sig} : highest frequency component of the input "analogue" signal to the analogue/digital converter - Fig 3.3.2

Therefore, T_S must be made as small as possible, for higher input signal frequency. But there are restrictions as will be shown below:

(a) The LOOP time T_L

The LOOP interrupt execution time T_L, will have a fixed period within a run, depending on the size of the simulated communication system at hand, and the time it takes to execute a signal sample within the LOOP. It is obvious that in the absence of other time-dependent interrupts, T_L will become equal to T_S, and this is the limiting time (minimum) within the system.

$$\text{Limiting time:} \quad T_L = T_S \tag{3}$$

(b) The other time-dependent interrupts time T_{OI}

The time needed for the execution of other interrupts, including interrupts manipulations, is variable and completely dependent on the problem at hand. The starting of this period (T_F) is triggered at the end of the LOOP interrupts execution time (T_L) and ended by the clock trigger.

Since those interrupts occur randomly, and may take any number at any one time, and each may take unknown length of time, then it is decided to spread it over a length of time, such as:

$$T_{OI} = \sum_{i=1}^{m} \sum_{1}^{n_i} T_F \qquad (4)$$

where m : number of interrupts waiting execution at one time.

n_i: number of time segments needed for one interrupt.

As mentioned above T_S has to be made as small as possible. But T_L is fixed for any particular problem. Therefore, the only variable is T_F and must be optimised.

To optimise T_F, the following constraints have to be considered:

(i) A_2 must be utilised as much as possible (Fig 3.3.3).

(ii) n_i must be made as small as possible, ie interrupts must be processed in as short a time as possible.

(iii) $T_S \rightarrow T_L$

Therefore a compromise solution must be reached. However, since this problem is unique and its solution depends entirely on the user, ie the user only can decide between obtaining quick result vs low sampling rate and vice-versa. Since T_S is variable and can be adjusted by the real-time clock - an external device for real-time operation, and which is under the user's command, then it is left for the user to decide on the sampling rate in order to achieve the best solution.

3.4 FLOW-CHART

The previous sections have indicated the pattern by which ICOSS would be implemented. Figure 3.4.1 shows the overall flow-chart of the system. The interaction between the groups of interrupts within the system is outlined in a simple form.

3.5 COMPLEMENTARY ITEMS

Before discussing the ICOSS operations in detail, there are a number of complementary items essential to these operations and it will make discussion simpler and clearer if they are explained first.

3.5.1 The signal processing modules input-output node arrangements

3.5.1.1 General notes

(*) Only those modules which have special mathematical transfer functions are included in the discussions, they do not include the output display routines, eg RMS, MEAN values etc not the auxiliary modules, eg branching.

(*) Signal processing modules may have unlimited numbers of inputs and outputs.

(*) The modules node numbering is performed automatically by the simulator, and presented to the user so that it can be referred to at a later stage.

(*) Any loose node must be short circuited.

(*) Signal value at each node within the system is stored temporarily (ie kept for one sampling period) and can be accessed by the user.

(*) In the simulated system construction mode, consecutive signal processing modules are automatically joined together with one signal link, unless otherwise specified.

3.5.1.2 Node numbering

A signal processing module (k) within a communication system, with multiple input nodes and multiple output nodes, Fig 3.5.1, has its nodes numbered as follows:

$$N_{ik} = N_{o(k-1)} + \Delta_{o(k-1)} - 1 \qquad \text{for input nodes}$$

$$N_{ok} = N_{ik} + \Delta_{ik} \qquad \text{for output nodes}$$

where Δ_{ik} is the number of (multiple) inputs to module k

Δ_{ok} is the number of (multiple) outputs from module k.

3.5.1.3 Signal value storage

The signal at each individual node is stored temporarily (one pass - within a sampling period), in a special stack (X), Fig 3.5.2. This double array stack stores the signal value together with the node condition, at any particular node. The condition is used for the system state purposes as will be described later (3.5.4). It is clear that the signal at any node within the system can easily be accessed at any time by simply indicating the node number. Therefore, during the RUN mode the signals are accessed using the node numbers as pointers, and modified as they are processed during the sampling period.

3.5.1.4 Example

To demonstrate the above terminology, consider the hypothetical communication system shown in Fig 3.5.3. The simulator automatically assigns node numbers in sequence, with the aid of Δ_i and Δ_o of each module, which are parameters characteristic of the modules themselves.

3.5.1.5 Implementation

(*) Δ_i and Δ_o are fixed for the particular signal processing module and permanently stored in (SPMNOD); they will be used only during the construction stage, when determining N_{ik} and N_{ok}.

(*) Once the inputs/outputs of a signal processing

module within a communication system are determined, the pointers (N_{ik} and N_{ok}) are stored in their position in a special stack to be called the system matrix (SM) as will be explained later (3.5.3).

3.5.2 The control parameters CPs and GCPs

3.5.2.1 Definitions, characteristics and general points

(a) There are two types of control parameters:

(i) The global control parameters GCPs: These are the parameters which are shared by more than one signal processing module, and their values are substituted automatically when they are needed. Eg sampling frequency f_s, as used by oscillators, filters etc.

(ii) Local control parameters LCPs: These are exclusive to their signal processing modules. They have to be supplied by the user when constructing the system. It is clear, that changing a LCP will have an effect only on the signal processing module concerned.

Example: A simple phase lock loop

(*) Block Diagram: as shown in Fig 3.5.4

(*) Signal processing modules and their control parameters

	Module	Local Control Parameters	GCP
1	SIGN	F1, A1, B1, G1	FS
2	PHSD	AA1	
3	FILT	FC	FS
4	GAIN	AA2	
5	VCO	FO, FINC	
6	SIGN	F2, A2, B2, G2	FS

(*) The above notations will be elaborated in later discussions.

(b) Most control parameters are used as supplied by the user, but some have to be modified in "secondary" control parameters. Therefore, the classes of LCPs are:

(i) Primary control parameters
(ii) Secondary control parameters

(c) Cases will arise where some secondary control parameters will be generated internally and will be used by the simulator only, as will be indicated later.

(d) The simulation will be using the secondary control parameters, whereas the user will specify the more familiar value of control parameter, ie the primary control parameter. Eg low pass filter has a primary control parameter f_c (cut off frequency), but this is converted to the secondary control parameters ($A = \frac{1}{1+c_o t(\pi f_c/f_s)}$) (see later). To retrieve the

primary control parameters, the backward conversion: $f_c = (\frac{f_s}{\pi}) \tan^{-1} (\frac{A}{1-A})$ is made, where f_s is the sampling frequency. It should be possible therefore for the simulator and the user to access both parameters without complication or difficulty.

(e) One of the main features of ICOSS, is the ability to interrupt the running action, and change LCPs and GCPs interactively. This will prove extremely useful in the running of the system and its behaviour when changing certain parameters.

3.5.2.2 Procedure for control parameters management

(*) User supply GCPs
(*) Simulator performs:
    (+) Store GCPs
    (+) Utilise GCPs in the appropriate modules
(*) Simulator specifies number of control parameters (CPs) of a particular module
(*) User supplies Control Parameters
(*) Simulator performs:
    (+) Check for secondary CPs and if necessary converts primary to secondary
    (+) Stores CPs (secondary)
    (+) Relates position of CPs in their stack, with the rest of the communication system structure for later signal processing
(*) Provision is made for accessing both primary and secondary CPs.

3.5.2.3 Storage arrangements

(a) Fixed data: Two types of fixed data are permanently stored; these are:

(i) The number of local control parameters related to the particular signal processor module

(ii) The signal processor module number, whose control parameters require conversion to secondary parameters.

These arrangements have to be decided upon carefully when building up ICOSS library of modules, making sure that the parameters are enough for efficient and flexible operation as well as removing all unnecessary calculations at the execution stage. By employing the conversion technique, the execution time of the loop is minimised which in turn gives more time for the other interrupts of the system.

(b) Variable data: These have been divided into two areas:

(i) The local control parameters actually used by the signal processing modules, ie for the modules where conversion to secondary parameters are required, only the secondary parameters are stored.

(ii) The global control parameters.

The separation was necessary in order to make the process in changing either parameter, if so required, simple to achieve.

3.5.3 The System Matrix SM

The simulated communication system structure as defined by the user, cannot be used directly by the simulator and therefore it must be transformed to a "working structure". This working structure is a 1:1 transformation, and the first step towards the actual signal processing procedure. The process of signal processing is further controlled by the system state constraints as will be explained later (3.5.4). A matrix, to be called system matrix SM, is used to represent the working structure of simulated communication system. It is a 7xN matrix, where N is the number of modules in the simulated communication system, Fig (3.5.5). Each row of the matrix represents a module (R_i, i = 1,..N). The numbering (1→N) is as specified by the user. The content of each cell in the columns of the matrix (C_k, k = 1,..7) is in fact a pointer devoted to a special job within the module,(Fig 3.6.3). Therefore these sets of pointers in each row describe completely the function of the particular module. Choosing pointer rather than storing actual values of module parameters, is due to the fact that the simulator has to deal with a communication system which may be varied in number of modules, or in control parameters of modules, while the system still running, as mentioned earlier. Also the complication of

the inter-relationship of various stacks, tables and links involved within the system makes it impossible not to induce errors during signal processing. A typical example which shows the complication involved is shown in Fig 3.6.3.

To summarise the various actions of the contents of each cell in the system matrix, the table shown in Fig 3.5.6 shows the three groups of modules which are used for the present work, namely: the signal processing modules, the output display routines and the auxiliary modules. Any function needed for the communication system must fall under one of these groups.

3.5.4 Loop directives

3.5.4.1 General notes

The building up of the system matrix SM for a communication system was sequenced according to the block diagram as defined by the user. But when processing a signal in a system which contains feedback links, the later structuring of SM will induce errors in the process if used without modifications, due to the possibility of not having the correct signal relationship in the multiple input modules. In order to have the right relationships of input signals into the multiple input modules, the SM must be restructured in such a way that the sequence of signal processing within the modules does not induce errors and must follow the system state constraints. Therefore there are two cases to consider:

(a) The maintenance of the original SM in order to present to the user the original structure which can be edited or varied as before in its familiar form.

(b) The internal restructuring of the SM in order to comply with the system state constraints.

These two conflicting requirements for SM are overcome by the introduction of the so-called "loop directives". But before discussing the mechanism of loop directives a brief look at the system state is necessary.

3.5.4.2 System state

Consider the simple system with a single feedback link shown in Fig 3.5.7. If it is assumed that all delays within the system are confined in the signal processing modules themselves, ie P_1 and P_2, then the time relationships will be as shown in the Figure:

$$y(k) = u(k) + x(k) \quad \text{for } k = 1,2,\ldots\infty$$

Therefore during the signal processing there are two types of modules to consider:

(a) single input type modules, which have to be processed first, but with condition that states modules having higher priority.

(b) Multiple input type modules, in which all input signals

have to be ready, ie all with present state situation, before output is produced.

3.5.4.3 Loop directive - implementation

(a) Method:

(i) System matrix SM remains unchanged

(ii) An array to be called the loop directive array (LDR) is constructed by storing the sequence of module processing of the simulated communication system. This is deduced in the following way:

(*) Assign (1.0) for condition ready present state to the nodes (inputs or outputs) of all the state modules, such as filters, differentiators etc, or to any other node which the user recognises as independent in its function.

(*) Scan SM from top to bottom and test for the condition of signal values at the module inputs, thus if

$x(I_i,2) = 0.0$ means module is not ready for processing yet, proceed to next line of SM

$x(I_i,2) = 1.0$ check next input of module (if any), if all input node conditions equal one, then module is ready for inclusion

in the signal processing.

Proceed to perform:

$x(I_o,2) = 1.0$ where I_o is number of all outputs of the module (I).

Store the module number in LDR and increase by one.

Then proceed to the next line of SM .

(*) Repeat until all modules are included.

(*) Once LDR is constructed, the signal value conditions are returned to "not ready" state, except for the independent modules and user's special nodes as defined above.

(iii) Every time editing to the simulated communication system is made, then LDR is reconstructed as in (ii).

(b) Example:

Consider the hypothetical system shown in Fig 3.5.8a. It is self-evident that there are a number of ways in which the signal can be processed, leading to different results at the output (o/p). Applying the concept described in (a), LDR is constructed in the way shown in Fig 3.5.8b, to contain

I	1	2	3	4	5	6	7	8	9	10	11
LDR(I)	1	2	7	8	9	11	3	4	10	5	6

Notice that I = N, the number of modules in the system.

3.6 THE LOOP GROUP OF INTERRUPTS

The construction of the system matrix SM and loop directive LDR in the way explained in the previous section, is performed as one of the teletype interrupts as will be described later (3.7). But once it is ready then simulator becomes set for triggering by the "real time" clock (RTC) as mentioned earlier(3.2). With the loop group of interrupts having the highest priority, they are executed in full. Each line of SM is considered as another internal interrupt to be dealt with in a sequence defined by the loop directive LDR , as illustrated by the state diagram of Fig 3.6.1. Any samples needed for output display routines are stored meanwhile in a specially allocated storage area (SAMSTO). Once the exact number of samples are accumulated for a particular display routine, another (5x3) auxiliary matrix to be called the flag matrix (FLM) is constructed. Each row of this matrix (FLM) will store a number of parameters necessary for the operation of the flag interrupt devoted to the execution of the particular output display routine as shown in Fig 3.6.2. The content of these cells will be erased as soon as the output display routine is executed in the flag interrupt as will be explained later.

It can be seen from Fig 3.6.1 that both auxiliary modules and output display routine modules are treated as independent interrupts. The first treats the situations such as branching and switching etc, and the second treats the sampling access and storage of samples etc, for the

output display routines.

As an illustration to the substitution needed for the execution of one loop interrupt representing a signal processing module (P_1) of a system, is shown in Fig 3.6.3 with all the stacks, matrices necessary for the operation. In Appendix E , a brief introduction to the various modules, parameters, tables, and stacks connected with the loop group of interrupts as used in the present prototype of ICOSS.

3.7 THE TELETYPE GROUP OF INTERRUPTS TTYFG

The teletype group of interrupts are mainly concerned with:

(a) Inputting data specifically relating to the construction of the signal processing system, ie building up the data structure of the system (either block by block or as a complete list), as well as editing an existing data structure.
(b) Varying the system's control parameters (local or global).
(c) Controlling the running of the simulation and the process in general.
(d) Looking at the output at any stage, graphs prints etc.

In the following sub-sections, only those interrupts which are included in the TTYFG group of interrupts at the present work are discussed. However, additional TTYFG

interrupts can be made with great ease, see Appendix G.

3.7.1 Construction CONS

The construction mode of TTYFG group of interrupts puts the simulator at stand-still, and is actuated by the command CONS to the teletype command request. In this mode the system matrix SM representing the communication system block diagram, as specified in section (3.5), is constructed. The END command will terminate the construction of the system matrix SM, but will set the simulator to accept more data in connection with intial conditions and levelling. Once these additional data are supplied, the simulator starts to construct the loop directive array LDR for the system, and at the end of which this mode of operation, ie CONS interrupt, comes to an end, and ICOSS becomes ready for a new mode of operation. To demonstrate this mode of operation, Fig 3.7.1 shows a simple system together with its interactive dialogue. More examples will be shown during the course of discussion.

Subsequent internal actions

The main target is to construct both SM and LDR for the system. However the approach may differ depending on the three distinctive types of modules already mentioned in earlier section (3.5), namely, the signal processing modules, the output display routines, and the auxiliary modules. With the entry of each module, the simulator

consults the modules library for its classification, and the procedure that follows depends on this classification. The intensified flow diagram in Fig 3.7.2 shows the method of SM construction for the system.

3.7.2 Editing EDIT

General points:

(*) Editing is only possible for already established systems.

(*) Two types of editings are possible; they are

(i) Insertion of new modules

(ii) Deletion of an established module

(*) LDR is updated at the end of every editing session and just before ICOSS becomes ready for a new mode of operation.

Insertion:

In order to insert an element in an already established system, the system matrix SM has to be modified by inserting a line of pointers at the appropriate position, having the exact relationship with the existing pointers in the matrix, as well as modifying some of the values of pointers in the lines preceding and following the inserted line. The algorithm used to fulfil the above objective can be summarised as follows:

(i) Check ICOSS modules library for existence (of the type) of the new module in ICOSS, and its position in the library if it exists. An error message is printed on the terminal if element does not exist.

(ii) Consult control parameters library (LCPLIB) for the number of control parameters required by the new module.

(iii) Consult input/output node library (SPMNOD) for input/output node relationship of the new module.

(iv) Call subroutine (CPMAN) for the manipulation of the local control parameters, ie if they need conversion to secondary control parameters or generation of new parameters, as well as storing these parameters in the control parameters store (LCPSTO) and noting their position in it.

(v) Call subroutine (MODIF) for the re-adjustment of the node relationship of the system. Since the numbering is done in blocks, then the modification in multi-input/output modules is performed in blocks as well. The introduction of the new module will make necessary the modification of the preceding and following input/output node pointers of most modules of the system.

(vi) Construct the new line in SM in a similar way to CONS discussed earlier.

(vii) Make a shift in SM at the appropriate position, enough to insert the new constructed line.

(viii) Up date the loop directive array LDR

The dialogue for this type of operation is shown in Fig 3.7.3 for a typical example.

Deletion:

Deletion of an existing module is simpler to implement than the insertion described above, but not greatly different. The line representing the module which is to be deleted in the system matrix SM is erased and modifications to the pointers of the preceding and following lines are also made. The gap caused by the deletion of the line in the system matrix SM is then closed. The algorithm used is summarised as follows:

(i) Define block number to be deleted.

(ii) Call subroutine (MODIF) for updating and modification of input/output nodes of remaining modules.

(iii) Erase the line representing the deleted module in SM ie SM(I,n)= 0 for n = 1,2,...7

(iv) Call subroutine (SHIFT) for closing the gap caused by the deletion of the line.

(v) Update the loop directive array LDR.

A dialogue for this type of operation is shown in Fig 3.7.4 for a typical example.

3.7.3 Changing control parameters (Local) CHCP

During the running of the system, it may be found necessary to vary the control parameters of some of the modules in order to investigate their effect on the behaviour of the system. There are two stages to perform this facility.

Stage one: receiving the new values of the control parameters and their element number from "the buffer" zone.

Stage two: inserting their secondary control parameters into CP stack in the appropriate locations.

This section is only concerned with stage one; stage two will be discussed in the next section (3.8). The procedure of implementing this facility is summarised as follows:

(a) The user specifies the block number and the new control parameters values, and stores them in the buffer zone ready for execution.

(b) At the first opportunity the CHCP interrupt takes place (CHCP priority is next to loop's interrupt priority) and those values are transferred into a special storage area (CPISTO), ready for the next stage.

3.7.4 Changing global control parameters CHGC

In response to CHGC teletype command the simulator comes

to a standstill, and the user specifies the global parameter number and its new value. The subsequent internal actions are as follows:

Search SM for modules whose LCPs required conversion from primary to secondary parameters. This is done by checking with (SPCSCP) which contains module type numbers requiring conversion to secondary parameters. Every module in SM involved in conversion process is operated on its LCPs in the following way:

(i) Backward conversion to primary LCPs using the old GCP.

(ii) Forward conversion to secondary LCPs using the new GCP.

(iii) Update LCPs in the (LCPSTO) stack.

A typical dialogue for this interrupt is shown in Fig 3.7.5.

3.7.5 Running an already established system ENTR

The long procedure in constructing a simulated communication system,as described in section (3.7.1) earlier, can be avoided if the same system is to be run again. The constructed system or any other standard system can be stored outside the computer memory in a magnetic disc or tape, etc and then "entered" into ICOSS in bulk in response to the command of ENTR. Any modification to the system is then done utilising the EDIT facility of the teletype interrupts.

The data format is similar to that used in CONS interrupt mentioned earlier. Compare with the MACRO facility which is employed in BLODIB[21], in which one single command generates a complete module (s) of the system.

3.7.6 Stopping (pausing) a running system

The command STOP is a teletype interrupt flag which brings ICOSS system to a complete halt. The user will then have to choose the options:

(i) Access the final results of any output display routines, if they are ready.

(ii) Continue running the system.

(iii) Logout.

Once the results, or sample values of the various output display routines are accessed and printed out according to the user requirement, they will be erased from their storage area. In the case of logging out, a flag (ISTOP), is generated which signals the system to come to a complete stop at the main program. The continue state will only make the system wait for the clock for another loop interrupt.

A typical dialogue of this type of interrupt is shown in Fig 3.7.6.

3.7.7 Block diagram display CMST

The final structure of the simulated communication system, ie its equivalent block diagram, can be displayed eg printed on the teletype terminal, using a teletype interrupt CMST. If specifying some controlling parameters, a part or all of the system structure is displayed. Internally, the pointers in the system matrix SM are substituted by their absolute values, and printed out in their final form.

Some typical response for this type of interrupt is shown in Fig 3.7.7.

3.7.8 Graph-plotting management routine PLTM

Presenting a graph plot may take a number of forms depending on the problem and the user, as well as the graph plotting peripheral in the computer network. In response to PLTM command, a pause takes place in the simulation and the user under the direction of the simulator, gives directions as to the X axis and Y axis forms as well as the number of graphs, number of points/ graph etc, as shown in Fig 3.7.8.

3.7.9 Graph-plotting routine PLTG

In response to a teletype command PLTG, the accummulated signal values and the plot control parameters as specified by the user in PLTM interrupt, are fed into the

graph plotting peripheral. As soon as the curves are plotted, the signal samples are erased from the store giving way for new signal samples. This type of plotting arrangement, an off-line procedure, is most suitable for the present set-up. The on-line graph plotting, ie plotting samples as soon as they become available on a screen of a VDU, can be implemented readily. This point will be elaborated in Chapter 6.

3.7.10 Initialisation routine interrupt INXM

In communication system problems, the situation may arise in which it becomes necessary to initialise the system, especially when multiple transient responses are needed for different signal processing modules control parameters. This interrupt will set the system state variables, the temporary signal sample storage to zeroes.

3.8 THE FLAG GROUP OF INTERRUPTS

In this group of interrupts, the output display routines (calculations), as well as the insertion of the new secondary control parameters of signal processing modules in their appropriate storage locations, are performed in segments as mentioned earlier (3.2). The same principle employed with the other groups is applied to this group of interrupts, as shown in the state diagram, Fig 3.8.1. The flag interrupts are triggered by a global flag (IFLAG), generated either by the loop group of interrupts in which

case it indicates that the right number of points (signal samples) has been accummulated in (SAMSTO) ready for execution by the output display routine concerned, or by the TTYFG interrupt CHCP in which case it indicates that a change of parameters need to be made in one of the modules of the signal processing system. The value of IFLAG indicates the number of interrupts at one time. There are four situations to consider.

(i) One flag interrupt for one job.

(ii) More than one interrupt for one type of job.

(iii) A flag interrupt is engaged processing one job and another call for a flag interrupt for a job of lower priority takes place.

(iv) A flag interrupt is engaged processing one job and another call for a flag interrupt for a job of higher priority takes place.

The procedure for treating the above situations and flag group of interrupts in general, is summarised in Fig 3.8.2. Further, a brief description of a number of output display routines (as flag interruts) is found in Appendix C.

3.9 DEVELOPMENTS AND FURTHER WORK

The structure of the prototype version of ICOSS described in the previous sections, shows that there are a number of areas which can be further developed, the main ones are:

(i) The expansion (and reduction) of the present complement of modules.

(ii) Improvement in the speed of execution.

(iii) The utilisation of some of the peripheral equipments.

(iv) The coupling with other simulator systems.

The modular form (structured programming), makes the first area of development possible, so that:

(*) Addition of new modules to complete the communication system (signal processing modules) library, is easy to implement.

(*) Replacement of an old module by another more sophisticated one is possible.

(*) Addition of groups of interrupts is also possible without disturbing the structure of ICOSS.

The developments regarding the second area can be made in many ways. They include either having a giant digital computer or network of processors (mini computers) ie multiprocessing operation, or both. These ideas will be elaborated in the next Chapter (4).

As an example to the further development of ICOSS mentioned in the third area, is the block diagram display in which a more efficient, storage type video display with a buffer, is utilised. The data structure regarding the system block diagram is stored locally, and any editing modification or display to it will be done there.

Finally, as mentioned earlier, ICOSS is mainly concerned with communication systems simulation. Coupling this simulator with another simulator concerned with the analysis

of other topics in telecommunication problems will be a great asset to the telecommunication engineer working in research and development areas.

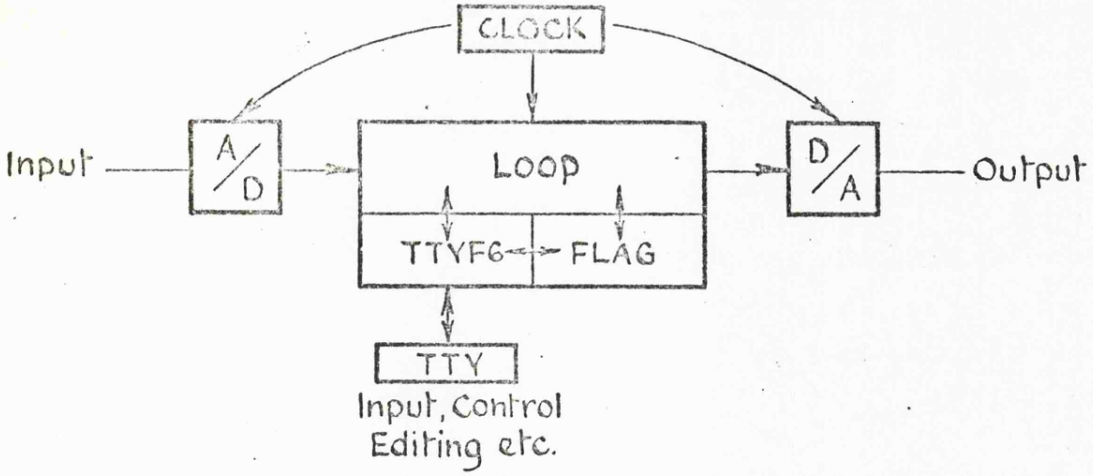
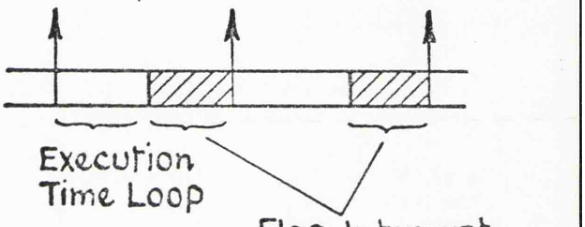
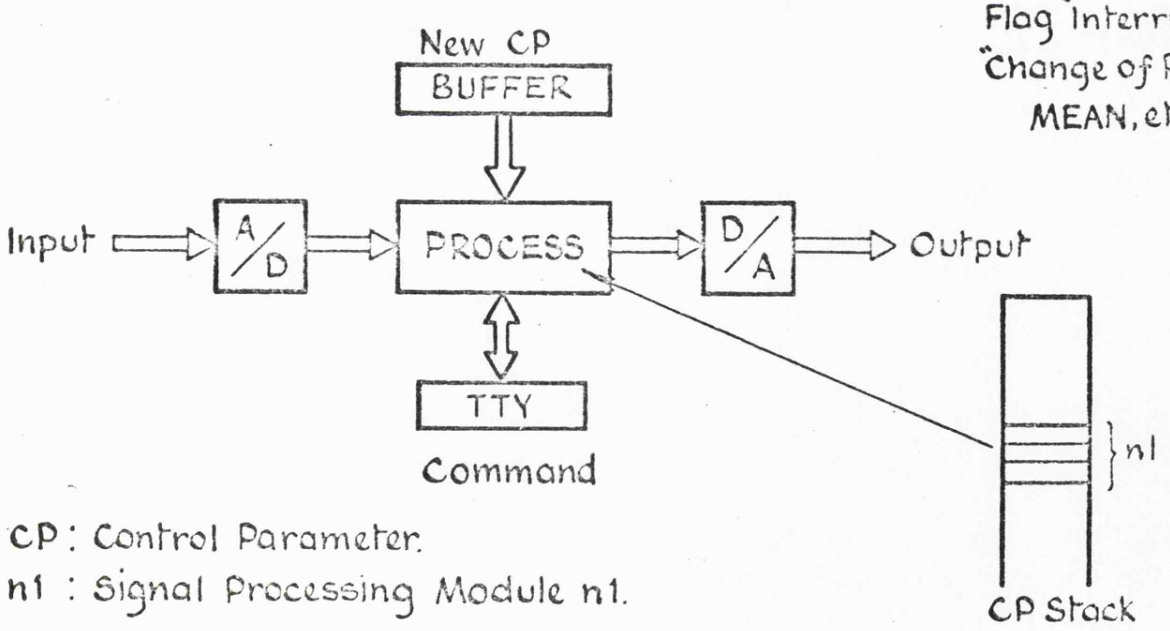


Fig 3 11. ICOSS OVERALL WORKING ARRANGEMENT





CP : Control Parameter.
n1 : Signal Processing Module n1.

Procedure

(i) Input New CP at Buffer.
(ii) Interrupt During Flag Interrupt Space if Priority Allows.
(iii) Replace Old CP by New in CP Stack.

Fig. 312 Change of LCP Arrangement

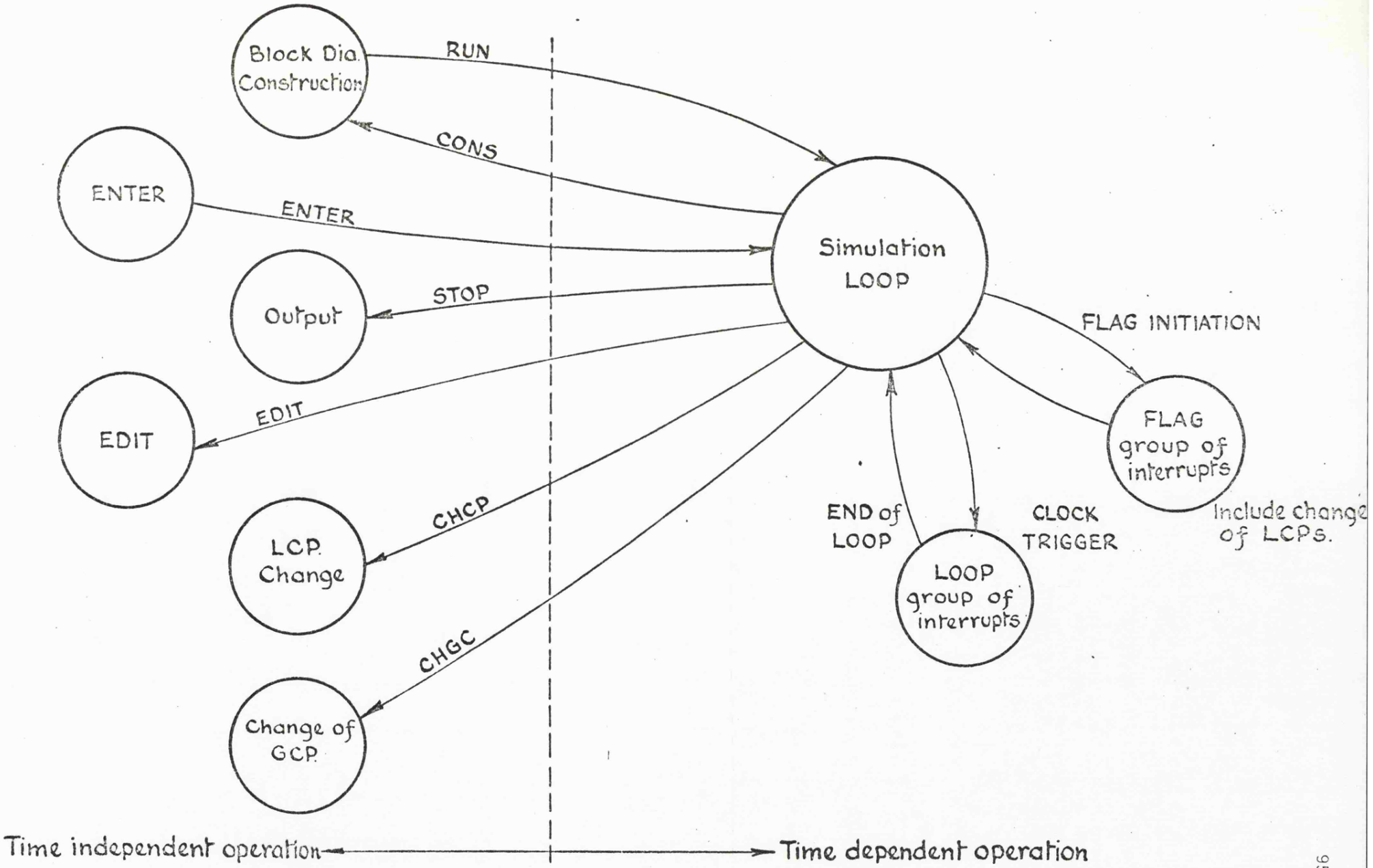


Fig. 321 ICOSS State Diagram

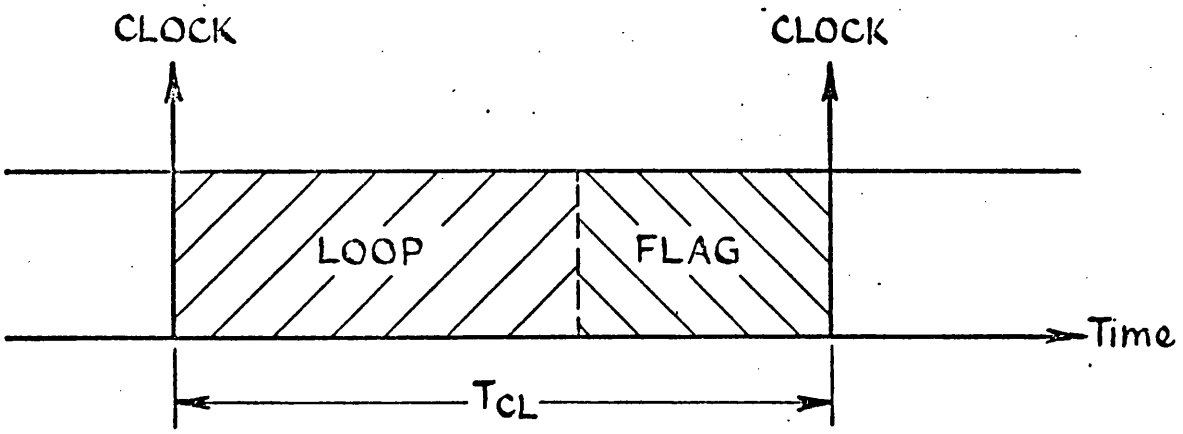


Fig. 322 Clocking Period Subdivision (Allocations)

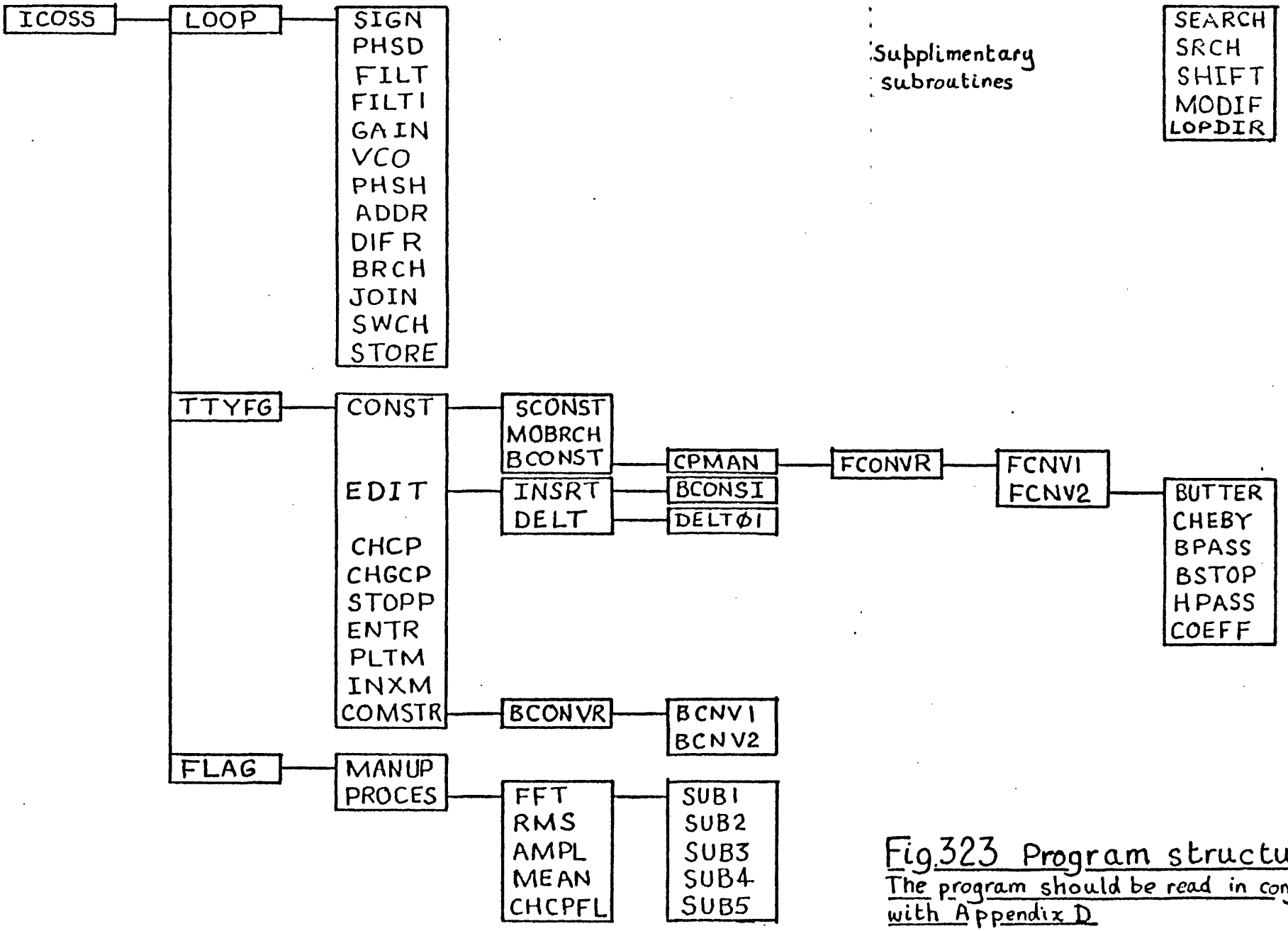


Fig.323 Program structure

The program should be read in conjuction with Appendix D

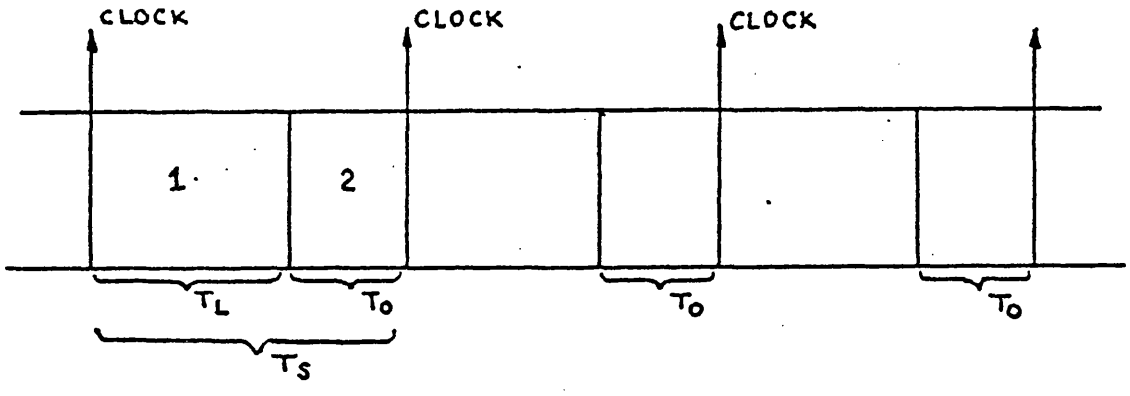


Fig 331 : Time allocations

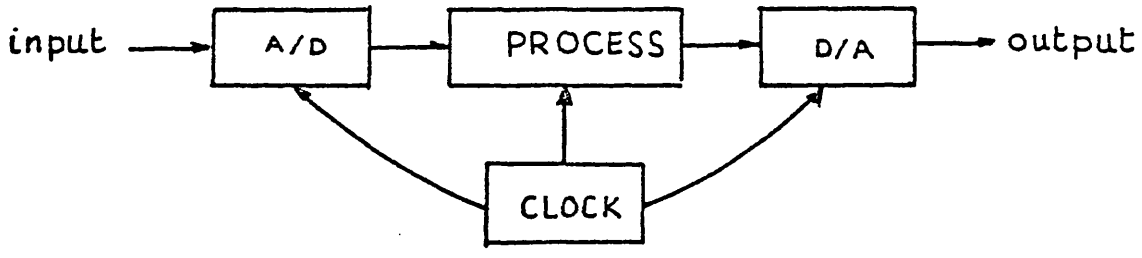


Fig 332 : System set-up

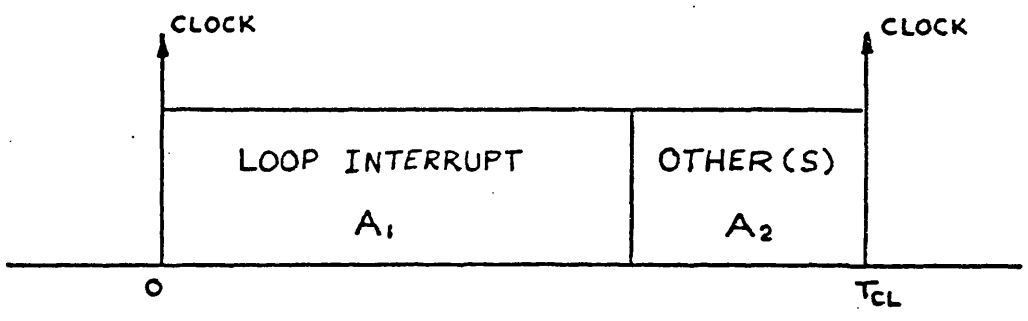


Fig.333 : Clocking period allocations

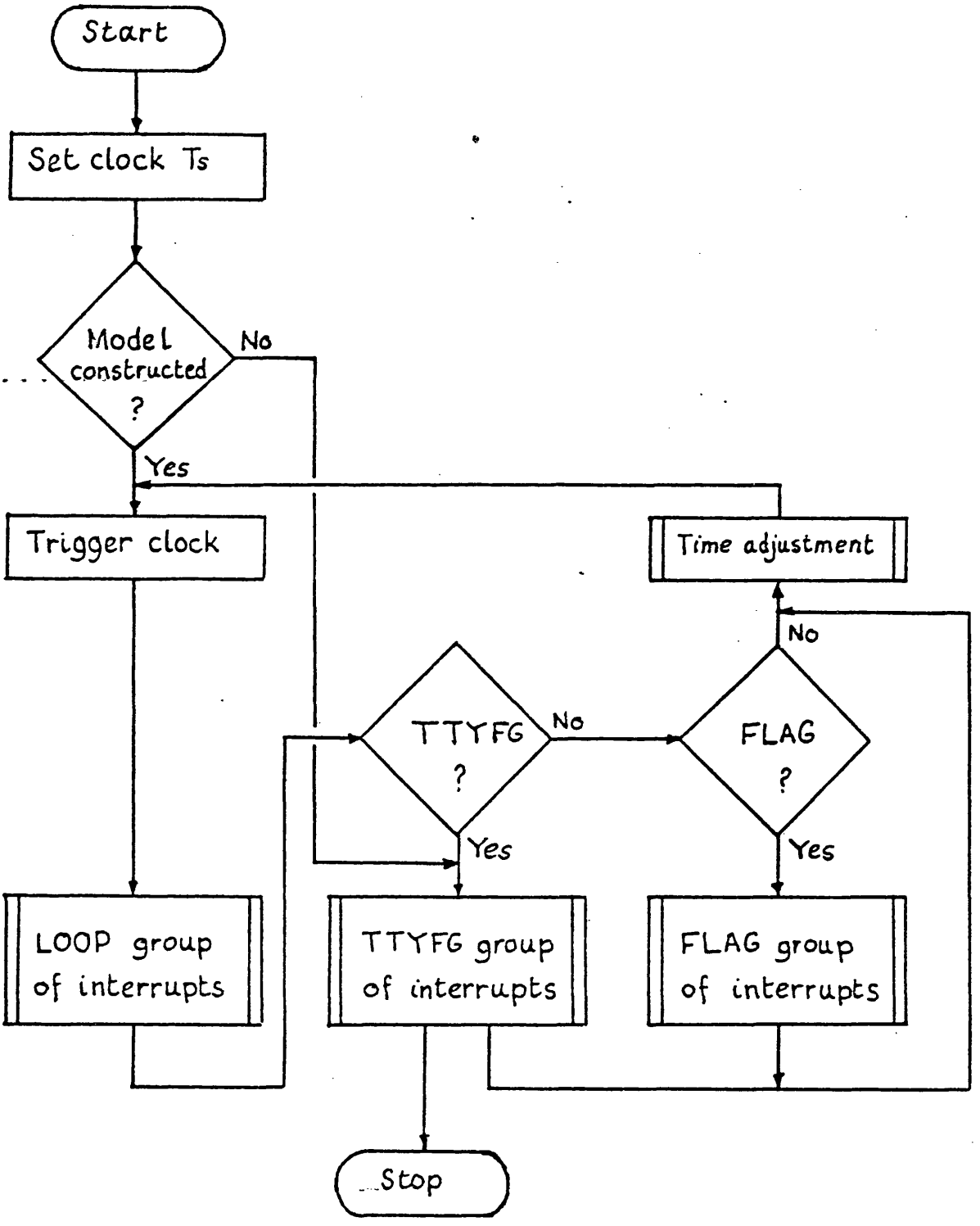


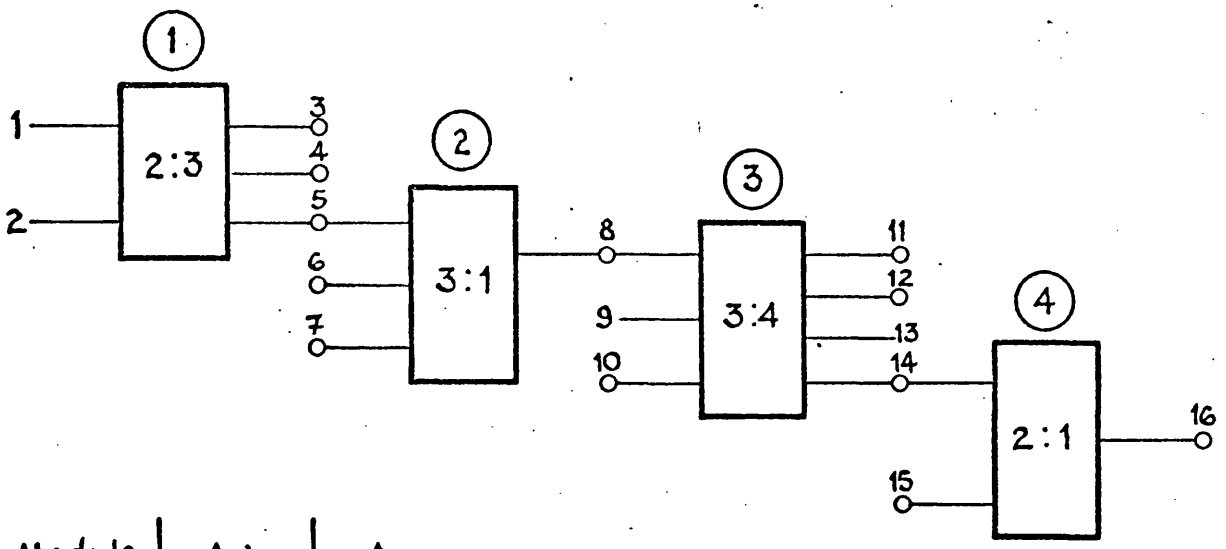
Fig.341 Basic System Flow-chart

Fig. 351

Node Number	Sig. Value	Condition
1	13·45	1 or Ø
2	etc.	
3		
4		
⋮		

1 Present State
Ø Previous State

Fig. 352 X Structure



Module	Δi	Δo
1	2	3
2	3	1
3	3	4
4	2	1

Fig. 353 Input/Output Node Relationship-Hypothetical System

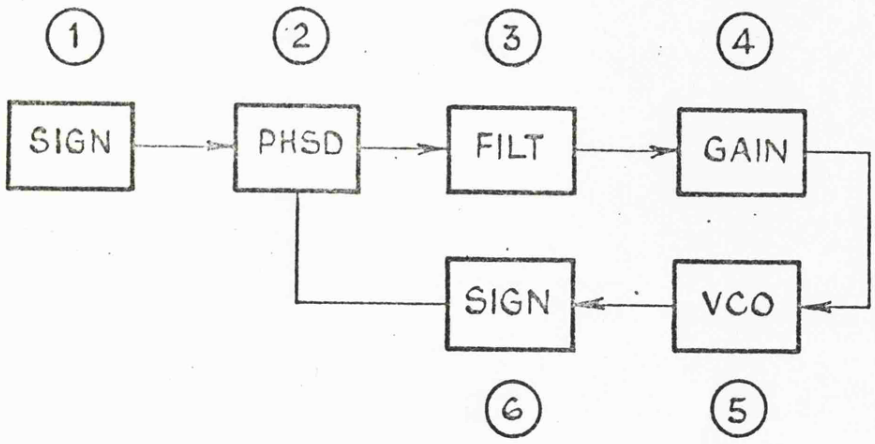


Fig. 354 Simple Phase Lock Loop PLL

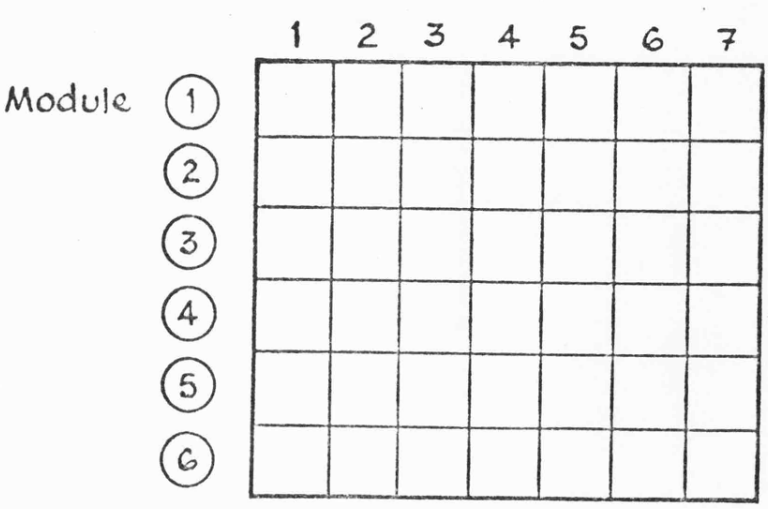


Fig. 355 SM Representation of PLL of Fig. 354

Column	1	2	3	4	5	6	7
Signal processing (SP) modules: SIGN, GAIN, FILT, etc	SP position in ICOSS library (SPM LIB)	LCP position in library (SPC LIB)	System state variables positions in (XM) store if any	GCP position in (GCP) store	Input nodes number indicator	Output nodes number indicator	Storage area identifier
Output display routines (ODR): FFT, RMS, MEAN etc	Display type number as it appears in (ODR LIB) library	Maximum number of points required for calculations of the output display routine	Latest reserved location in the storage area (STO SAM)	Priority of output display routine	Counter value	Output node number at which the signal is accessed	Storage area identifier
Auxiliary modules (AM): ARCH, etc	Position of AM in (AXM LIB) library	0	0	0	Input node number	Output node number	Storage area identifier

Fig 3.5.6 Description of the contents of the system matrix cells for the modules used in ICOSS

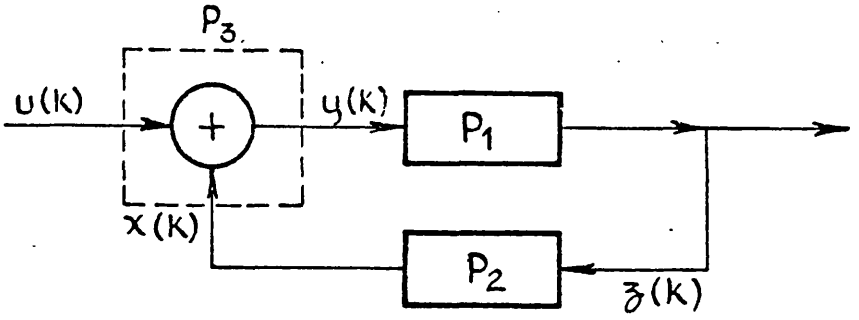
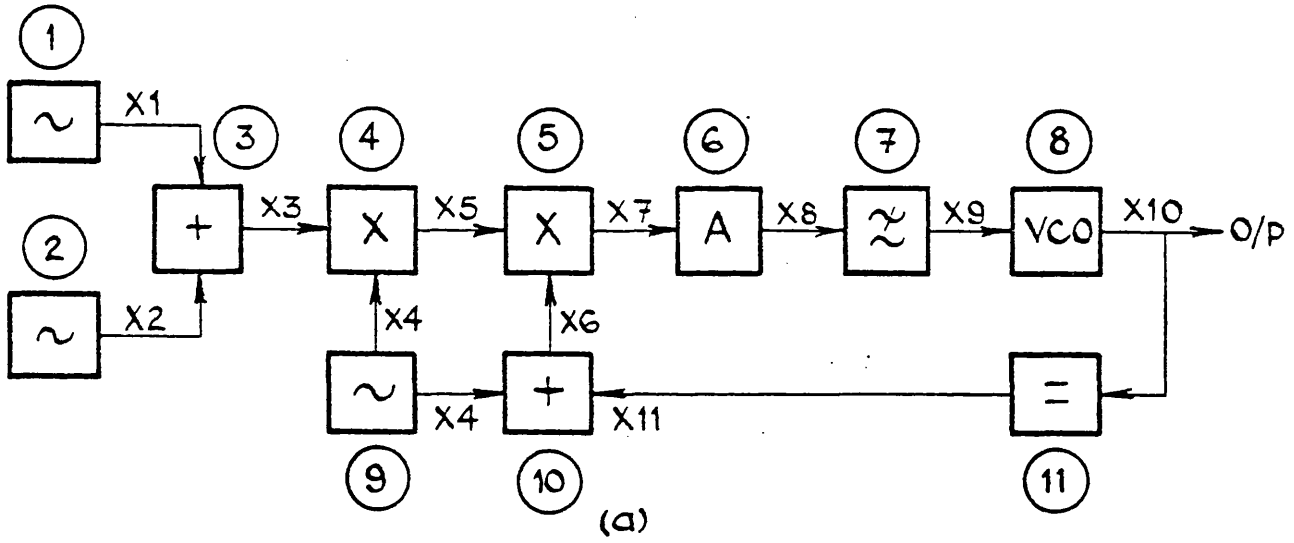


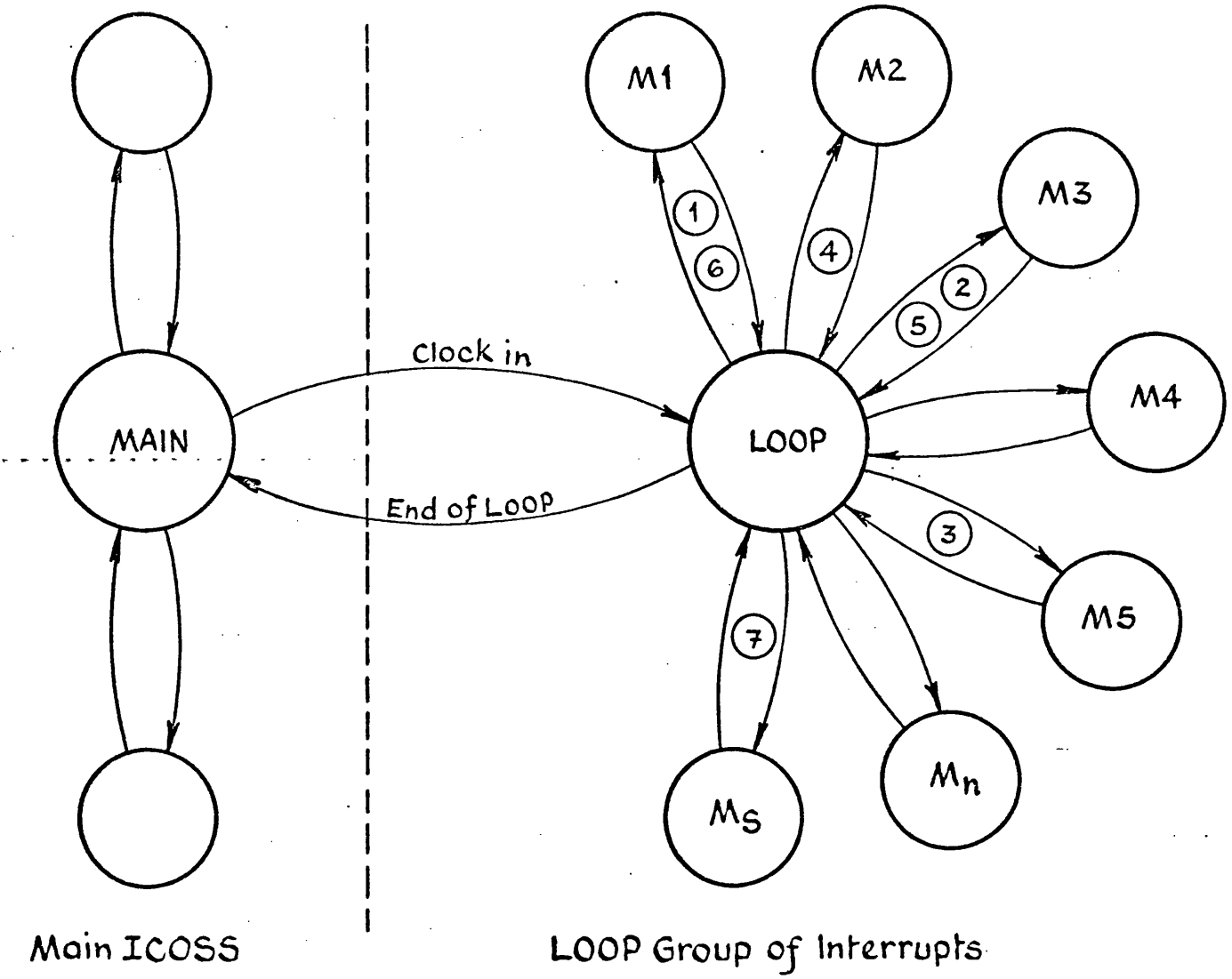
Fig. 357



			Pass 1	2	3
1	SIGN	X1	✓		
2	SIGN	X2	✓		
3	ADDR	X1+X2→X3		✓	
4	MULT	X3•X4→X5		✓	
5	MULT	X5•X6→X7			✓
6	GAIN	X7→X8			✓
7	FILT	X8→X9	✓		
8	VCO	X9→X10	✓		
9	SIGN	X4	✓		
10	ADDR	X4+X11→X6		✓	
11	BRCH	X11→X10	✓		

(b)

Fig. 358 [LDR] Construction



Sequence of Interrupts
M1
M3
M5
M2
M3
M1
MS
etc.
Main
etc.

Fig. 361 Loop State Diagram

n	1	2	3	4	5
FLM(I,n)	Type Priority	Pointer to Last Stop Storage Space	Transient	Number of Points	Count Ø

Fig. 3 62 Flag Matrix [FLM] (3x5) Matrix

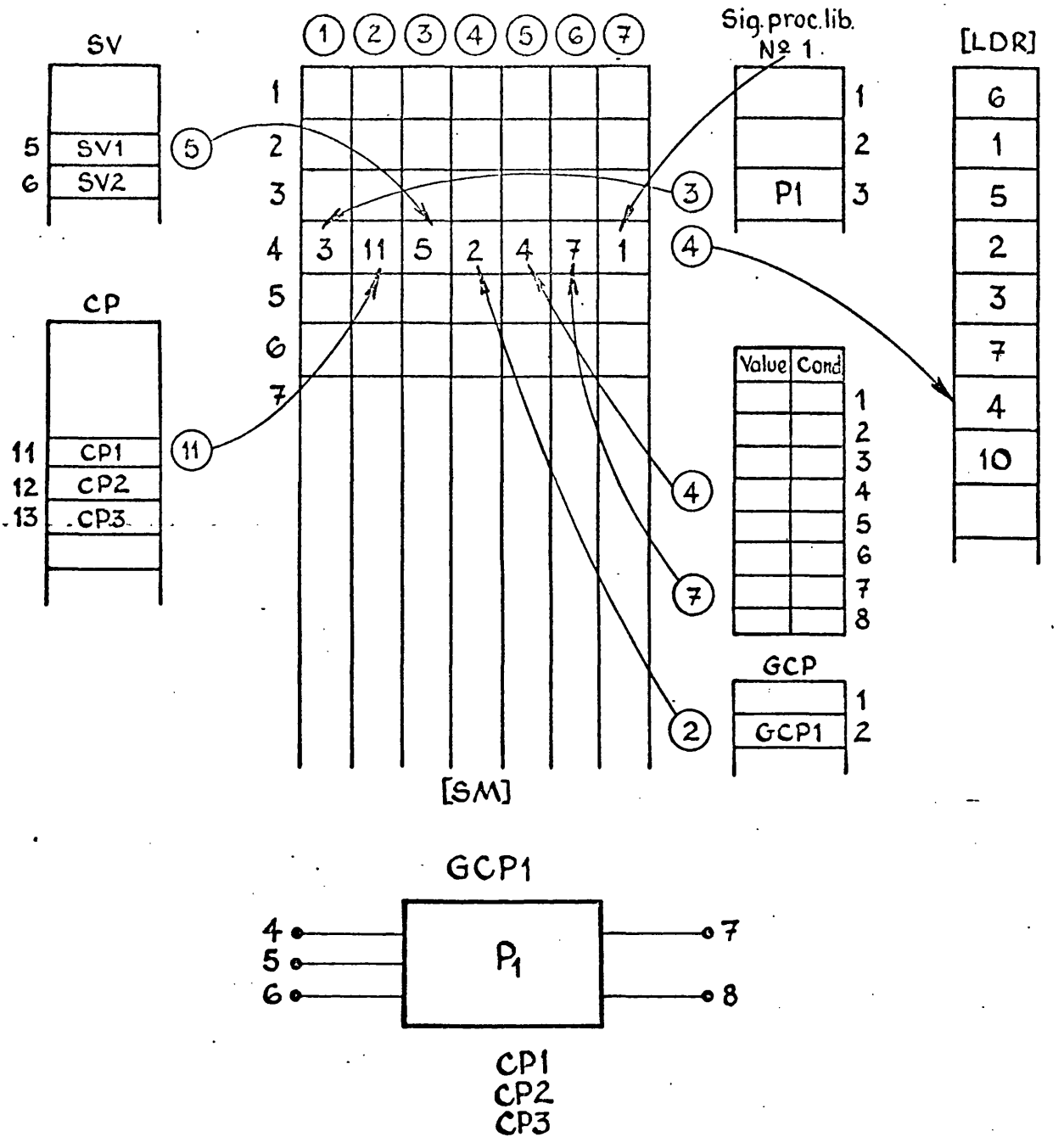


Fig. 363 SM Internal Relationshio & Module Representation

Figure 3.7.1a

```
O5? TTY COM? CONS
O5? SAMPLING FREQ? 100.0
O5? BLOCK 1: SIGN
O5? I/P = 1, O/P = 5, LCP = 4
O5? 15.0
O5? 1.0
O5? 0.0
O5? -1.0
O5? BLOCK 2: SCOP
O5? MEAN
O5? 1. NO OF PTS? 2. NODE NO? 3. PRIORITY? 10  5  1
O5? BLOCK 3: SCOP
O5? TRAN
O5? Y
O5? 1. NO OF PTS? 2. NODE NO? 15  5
O5? BLOCK 4? END
O5? 1. NODE NO? 2. EXT SIG? 3. IDNT?
O5? 1  0.0  1
O5? 2  0.0  1
O5? 3  0.0  0
O5? TTY COM? . . . .
```

Figure 3.7.1b CONS dialogue for the system (a) above

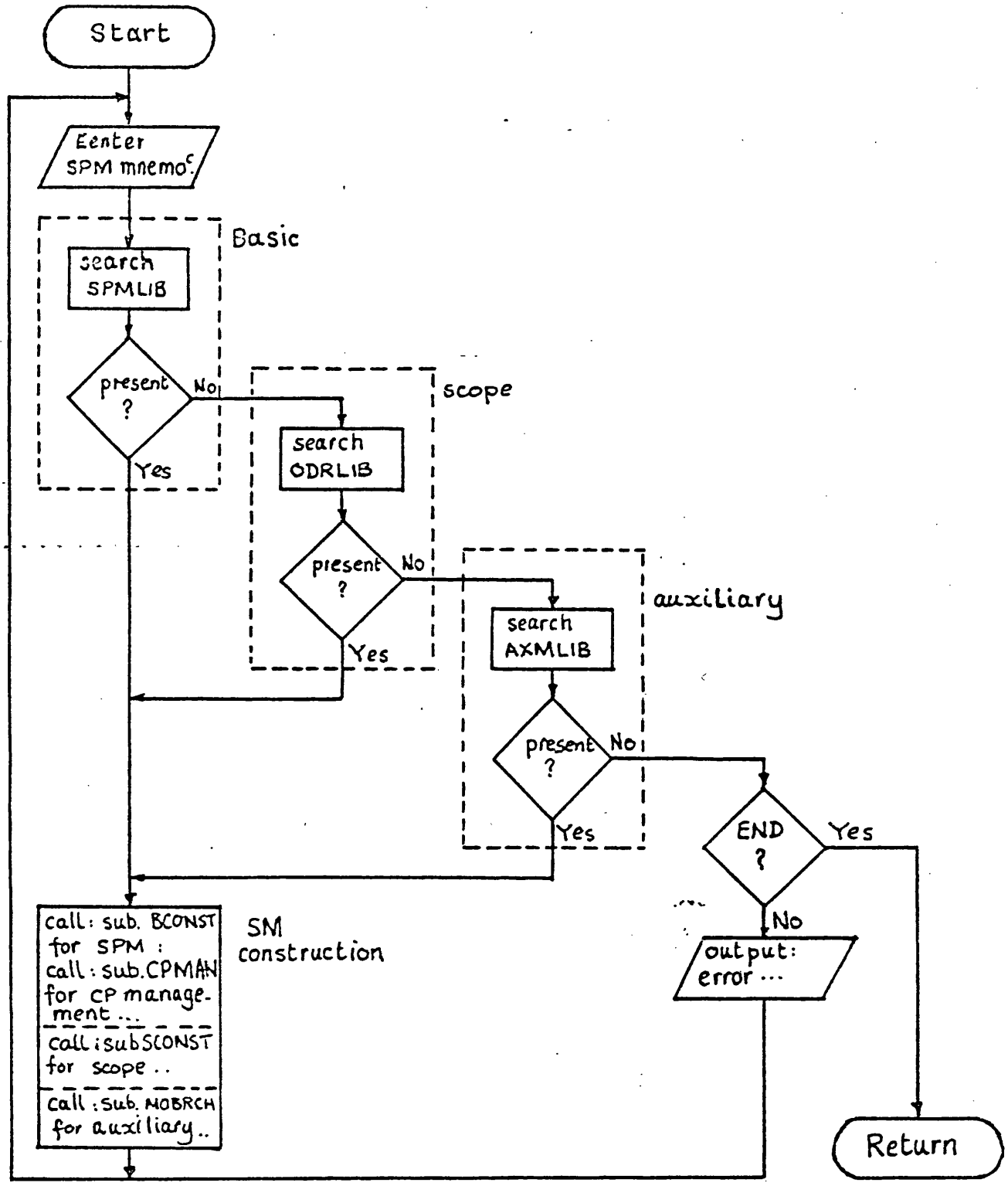


Fig. 372. Flow chart for SM construction

```
O5? TTY COM? EDIT
O5? TYPE OF EDIT? INSRT
O5? PREC BLOCK NO? 1
O5? TYPE OF NEW ELMNT? GAIN
O5? I/P = 4, O/P = 5, LCP = 1
O5? 3.5
O5? TYPE OF NEW ELMNT? END
O5? TTY COM? . . . .
```

Figure 3.7.3 Editing; inserting a new module to an existing block diagram

```
O5? TTY COM? EDIT
O5? TYPE OF EDIT? DELT
O5? BLOCK NO? 2
O5? BLOCK NO? O
O5? TTY COM? . . . .
```

Figure 3.7.4 Editing; deleting a module in an existing block diagram

```
O5? TTY COM? CHGC
O5? 1
O5? 150.0
O5? TTY COM? . . . .
```

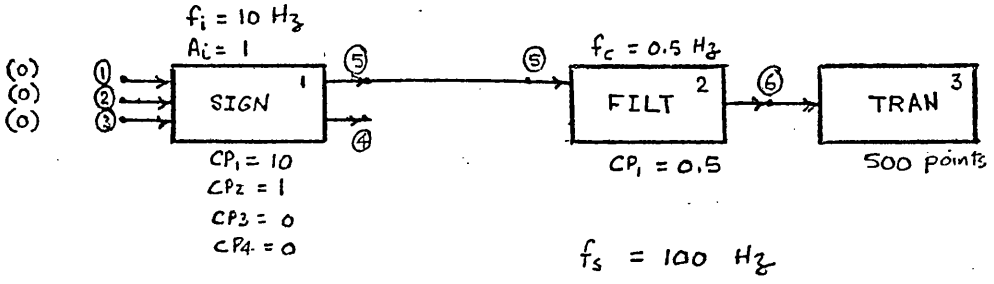
Figure 3.7.5 Dialogue for change of global control parameter CHGC

```
O5? TTY COM? STOP
O5? IS RESULT REQUIRED? YES
O5? TYPE OF SCOP? MEAN
O5? MEAN VALUE = .......
O5? CONT OR LOGT? L
O5? EDJ OO .....
```

If results not available yet, then:

```
O5? TTY COM? STOP
O5? CONT OR LOGT? C
O5? TTY COM? . . . .
```

Figure 3.7.6 Stopping (pausing) the running system; getting results and/or logging out.



In response to:

```
CMST
      1     1
```

The teletype will print:

```
BLOCK NUMBER:     1
SIGNAL PROCESSING MODULE: SIGN
INPUT NODE:
                  1
                  2
                  3
OUTPUT NODE:
                  4
                  5
LCP:
                 10.0
                  1.0
                  0.0
                  0.0
BLOCK NUMBER:     2
SIGNAL PROCESSING MODULE:  FILT
INPUT NODE:
                  5
OUTPUT NODE:
                  6
LCP:
                  0.5
BLOCK NUMBER:     3
SIGNAL PROCESSING MODULE:  TRAN
NUMBER OF POINTS = 500     NODE NUMBER = 6

GCP:
                 100.0
```

Fig 3.7.7 CMST command and response

(i) For indirect x axis

```
O5? TTY COM?  PLTM
O5? NO OF XAXIS POINTS: 100
O5? DIRECT OR INDIRECT: 0
O5? 1. STARTING PTS,2. STEP: 1.0 1.0
O5? 1. NO OF PTS/GRAPH, 2. NO OF GRAPHS: 100  5
O5? TTY COM? . . . .
```

(ii) For direct axis

```
O5? TTY COM?  PLTM
O5? NO OF XAXIS POINTS: 100
O5? DIRECT OR INDIRECT: 1
O5?  1.0
O5?  2.0
      -.-
      -.-
O5?  100.0
O5? 1. NO OF PTS/GRAPH, 2. NO OF GRAPHS: 100  5
O5? TTY COM?. . . . .
```

Figure 3.7.8:PLTM Dialogue

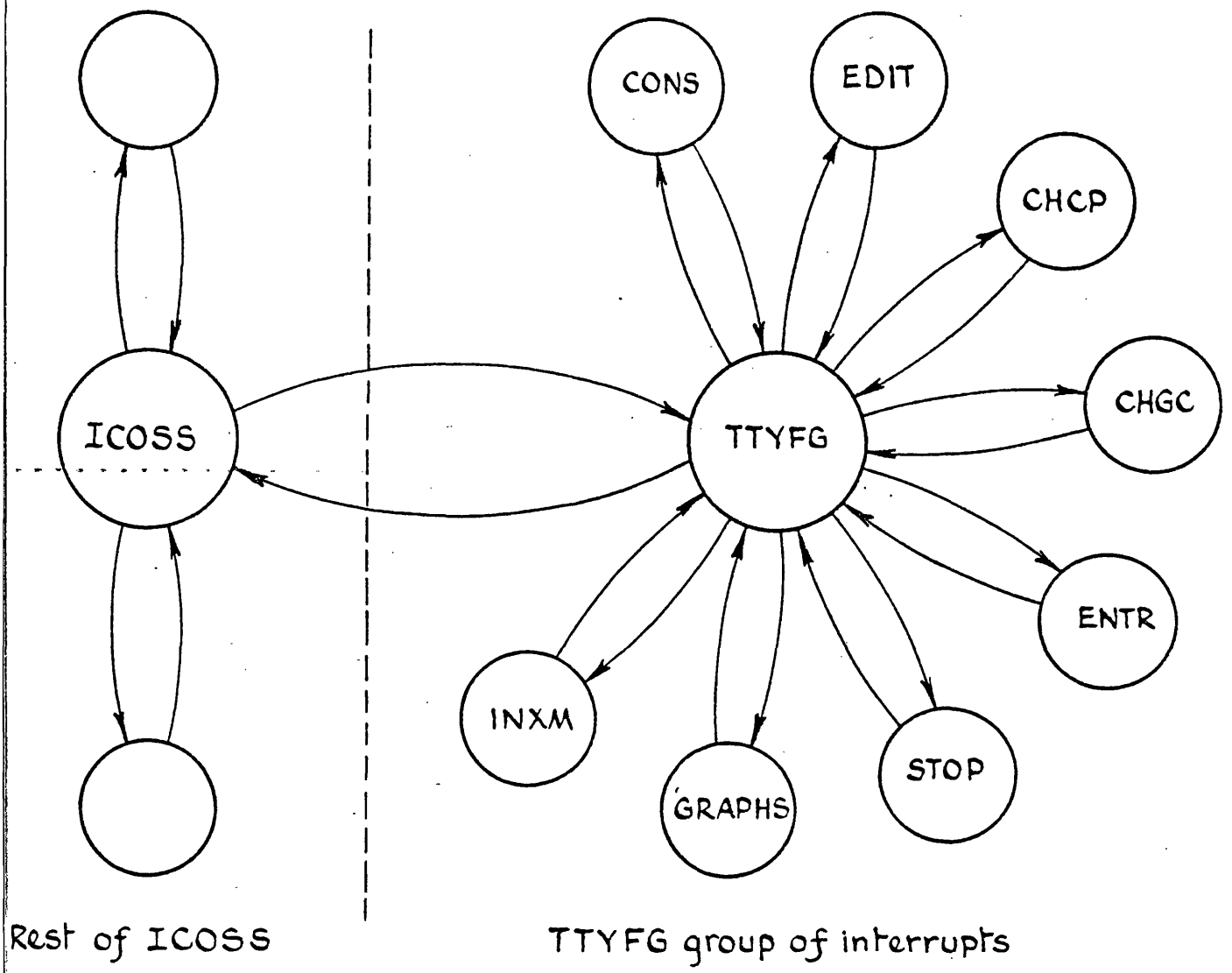


Fig. 379 State Diagram of TTYFG Group of Interrupts

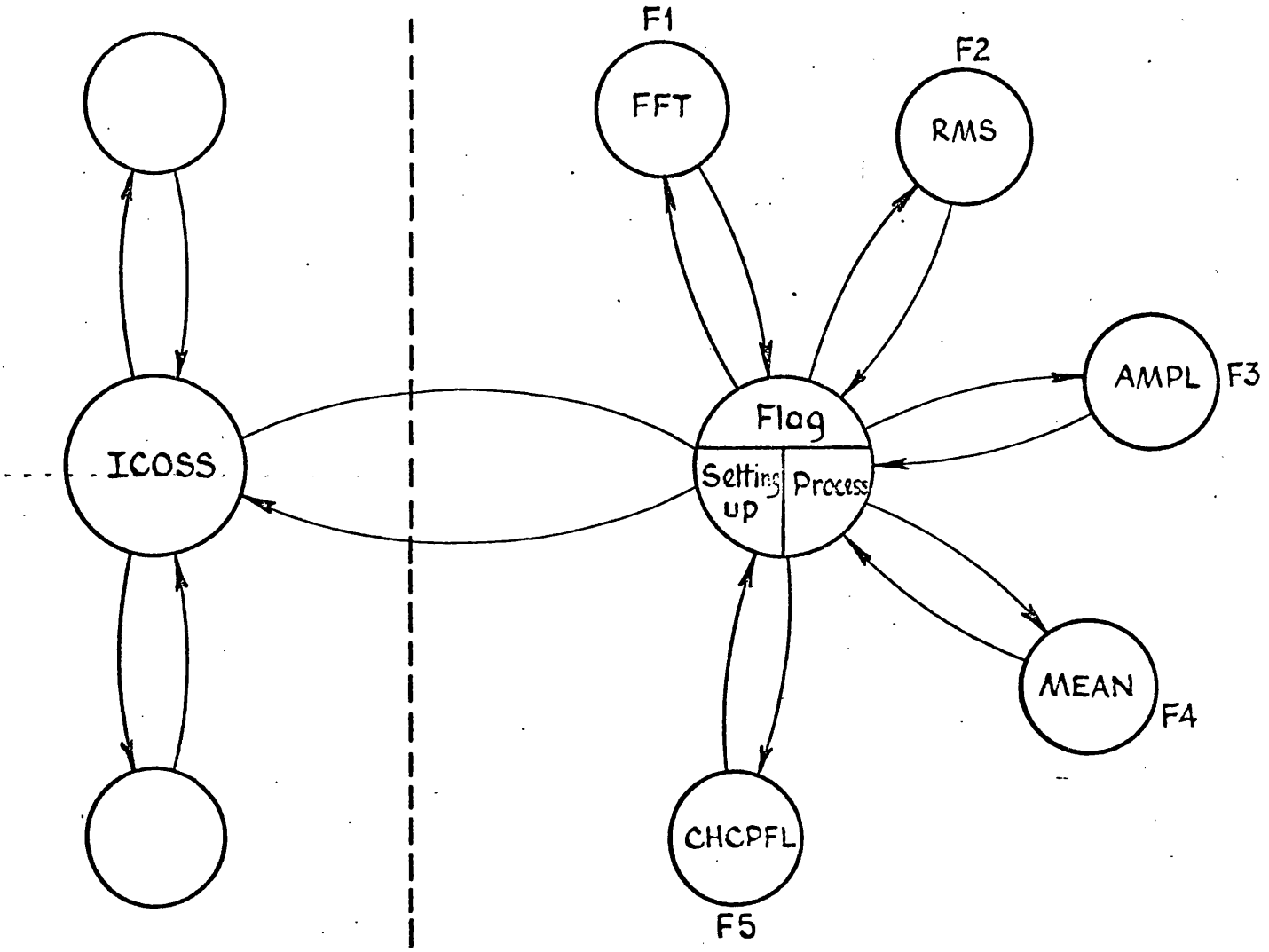


Fig.381 State Diagram of Flag Group of Interrupts

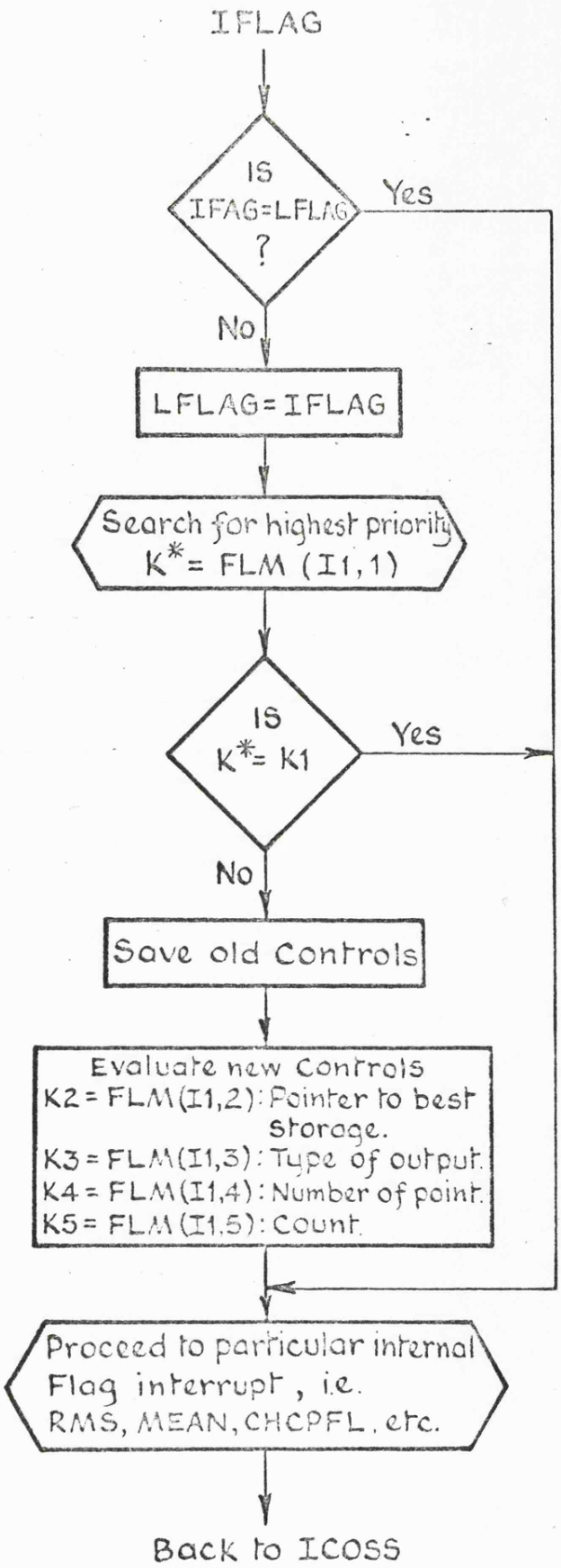


Fig. 3.8.2 Flow Chart for the Flag Group of Interrupts

CHAPTER 4

MICROPROCESSOR APPLICATION: BANK OF DIGITAL FILTERS (BOF)

4.1 INTRODUCTION

Having presented ICOSS and the areas for further developments, the present Chapter is devoted to one area only, namely the improvement in the speed of execution. The reason for this choice is to direct the process of discussion toward the original target, which is real-time time-domain simulation, in which speed of execution is an inherent problem. There are two ways of speeding up the execution.

The first, is having a large (Giant) computer with right execution speed, which may prove to be expensive and impractical, in the University research environment. The second technique, is having multiprocessing system in which certain parts of ICOSS process is extracted out of the main (Host) computer memory, and operated on by another, high speed "Slave" processor, concurrently. In this chapter, the latter technique is examined, by testing a time dependent section of ICOSS working in conjunction with the main simulation. There are a number of candidates (sections) which can be chosen from, for this purpose. The main ones are:

(a) any of the FLAG group of interrupts (FFT, MEAN,etc).

(b) The LOOP interrupt (the string itself).

(c) Any of the signal processing modules.

However, the section of ICOSS chosen for this purpose, is a signal processing module, for the following reasons:

(a) The desirability of having a flexible signal processing module, in its function, structure, programming, and its addition to ICOSS memory, with maximum ease.

(b) Building a signal processing module on a single board, separately, will enable the user to program and reprogram the module completely independently from the main (Host) computer.

(c) The ICOSS library will act as a shelf where the signal processing modules (semihardware) are added to or taken from the library depending only on the application and requirement.

The signal processing module implemented, simulates a Bank of digital Filters (BOF) - of any type, Chebychev, Butterworth etc. But number of filters simulated at any one time within a system is limited to two, and the order of any of those filters must not be higher than four. Each filter is made up of number of second order sections whose coefficients, although stored within the module itself, are provided by the Host computer during the construction mode.

Since the main source of error, when treating signal pro-

cessing of high resolutions, is the arithmetic and rounding off errors during the simulation, the choice of "word length" for the signal value, as well as the method adopted in the arithmetic calculations (linear or logarithmic) are of vital importance.

Since the Motorola MP6800 Kit (which comprises a microprocessor ROM and RAM) is available at the place of research, it was used to investigate the design of BOF system.

4.2 MICROPROCESSOR SYSTEMS

4.2.1 Introduction

Microprocessor based systems have been developed in recent years, as a natural progress of semi-conductor technology. These systems are developed in order to satisfy ever increasing and challenging requirements of the user. In this section, concentration will be made, not on the microprocessor technology of today as such, but on the services that it provides in order to reach the final design requirements of ICOSS.

A microprocessor system is basically made of a microprocessor element, read only memory, random access memory and a data interface. Each part of this system will be introduced and suggestions are made to the type of microprocessor arrangements which can be utilised in ICOSS system.

4.2.2 Microprocessor Architecture

Since microprocessors are the "heart" of computers, then a typical computer structure must be introduced[43] before describing the general form of a microprocessor architecture. Fig 4.2.1 shows a typical computer in its most simplified form; whereas Fig 4.2.2 shows the basic structure of a microcomputer which is related to microprocessors in a more obvious way. It is clear from Fig 4.2.2 that there are six basic parts to make a microcomputer. They are:

(a) Clock
(b) CPU
(c) O/P ports
(d) I/P ports
(e) ROM
(f) RAM

The term microprocessor is sometimes used in a general way to mean this microcomputer structure, in contrast to the transducers, actuators, displays and other things which are needed to meet the requirements of the instrument. More typically, the microprocessor is used to designate the large-scale integration (LSI) chip, or integrated circuit, which includes the CPU.

Sometimes (particularly among the early microprocessors to appear on the market) some standard integrated circuits arrangement the microprocessor chip in order to construct

a fully functioning CPU which can communicate with, and control, input ports, output ports, RAM and ROM. On the other hand, some microprocessor chips include not only a self-contained CPU but also one or more of the parts of the microcomputer such as the clock or input/output ports. Historically, Wilkes[4] (in 1950), suggested that configuration shown in Fig 4.2.3, as the microprocessor. However, that final and most general universal form of microprocessor, is the one shown in Fig 4.2.4, whose internal structure[43]

(a) General purpose registers: accumulator
(b) Arithmetic logic unit (ALU)
(c) Memory address register
(d) Program counter (PC)
(e) Instruction register
(f) Instruction Execution control logic

This is only the basic structure, and the actual set up varies from one manufacturer to another [43]. The basic difference is the size of word they are capable of handling, and cycle time. Obviously, the longer the word length (number of bits) and the shorter the cycle time, the better. As examples:

	Word Length	Cycle time
Motorola MP 6800	8 bits	1 μs/instruction (minimum)
Plessey MIPROC 16	16 bits	350 ns/instruction

Microprocessor Operation

A microprocessor incorporates the various functional units above in order to supervise and manage the operations of a system. Besides the control circuitry, the microprocessor has the ALU and the other registers which provide a temporary storage[37]. The accumulator constitutes the one essential general-purpose register. It can serve both as the source and as the destination register for operations involving some other registers, the ALU, or memory. Other general purpose registers often included in a microprocessor can be used to store operands or intermediate data, thereby lessening the possibility of accumulator bottlenecks[2]. Additional registers have dedicated uses. The PC, for example, keeps track of program instructions by maintaining the address of the next instruction in memory. The fetch instruction (top code) goes to another dedicated register, the instruction register, and is decoded by internal logics.

The microprocessor tackles each instruction in sequence. It proceeds from numerically lower memory addresses that give the instructions to be executed early, to higher addresses that give later instructions. However, the sequential order can be broken by "jump" instruction, which directs the microprocessor to a different part of the program. The order is broken by a "call" instruction that gives rise to the execution of a subroutine. Prior to its handling of a subroutine, a microprocessor makes use of a storage area, the stack, which may be either on the chip

(a hardware stack) or in memory (a software or pointer stack). The stack is used to save vital microprocessor information, such as the address in the program counter, while the subroutine is being executed. The information saved can then be used to resume operation of the main program once the subroutine has been executed. Stack is also used to reset subroutines, the extent of this capability is limited by the depth of the stack and its ability to store return addresses following each subroutine.

4.2.3 Read Only Memory ROM

Read only memory (fixed memory) is any type of memory that cannot be readily rewritten and ROM requires a marking operation during production to permanently record program or data pattern in it. The information is stored on a permanent basis and used repetitively. Such storage is useful for programs or tables of data that remain fixed and is usually randomly accessible. However programs can be rewritten, with some complication, on some types of read only memories outside the manufacturing environment.

Classifying ROMs, there are three general classes (types):

(a) Masked ROM: in which the memory contents (bit pattern) is produced during actual fabrication of the chip by the manufacturer by means of a masking operation.

(b) Field programmable ROM (fusible link): The memory

contents are programmed by the user with a special PROM programmer as part of the process of fabricating an instrument.

(c) EPROM - floating gate - (erasable programmable read only memory): The memory contents are programmed by the user but can be subsequently erased. This permits a unit to be reprogrammed, should changes be desired in the original programming. In contrast ROMs do not permit changes at all, and PROMs only permit previously unprogrammed bits to be programmed at a later date.

Any one of the above three ROMs will provide efficient system operation[37]. Since the purpose of making use of a microprocessor controlled system in ICOSS, is to have flexible system in which the signal processing module is separately programmed, the module may contain any type of process according to the situation and modification to it may be required at later application. Therefore EPROM is the overwhelming choice for the signal processing program of the module. However PROM may find some usefulness in some parts of ICOSS operation, in the control side of the complex interrupt system within ICOSS.

4.2.4 RAM

The term RAM (random access memory) is given to LSI type chip which data associated with Up operation is stored and later accessed. For this reason a more appropriate designa-

tion is sometimes used to indicate RAM, is a read-write memory, indicating the ability first to write data into it and then later to read data out.

A. The Basic RAM[43]

Every RAM has input address lines, data input lines, control inputs and data output lines. A memory plane contains memory cells which stores the data bits. An input-address decoder directs a read or write request to the desired memory location.

To write into the RAM, the data is placed on the data-input lines. Then the address of the desired memory location is placed on the input address lines. Finally, the write-control line is brought to the appropriate voltage level.

To read data from the RAM, the address of the desired memory location is placed on the input address lines, the read-control line is brought to the appropriate voltage level, and the data are read on the RAM output data lines. Fig 4.2.5 shows the basic structure of the basic RAM.

B. RAM structure

The typical structure for the RAM used in conjunction with microcomputers is that shown in Fig 4.2.6, with WRITE, ENABLE OUTPUT, and one or more chip Enable control inputs.

It may have more than 8 address lines, for larger capacity of RAM within one chip. It may have a word length of only one bit or of 4 bits, requiring the interconnection of several chips to obtain the desired word size, as in Fig 4.2.7. Also it may have data output lines which are distinct from the data input lines. For use with a data bus, corresponding input and output lines need only be tied together.

C. Types of RAM

(a) Static RAM: The term static RAM means that they will retain their data reliability (as long as power is maintained) regardless of whether any of the inputs change.

(b) Dynamic RAM: This type places specific requirements upon how often data must be accessed if it is to be retained. For example, Intel 2107 (4096x1) dynamic RAM requires a "READ" cycle to be performed on each of the 64 possible combinations of six address lines once every 2 ms. However, these 64 read cycles can be performed with the output actually disabled. Consequently, the refreshing of all dynamic RAM chips can be undertaken with a data selector which switches these six address lines from the address bus to the output of a 6 bit binary counter whenever there is an operation not involving RAM. Then a special refresh input is stored. The 6 bit counter can be counted with the trailing edge of the strobe pulse.

(c) Static/dynamic combined: Combining static and dynamic memory techniques, the chip achieves a maximum access time say (200 ns - typically 150 ns - for the MK 4104); and maximum cycle time of 260 ns for the same chip[32] . Yet it dissipates typically only 80 mw of active power at 4 MHz and very low 8 mw in standby. An additional low-power mode of 1.0 mw is available for battery back-up operation, achieved simply by lowering the power supply voltage from 5 volts to 2-3 volts.

D. Application RAM

The RAM actually used in the project is the 128x8 bit static random access memory (MCM 6810). It is a byte organised memory designed to use in bus-organised systems. It is fabricated with N-channel silicon-gate technology. The RAM operates from a single power supply, has compatibility with TTL and DTL, and needs no clocks or refreshing because of static operation.

The memory is compatible with the M6800 microcomputer family, providing random storage in 128 byte increments. Memory expansion is provided through multiple chip select inputs. Some of the important features of the RAM are:

(a) Organised as 128 bytes of 8 bits

(b) Static operation

(c) Bi-directional, 3-state data Input/Output

(d) Six chip select inputs (4 active low, 2 active high)

(e) Single 5 volts power supply

(f) TTL/DTL compatible

(g) Maximum access time 1.0 μs for MCM 6810L.
575 ns for MCM 6810L-1.

4.3 CONFIGURATION

One of the limitations self-imposed on On-line simulation is the computer processing power. The tendency in improving the processing power is to enlarge the computer at hand with much increased expense. In fact a new generation of large general purpose computers, sometimes termed super-computers, have recently been introduced, providing speed sufficient for even the most ambitious simulations, however at very high cost[42] . Therefore, the overall cost and size of a general purpose computer has limitations as well.

In order to improve the processing power, and remembering that a main computer operates sequentially, either of the following approaches can be made:

(i) Parallel processing (genuine), in which results of operations may come out simultaneously. The difficulty with this mode of operation is a management problem, ie synchronising the result together, hence increased programming complexity.

(ii) Sequential processing, but putting out tasks to special-purpose hardware which executes

processes very quickly. Programming this type of computer configuration is easier. Therefore, this method will be adopted in the present work for ICOSS development, in which some of the processes, eg a signal processing module of the loop group of interrupts, is taken out of the main computer and processed by another, small inexpensive fast processor.

In this section the concept of "multiprocessing" is introduced and the final set-up, "the master-slave", configuration employed for ICOSS development is described.

4.3.1 Multiprocessing systems(*)

A multiprocessing system uses more than one central processing unit in the system configuration. There are many reasons for multiprocessor systems:

(a) Greater system efficiency and use of system resources.
(b) Increase in system capabilities in responding to real time situations.
(c) Fault tolerance: the greater ability to deal with system malfunctions.

There are a number of different ways to classify multi-processor systems, based on the following characteristics:

(a) The type of processors.

---

* Refs 13,37,42,45

(b) The interconnections between processors.

(c) The relationship of the processors to memory and I/O units.

(d) Operating software for the processors and the systems.

On the basis of these characteristics, one can refer to certain processors as being "array" processors, "pipeline" processors, "ring" processors, "parallel" processors, or "reconfigurable" processors. One can also describe the system structure as being "tightly coupled" or "loosely coupled".

Although microprocessors may be utilised for various functions in a multiprocessor computer system, the most interesting concept is a multiprocessor system constructed of a plurality of microprocessors. The particular type of multiprocessor architecture that is particularly worth considering is a reconfigurable architecture using microprocessors.

Reconfigurable microprocessor architecture: a multiprocessor computer system in its most basic form consists of a determined configuration. At certain times it may be desirable to change this configuration based on a particular internal or external event. A computer system which possesses the hardware or software capabilities to implement such configuration is called a reconfigurable architecture. One of the main applications of this type of computer system is the interactive multiprocessor system.

In this system, simultaneous processing by a relatively large number of discrete users running jobs with different characteristics, take place. To increase throughput, a reconfigurable system operates by allocating an optimum number of processor memories and I/O units to each respective user. In this sense the system reconfigures itself by partitioning the system into independently operating units, either on a space-division or time-division multiplex basis.

Microprocessors may be utilised to implement computer systems based on such reconfigurable architecture. The basic system can be implemented by means of a multiprocessor array, together with supervisory and data transfer function controlled by other processors. The arrangement between these processors determines the types of multiprocessor systems implemented. There are several basic types of multiprocessor configurations which may implement a reconfigurable system. The type which will be adopted for ICOSS development is called Hierarchial system[37]. This system is based on one "master" processor and two or more "slave" processors in a hierarchial relationship. The master processor controls or supervises the operation of the "slave" processors in either a "tightly" or "loosely" coupled manner as shown in Fig 4.3.1. A further description will be made on this system in subsection 4.3.2.

There are other types of configurations, namely:

(i) Parallel

(ii) Ring

(iii) Switched

(iv) Pipelined

which can be found in the literature(*)

4.3.2 Master-slave processors

The hierarchial type configuration mentioned earlier is an alternative approach to the attainment of very high-speed real time capability, in which moderately sized general purpose digital computer and a special purpose digital processor are interconnected as shown in Fig 4.3.2. The host computer which acts primarily as a buffer to the communication lines and I/O equipments can be a large minicomputer eg PDP 11/70. The "peripheral processor" is a digital computer employing a high degree of pipelining and parallelism. Because it is designed solely to facilitate high speed arithmetic computation, it is capable of providing computing speed considerably superior to those of large general purpose computers at a very moderate cost (comparison: $8,000,000 old technique, $300,000 new technique).[42]

In the present work, the general-purpose computer at hand is the Digital Equipment PDP8/e, which will contain the main body of ICOSS (see later); while the slave processor is a microprocessor controlled signal processing module.

---

* Refs 13,37,42,45

4.3.3 Signal Processing Module

The slave processor of the hierarchial system adopted for ICOSS development is made of:

(i) Microprocessor element
(ii) Random access memory RAM
(iii) Read only memory ROM*
(iv) Arithmetic unit AU

The internal connection of these elements and the processor external connection is shown in Fig 4.3.3. The signal processing module chosen out of the present ICOSS library for the slave processor is a "bank of digital filters - BOF". BOF subroutine will be stored in RAM but eventually the final module program will be stored in ROM, whereas the various signal values, state variables and control variables will still be stored in RAM. The arithmetic unit, which performs the logarithmic number manipulation is coupled within the processor. The suggested type of this unit, is a modified logarithmic arithmetic unit, see Appendix F for detail.

There are a number of advantages in choosing signal processing module for slave processor, mainly:

---

* To be more precise - EPROM - erasable programmable read only memory should be used

(a) The obvious increase in computer processor power, hence better on-line simulation.

(b) The basic signal processing modules are separated from the main ICOSS body. This means that the ICOSS library will act as a "shelf" for the various signal processing modules, and each new module is prepared (designed, programmed etc) independently outside ICOSS. However in the signal processing library of the prototype ICOSS discussed earlier, the modules identification procedure has to be modified, for true interchangeability. The changes in the main ICOSS will be on interfacing and identification, which can be made minimal.

There remain a number of supplementary devices which have to be treated, for the final system design, in order to make the operation more compact, such as the power supply, interfacing, console control etc, see Fig 4.3.4.

Another improvement, which can be made is to make the slave processor as a single chip, coupled to the main computer via an easy interconnector (interface).

4.4 DESIGNING A SLAVE PROCESSOR FOR THE BOF MODULE

4.4.1 General

A slave processor is to be designed which simulates a bank of digital filters (BOF), having the following

characteristics and considerations:

(a) The maximum number of filters at any one time must not be more than two. This limitation is for the present set-up, since the number of filters is limited by the size of memory (RAM) available in the slave processor.

(b) The highest order of any one filter must be defined in advance, depending again on the memory (RAM) available; and it was decided that the highest order of any one filter for the present set-up is to be four.

(c) Wide dynamic range, in which case logarithmic numbers for signal value representation is used.

This section is mainly concerned with the theoretical background and the hardware requirements; whereas in the following section, a more specific development of the BOF module will be made.

4.4.2 Theory

A digital filter can be represented,in time domain, as cascaded second order segments [1] , as shown in Fig 4.4.1. Assuming that the input signal is x(t) in time domain, and x(n) in sampled form, and the output signal is y(t) in time domain, and y(n) in sampled form, the filter transfer function

$$H = a_o \prod_{i=1}^{i=k} hi$$

where hi is the second order segments

and k is the number of those segments in the filter

and $$h_i = \frac{1 + a_{1i} z^{-1} + a_{2i} z^{-2}}{1 + b_{1i} z^{-1} + b_{2i} z^{-2}}$$

where a_{1i}, a_{2i}, b_{1i} and b_{2i} are the segment's coefficients.

Representing the filter transfer function as difference equations:

$$y_i(n) = x_i(n) + a_{1i}x_i(n-1) + a_{2i}x_i(n-2) - b_{1i}y_i(n-1) - b_{2i}y_i(n-2)$$

for i = 1,2,...k

Thence the final output $y = a_o\, y_k(n)$.

4.4.3 BOF module design requirements

In order to build any signal processing module using a microprocessor control unit incorporated with a host computer, the following requirements must be met:

(a) A program for the signal processing module: this is a machine coded program stored in ROM.

(b) Storage area: enough memory space (RAM), must be provided for the storage of:

(i) control parameters of the signal processing module
(ii) state variables
(iii) other supplementary parameters
(iv) the program (a) above, in the case of the prototype only. In the final design, the latter is stored in EPROM.

(c) Interfacing arrangements with the host computer.

4.4.4 Module Structure (hardware)

The basic structure of the "slave" processor is shown in Fig 4.4.2. The figure shows a prototype model, containing the following units:

(a) RAM: random access memory, which contains

(i) the signal processing module program
(ii) module control parameters
(iii) module state variables
(iv) supporting parameters, etc; as will be explained in the next section (4.5).

(b) MP: microprocessor, the Motorola MP6800 [34].

(c) AU: arithmetic unit; the modified logarithmic arithmetic unit; see Appendix F.

(d) PIA: peripheral interfacing adaptor. This is a parallel lines, for data and control. It is mainly used for the intercommunication of data between the "slave" processor and the host computer. For the present work, it is 8 bi-directional lines, which can be programmed, controlled and treated as simple memory locations.

(e) ACIA: asynchronous interface adaptor. This is a serial data line, used mainly for the control and data transfer of teletype or console.

(f) TTY: teletype.

(g) ROM: read only memory: contains the Microprocessor supporting software

(h) Power Supply: ±12 volts, + 5 volts, and earth. All data and control voltages are TTL compatible.

4.5 SOFTWARE DEVELOPMENT OF BOF MODULE

4.5.1 Program Formulation

Following the digital filter formulations developed in 4.4.2 earlier, and assuming that a fast arithmetic unit, similar to that suggested in Appendix F is used, and whose function is limited to:

$$a = b \cdot c + d \qquad (1)$$

Also considering one second order segment of the filter, whose difference equation is:

$$y_i(n) = x_i(n) + a_{1i}x_1(n-1)+a_{2i}x_i(n-2)+b_{1i}y_i(n-1)+b_{2i}y_1(n-2) \qquad (2)$$

This equation (2) can be split into four smaller equations in the form of equation (1), as follows:

$$\begin{aligned} w_{1i} &= x_i(n) + a_{1i}x_i(n-1) \\ w_{2i} &= w_{1i} + a_{2i}x_i(n-2) \\ w_{3i} &= w_{2i} + b_{1i}y_i(n-1) \\ w_{4i} &= w_{3i} + b_{2i}y_i(n-2) \end{aligned} \qquad (3)$$

With the necessary reshuffling of the segments state variables as indicated earlier, this can be repeated (k) times for k segment filter, and the final output is obtained as:

$$y_{out} = a_o\, y_k(n) \qquad (4)$$

Hence with the use of the arithmetic unit (AU) and simple access and reshuffle procedures, the process becomes simply a matter of repetition, reshuffling of numbers in the state variables stack, and temporary storage for the AU function, as illustrated in Fig 4.5.1.

Execution time determination

In order to determine the time taken for one signal sample being processed by a filter having k second order segments, there are three periods to consider:

(i) The AU instruction time:

Assume T_{AU} is the AU instruction time.
There are four basic AU instructions/segment.
If equation (4) is rewritten as:

$$y_{out} = a_o y_k(n) + O \qquad (5)$$

and (5) is considered as one AU instruction, then:
The total AU instructions needed for the filter:
4k+1
The total AU time needed for the filter:
$(4k+1)\ T_{AU}$.

(ii) The state variables shuffling time:

Assume T_{sh} is the shuffling time/instruction
There are 4 x T_{sh}/segment
∴ The total shuffling time needed for the filter = $4k\ T_{sh}$.

(iii) Control parameters access time:

Assume T_a is the access time/parameter

There are 4 x T_a/segment

$\therefore$ The total control parameter access time

= 4 x k T_a.

$\therefore$ The total time needed (execution time) =

$$4(T_{sh}+T_a)k + (4k+1)\ T_{AU}$$

However, this is the minimum time needed and some more time needed for the microprocessor management.

4.5.2 Implementation

Microprocessor controlled module is a low level device, ie accepts machine code programs only. The user, who usually writes his program in a higher level language, such as assembly language, has to compile and assemble his program in order to produce the desired machine code program. An Excorcisor is used for this purpose, which has a system supporting software, the Editor and the Assembler.

The procedure to produce the machine coded program for the microprocessor controlled module is therefore as follows:

(a) Formulate the module program, in the way explained in subsection 4.5.1.

(b) Draw flow chart.

(c) Write the program in assembly language.

(d) Run through the Editor and Assembler.

(e) the final product is the machine code program.

4.5.3 BOF Program

There are two distinct functions in BOF module, which the program must perform:

(i) Housekeeping: The filter control parameters (secondary) as supplied by ICOSS, are stored in their allocated area in RAM, with the necessary identifiers for the relevant filter. Also allocations to the state variables of the filter are made, and the relevant pointers are recorded.

(ii) Signal Processing: This is the filter action to the input signal.

However, there is another fixed procedure in BOF program devoted to the interfacing problem and transmitting and receiving data to and from the main (host) computer, respectively.

In the following sub-sections detailed descriptions are made to each function of BOF program.

4.5.3.1 Interfacing

In the MP6800 microprocessor there are two sets of data and control lines (registers), Fig 4.5.2: set A and set B.

Each having 8 parallel bi-directional lines and number of control lines, which are programmable by treating each set as a memory location. In the present BOF case, set A was assigned to Receive from host computer side, and set B was assigned to Transmit to host computer side, of the microprocessor. Each side has:

(a) 8 lines (registers) - bi-directional
(b) Cx1 register for flagging-data ready
(c) Cx2 register for acknowledgement of job done
where x is either side A or side B.

In programming the interface, the following procedure is followed:

(a) Clear registers
(b) Open interrupt
(c) Check flag register for any changes in Cx1 registers
(d) Transfer data
(e) Send acknowledgement flag via Cx2 registers

All voltages are TTL compatible which matches with the host computer voltage, Fig 4.5.3.

4.5.3.2 Memory Map

One of the main limitations imposed on the design of a microprocessor software is the availability of memory space in RAM. The first step, therefore, is to divide the

memory locations available into dedicated sections, producing the microprocessor memory map. In the present work for designing BOF program, the memory is divided into the following sections, Fig 4.5.3.:

(i) PIA locations: These are fixed by the present hardware, and used in conjunction with the interfacing and transmitting and receiving of data. In the case of the Motorola MP6800 microprocessor, there are four bytes addressed by: $8008 - $800B (where $ indicates Hexadecimal).

(ii) Internal parameters: ($0000 - $0009)
All the parameters necessary for the subsequent operation of the module, such as counters, filter number, temporary storage of input and output signal.

(iii) Module control parameters: ($0010 - $0022)
The control parameters are supplied from the host computer in their final form and stored at their appropriate positions. There are 18 locations reserved, for two 4th order filters. Remembering that each filter has 9 control parameters; see subsection 4.4.2.

(iv) State variables: ($0023 - $0032)
The locations for the state variables are reserved, and initialised to zero. Sixteen locations are reserved for the present case of two filters of 4th order; see subsection 4.4.2.

(v) Initialisation ($0035 - $0044)

The portion of the program concerned with the system and interfacing initialisations is stored in this area.

(vi) Program: ($0045 - $00F4)

The filter function part of the program is stored in this pre-reserved area.

(vii) Subroutines ($00F6 - $012D)

Similar to (vi), an area of the memory is reserved for the existing subroutine of module. In the present situation, there are two sub-routines, one for receiving data from the host computer, and the other for transmitting data to it.

4.5.3.3 Construction mode (CONS)

In this mode of operation, all the "house-keeping" of the BOF module takes place. These include:

(i) The filter identification: this is assigned by the module and recorded by ICOSS in the host computer.

(ii) The storing of the filter control parameters in module.

(iii) The initialisation of the state variables locations of the filter, ie storing zeros in them.

The process is triggered by a mode identifier from ICOSS, and the operation takes place as shown in the flow-chart in Fig 4.5.5.

4.5.3.4 The Run mode (RUN)

The filter action is actuated by the RUN instruction, which includes: mode identifier, filter number and the signal value; and then the input signal is processed by the appropriate filter, in the way shown by the flow-chart, Fig 4.5.6.

Since the arithmetic unit (AU), mentioned earlier, was not available, the investigation was mainly concerned with the module organisation, not the arithmetic one. The AU action was performed internally. (However, if the AU is interconnected, then it can behave as another assembly instruction, reducing the filter action time considerably).

4.6.1 THE HOST MACHINE: THE DIGITAL EQUIPMENT PDP8/e *

The PDP8/e is specially designed as a general purpose computer. It is designed to meet the needs of the average user, yet it is capable of modular expansion to accommodate most of the user requirements.

The PDP8/e basic processor is a single address, fixed word length, parallel transfer computer using 12 bit, twos complement arithmetic. The cycle time of the random access

memory is 1.2μs for fetch and defer cycles without auto-indexing and 1.4μs for all other cycles. Standard features include indirect addressing and facilities for instruction and skipping and program interrupts as a function of input/output device conditions.

Five 12 bit registers are used to control computer operations, address memory, perform arithmetic or logical operations and store data. A programmer's console provides switches and indicators that permit convenient monitoring and modification of machine states and major registers.

The 1.2/1.4 μs cycle time of the PDP8/e provides a computation rate of 385,000 additions per second. Each addition requires 2.6μs (with the addend in the accumulator), while subtraction requires 5.0μs (with the subtrahend in the accumulator). Multiplication is performed in 256.5μs or less by a subroutine that operates on two-signal, 12 bit number to produce a 24 bit product, leaving the 12 most significant bits in the accumulator. Division of the two signed, 12 bit numbers is performed in 342.4μs or less by a subroutine that produces a 12-bit quotient in the accumulator and a 12 bit remainder in memory. Similar signed multiplications and division operations are performed in approximately 40μs utilising the operational KE8-E Extended Arithmetic Element. Fig 4.6.1 shows a block diagram of the basic PDP8/e that illustrates the

---

* Digital Equipment: PDP8/e, PDP8/m and PDP8/f small computer handbook. Pub by PDP8/ handbook series 1973.

signal paths between the central processor, the memory system and the OMNIBUS. (In PDP8/e a bus is defined as a group of 12 signal lines carrying related information, such as the 12 bits of an instruction or data word. The OMNIBUS may be considered as a wide bus containing several buses, along with many other signal lines). Signals that do not pass through the OMNIBUS are routed between adjacent modules by means of Edge connectors.

4.6.2.1 Main program: ICOSS

This program controls the overall operation of the multi-processing system. Only two main interrupt systems are included in this modified version of ICOSS*. They are:

(a) the teletype group of interrupts
(b) the loop group of interrupts.

The program structure is shown in Fig 4.6.2

4.6.2.2 Teletype group of interrupts: TTYFG

The time independent interrupts which are grouped as "teletype" group of interrupts, are advocated for the construction of the simulated signal processing system, as

---

* See Chapter 3 for description of ICOSS

well as the "house-keeping" and control parameters management of the "slave" processor. The teletype commands employed for this group of interrupts are:

(a) CONS: for system construction.
(b) RUN: for running the signal processing.
(c) STOP: for logging out of the system simulation.

4.6.2.3 Construction Mode

The simulated signal processing systems construction is executed in the usual way(*), but with the additional problem of the parameters manipulations of the "slave" processor. Subroutine CPMAN is modified to include the later problem (see later).

There are only two basic modules included in the modified ICOSS library. The first is the signal generator (sinewave) and the second is the filter module. The reason for simulating the signal generator is because the real time operation is neither required, nor possible for this exercise.

Since the actual simulation will be off-line, then the scope action will be simulated as well. This requires a storage area for the accummulation of the output signal samples. The process is executed, as part of the teletype interrupts system similar to the original procedure of ICOSS.

---

* See Chapter 3

Control parameter manipulation: CPMAN

This is a two fold operation: the first is the usual control parameter manipulation which was discussed in Chapter 3. The second is the transfer of parameters to the "slave" processor in the case of the filter module. This means that the filter parameters will be processed by the host computer (accepting primary parameters from the user and calculating the final-secondary-parameters) which will then be stored in the main ICOSS body in the Host computer, as well as storing it in the "slave" processor as part of the filter module.

Subroutines

CONST : Overall controller for the construction mode.
BCONST : Basic element construction tool.
CPMAN : Control parameters manipulation.
SCONST : Scope action manipulation.
SIGN : Signal generator module (sine wave).
FILT : Filter module - as second order segments.
STORE : Store for further scope action manipulation.
LOOP : The running mode controller.

4.6.2.4 Running Mode

The final signal processing of the simulated system is executed as a loop interrupt of ICOSS in the usual way. However, in the case of the filtering action in the loop,

the signal is transmitted to the "Slave" processor, via the Host computer buffer, to be processed by the filter module of the "Slave" processor. While the signal is being processed, the simulator ICOSS keeps hunting for the output signal of "Slave" processor (the filter module). The waiting period is decided by two factors:

(a) The filter action execution time.
(b) Data transfer rate of the Host computer buffer.

4.6.3 The Host Computer interprocessor Buffer *

This interface allows the transfer of 12 bit digital data to or from the PDP8/e computer and provides signal lines for hand-shaking sequences between the processor and an external device (the microprocessor controlled unit).

The interface comprises a 12 bit Transmit Buffer and a 12 bit Receive Buffer, each with associated Flag input and pulse output signal lines. When the processor transfers data to the transmit Buffer or data from the Receive Buffer a 100 nsec pulse appears on the associated Pulse Line which may be used to indicate to an external device that data is ready to be transmitted or that data has been received. Each buffer has an associated device flag which is cleared by the processor IOT instructions. The flags may each be set to (1) by pulses from an external device

---

* Refr PDP8/e peripheral data sheet.
DB8 - E interprocessor Buffer.

indicating that data has been transmitted or that data is ready to be received. When enabled under software control either flag may cause an interrupt when set to (1).

Technical specification

Maximum transfer rate	One 12 bit word at approximately 5 kHz.
Data output	8 mA (5TTL loads) and still maintain standard TTL noise immunity. Series terminated with 100 ohms.
Rise and fall times	Less than 50 nsec without cable.
Data inputs	One unit load (1.6mA) each and clamped to -0.6 volts.
Data level	True at 3.0 volts. False at 0 volts.

4.7 RUNNING OF FINAL SYSTEM

(a) General layout

Having prepared both programs, ICOSS for the Host machine, and BOF for the microprocessor unit, the combined system may now be run. The final system configuration is shown in Fig 4.7.1; it must be noted that the address lines (16 bits) and the control lines are omitted from the Figure. A detailed block diagram of the microprocessor unit is shown in Fig 4.5.2. Utilising the facility provided in the microprocessor itself, section A of the bi-directional data line (8 bits) of PIA is used for Transmit of data from microprocessor unit to the Host computer buffer and Section B of PIA for the Receive of data from the Host computer buffer.

(b) Attempted runs

In attempting to run the combined system, there were two major obstacles:

(a) the arithmetic unit was not available.
(b) the transmit/receive rate of the PDP8/e buffer was very low, 5 KHz as mentioned earlier.

Since the BOF operation in the microprocessor unit is a signal processing function and has no bearing

on the multiprocessing action of the combined system, it was decided therefore to test only the interaction between the two systems; by circulating a number between them, which was performed accurately. However the following points are raised:

(i) The microprocessor data lines and the memory locations related must be increased to an appropriate number and compatible with the Host computer buffer. In the present case, the Host computer has 12 data lines, whereas the microprocessor unit has 8 data lines.

(ii) The transfer speed of the Host computer buffer must increase considerably in order to make the process advantageous.

(iii) The microprocessor speed can be increased as well, allowing more segments of digital filter in BOF*.

---

* A microprocessor unit the MIPROC 16 say, which has 16 data lines and 350 ns/instructions, will produce better results.

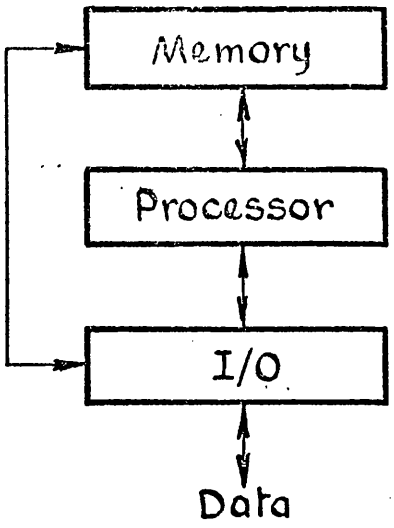


Fig. 4.21 Typical Computer Configuration

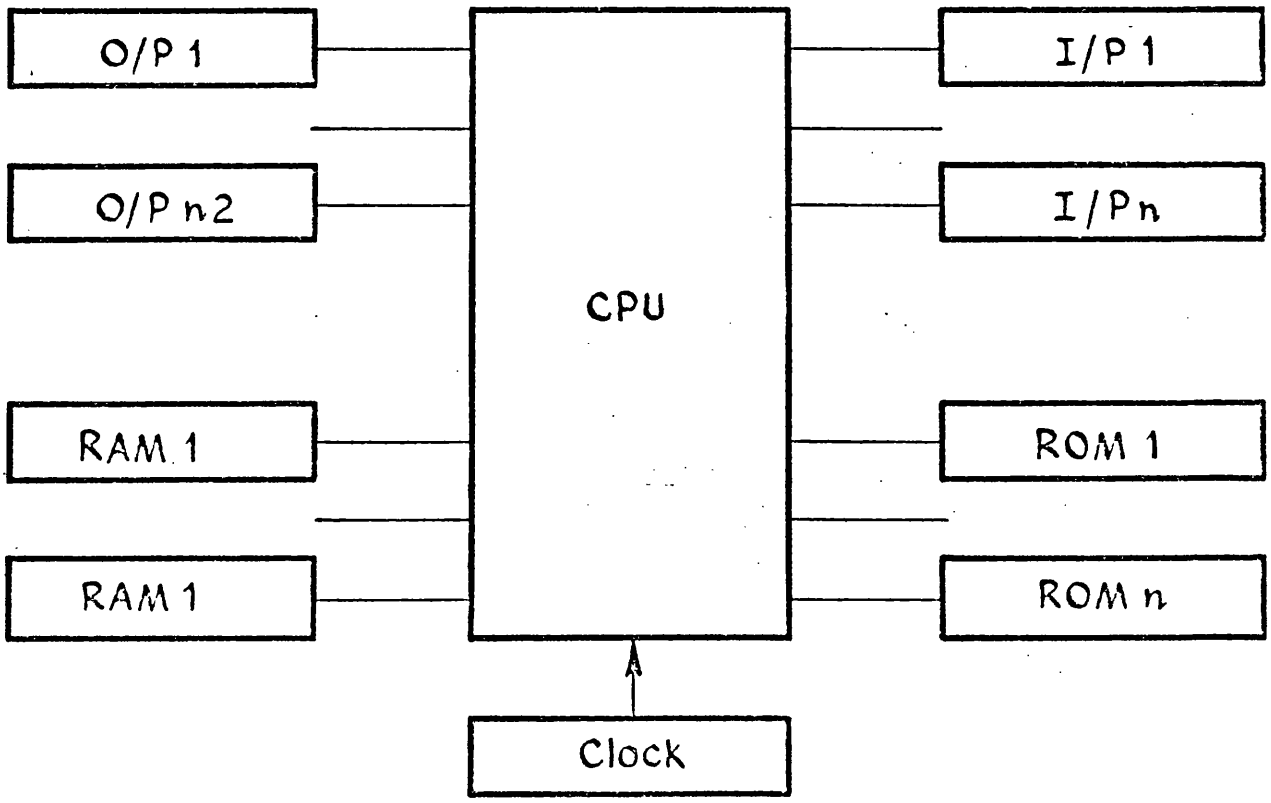


Fig. 4.22 Microcomputer Structure

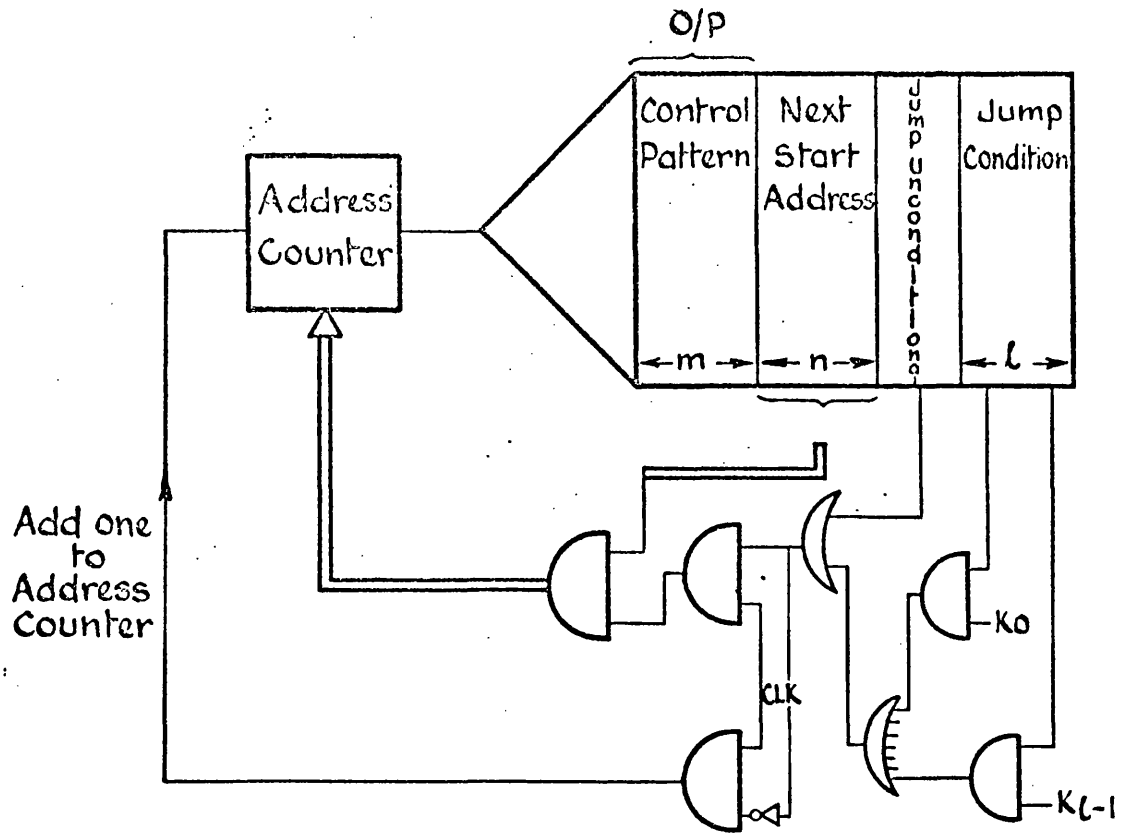


Fig.423 Wilkes Microprocessor

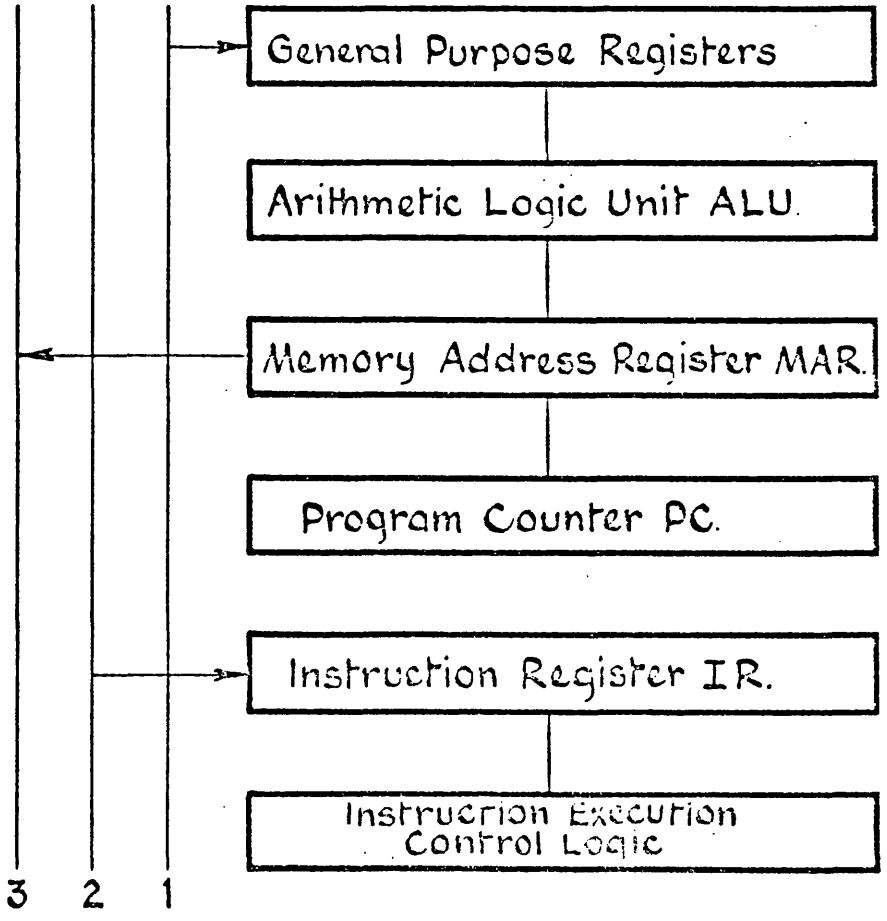


Fig.424 Internal Structure of a Microprocessor

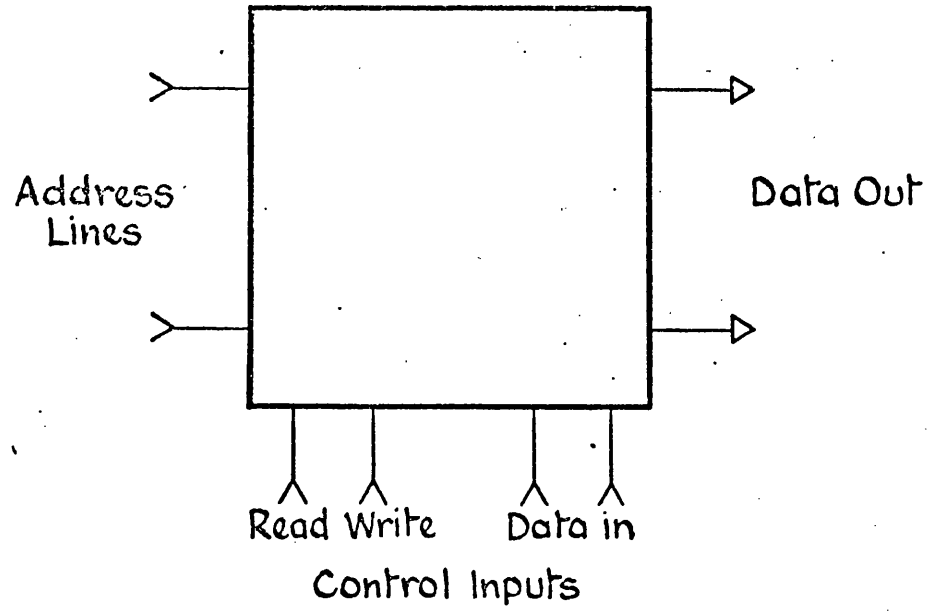


Fig. 425 Basic Structure of Basic RAM

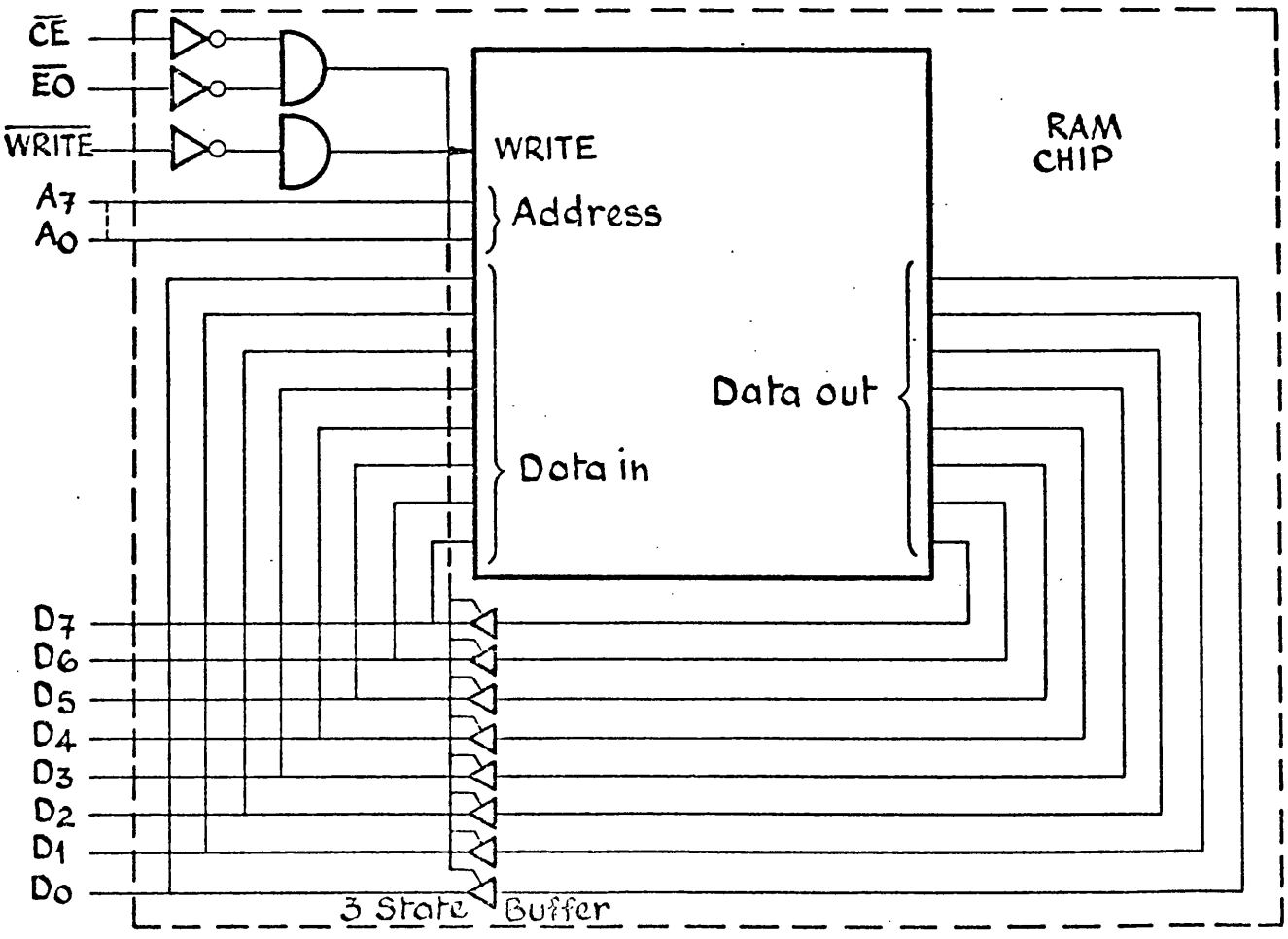


Fig. 426

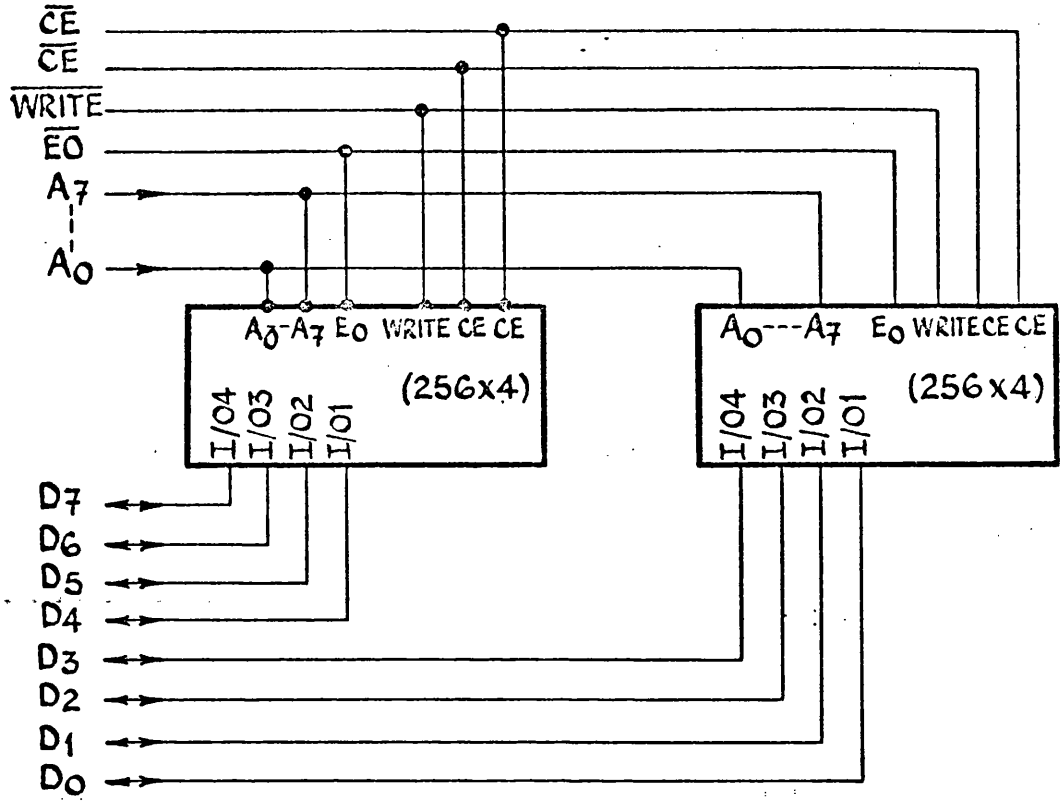


Fig. 4 27

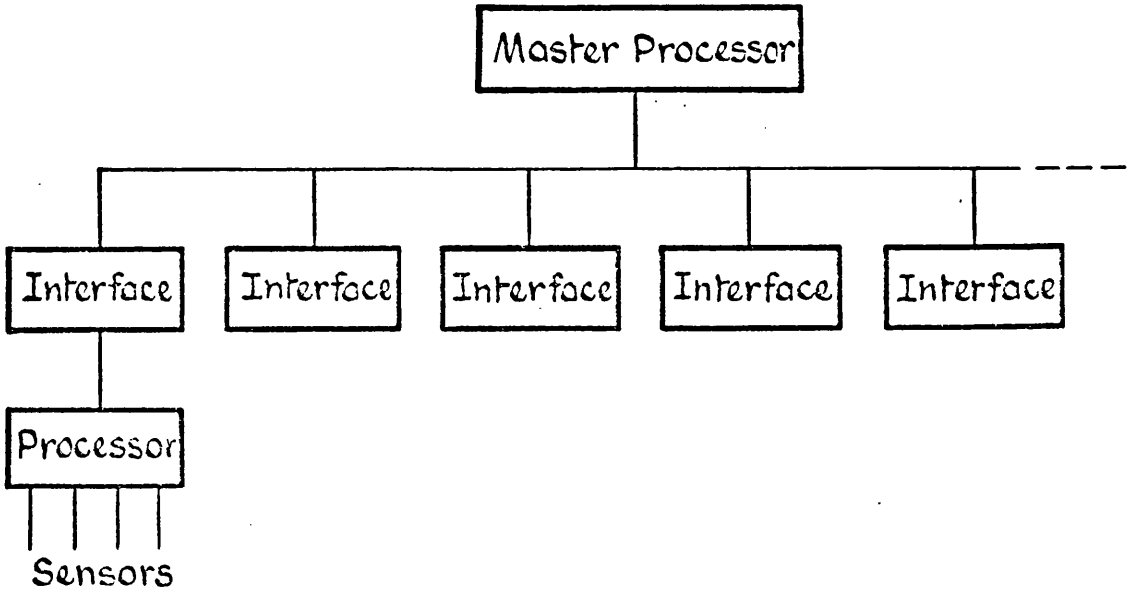


Fig. 4 31 Hierarchical System

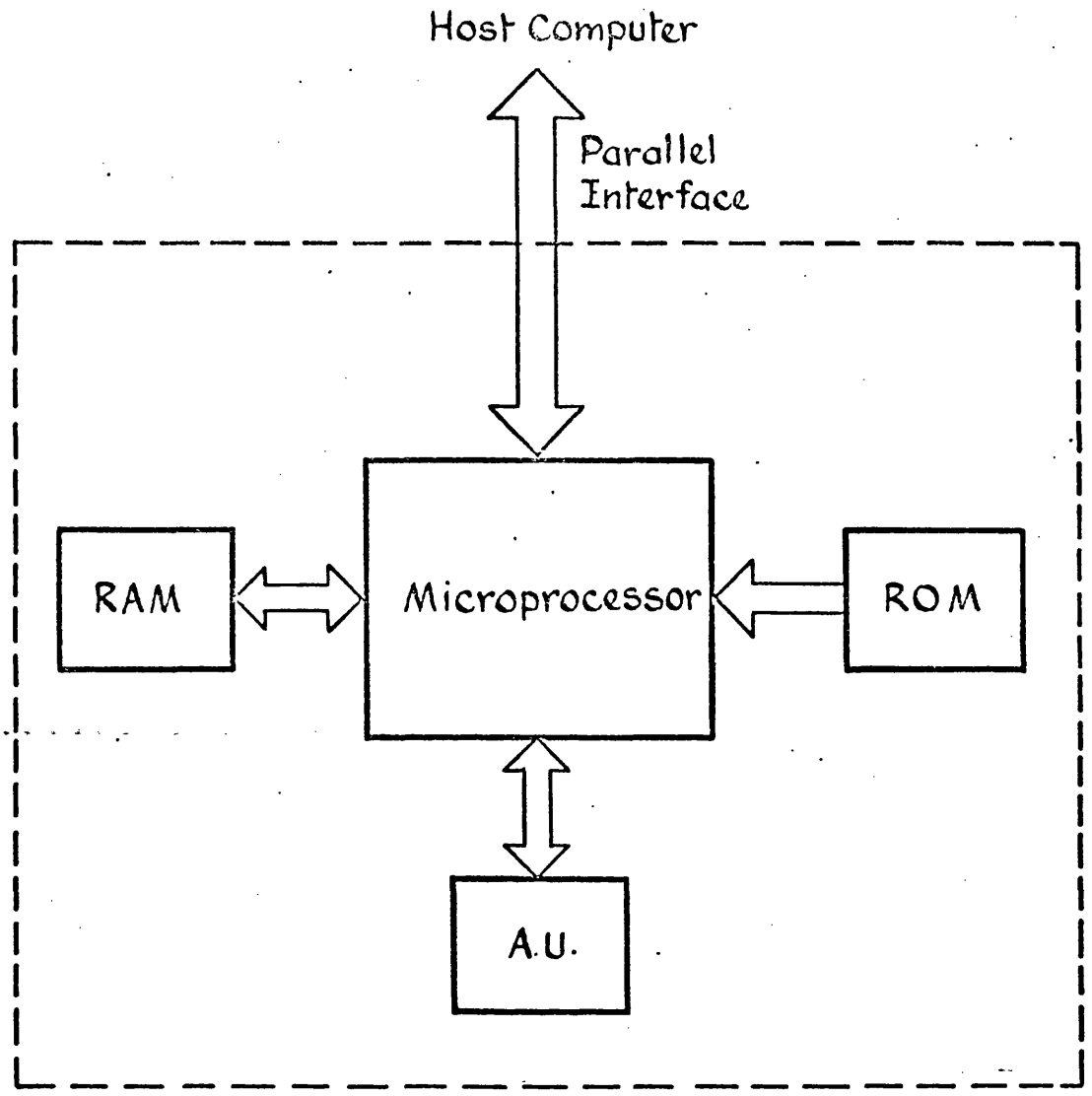


Fig. 433 Microprocessor Controlled System

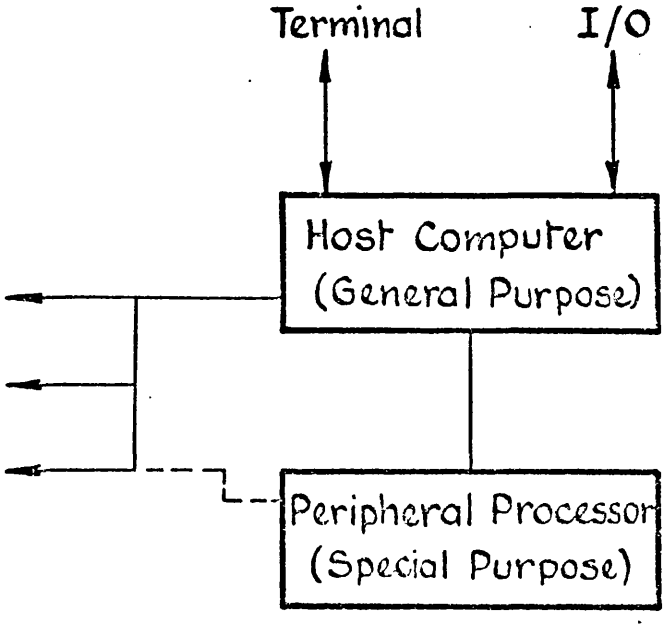


Fig. 433 Master/Slave System.

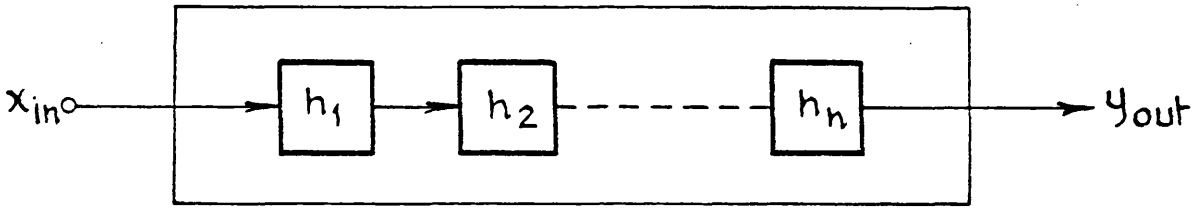


Fig. 441 A Digital Filter of n Second-order Segments

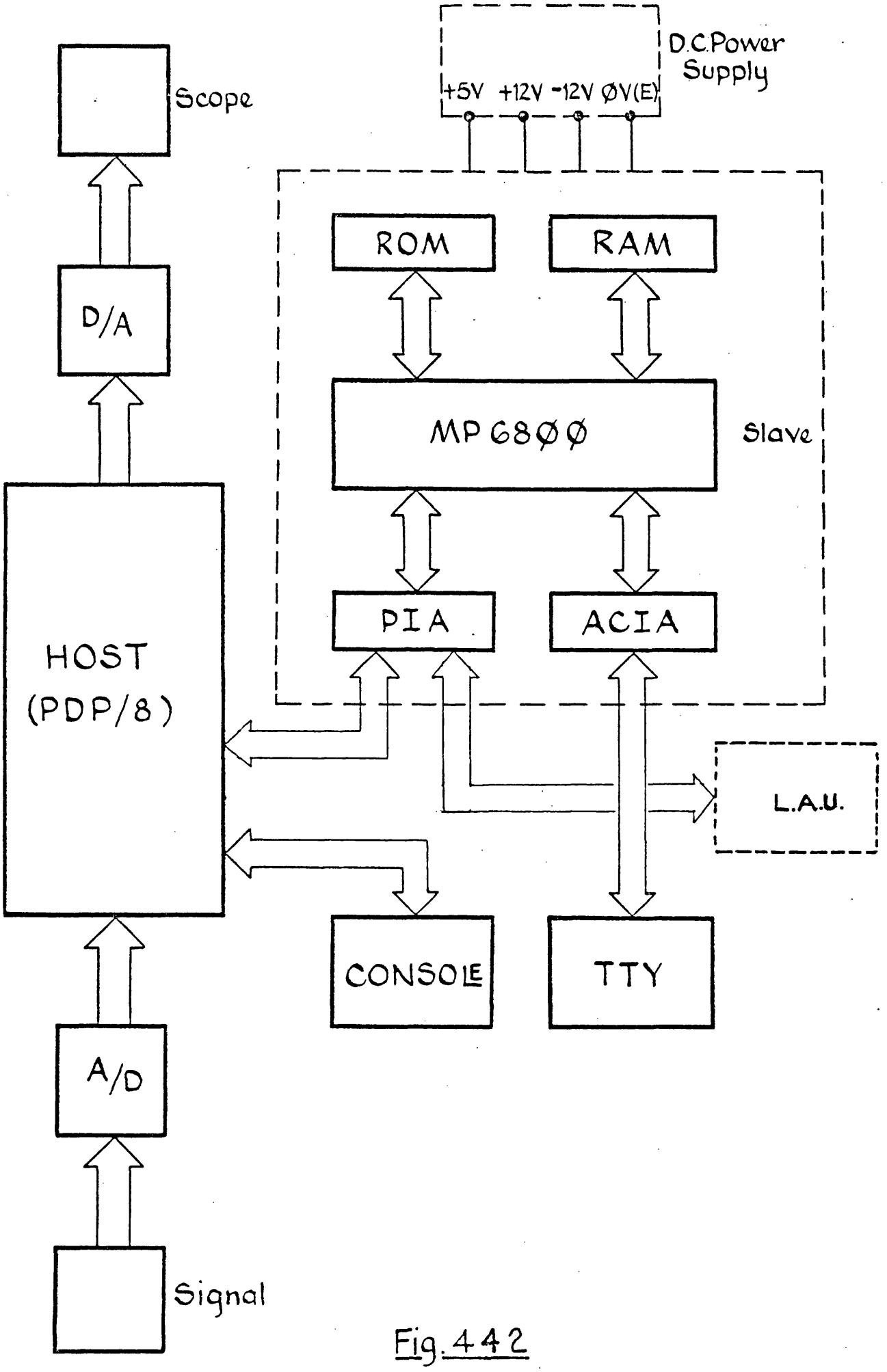

D.C.Power Supply
+5V +12V -12V ØV(E)
Scope
ROM
RAM
D/A
MP68ØØ
Slave
HOST (PDP/8)
PIA
ACIA
L.A.U.
CONSOLE
TTY
A/D
Signal

Fig. 442

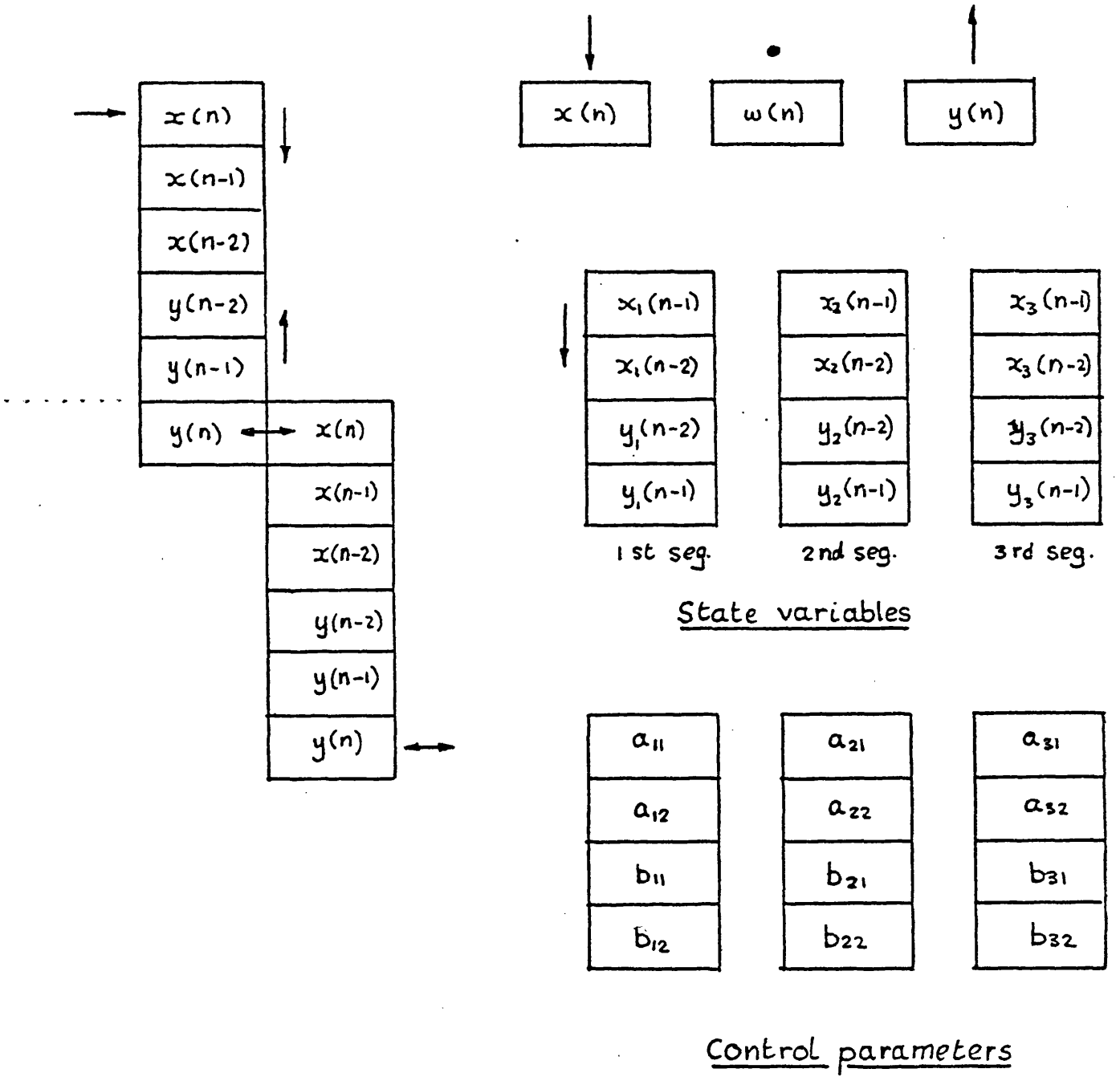


Fig. 451. Access and shuffling procedure

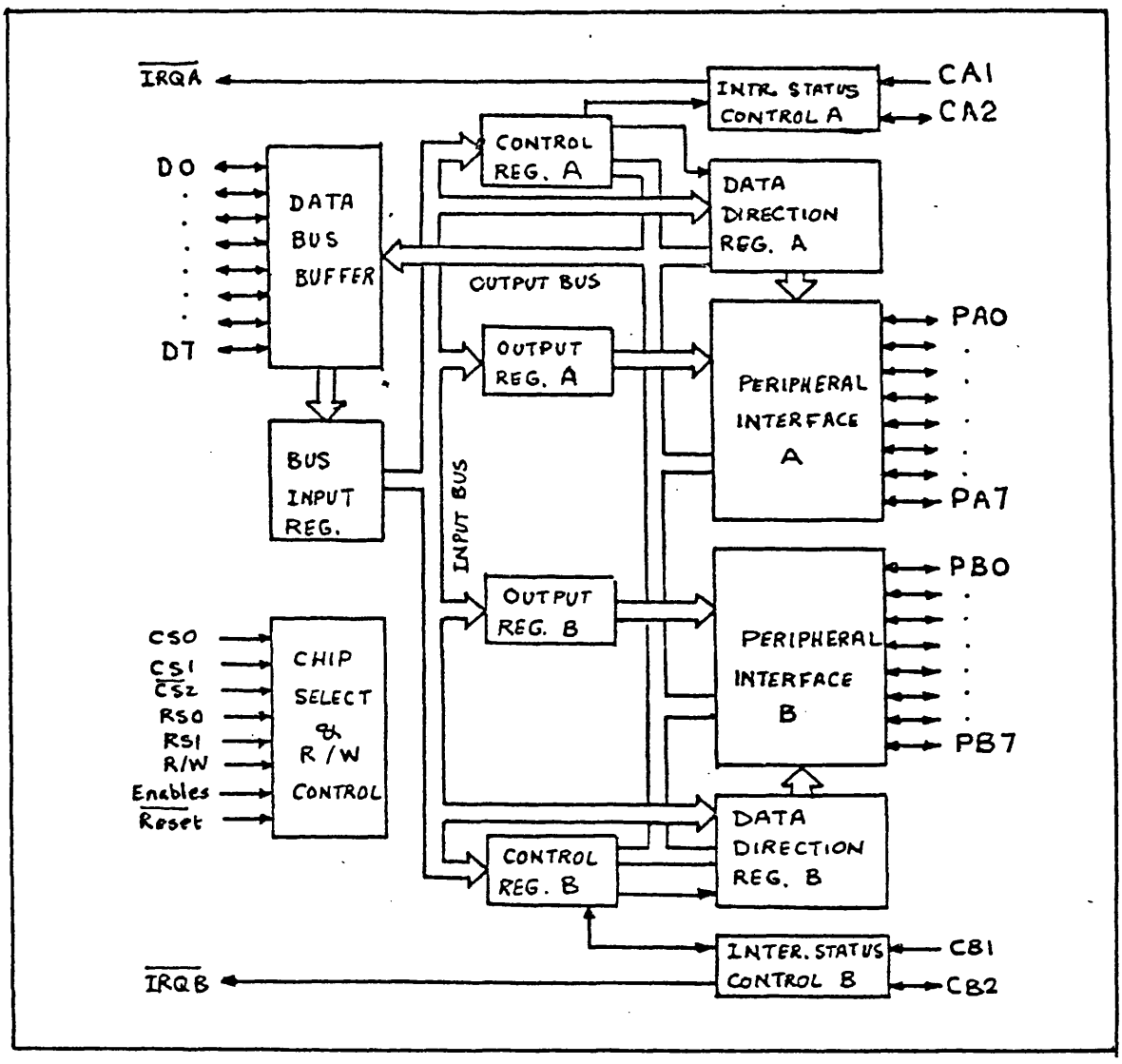


Fig. 452. PIA Register Structure

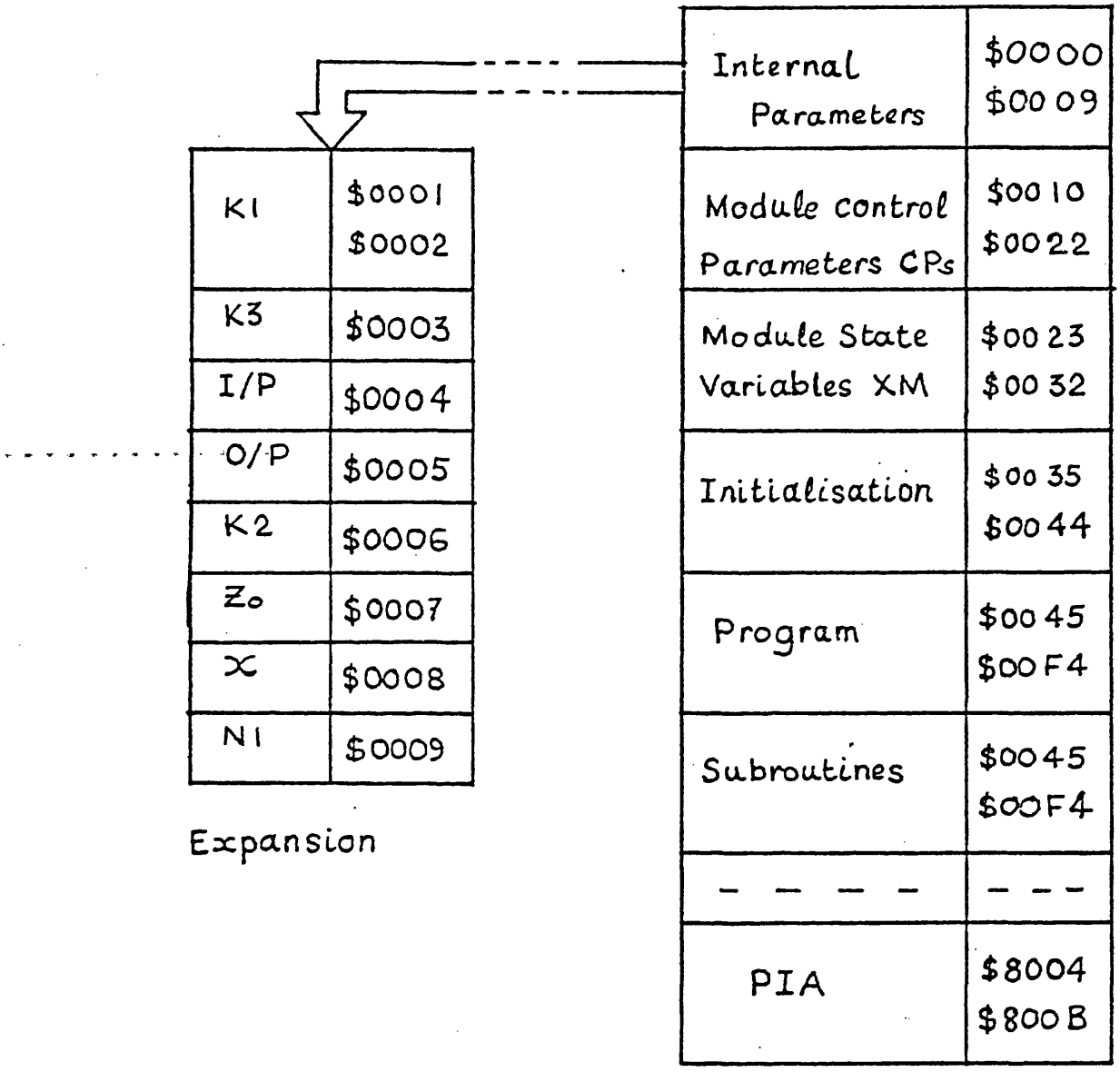


Fig. 453. Memory Map

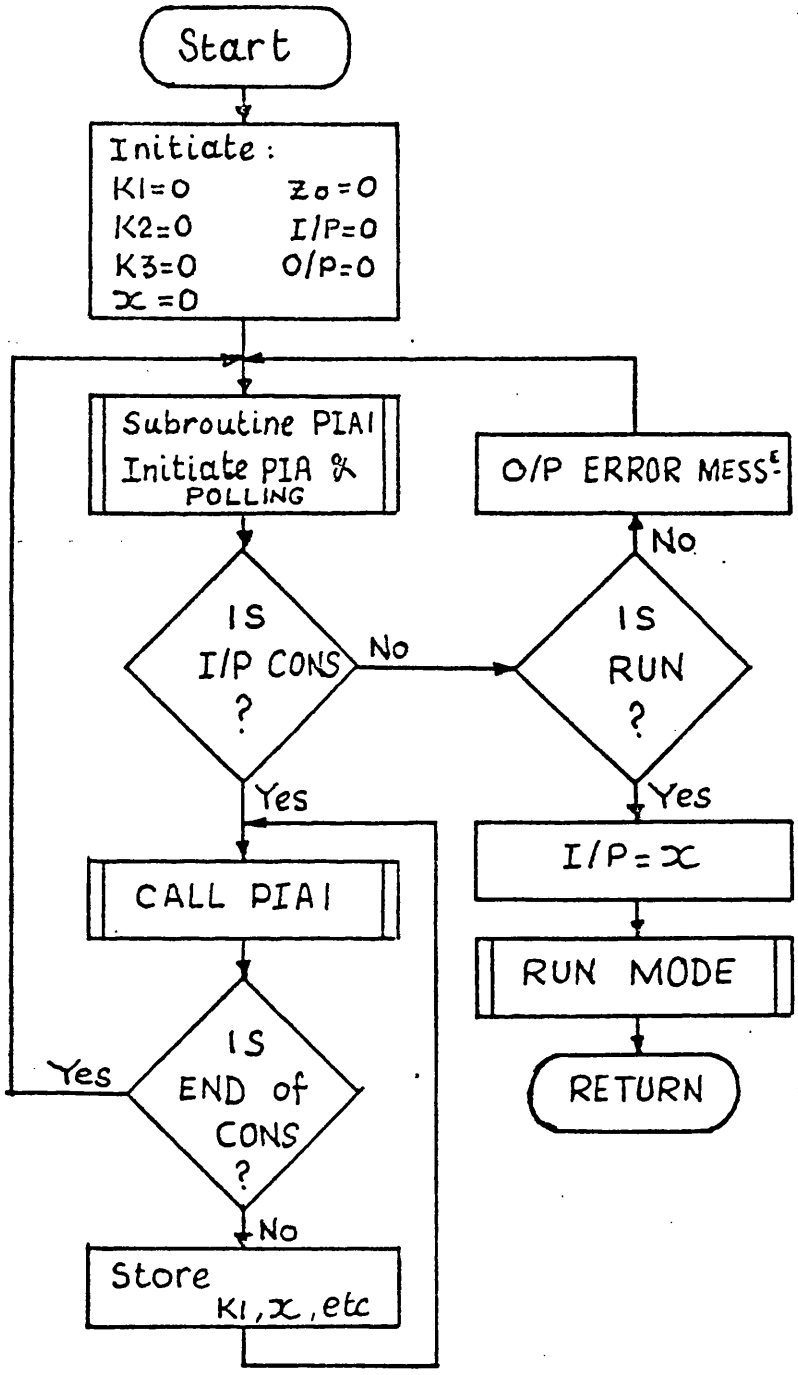


Fig. 454 : CONS Mode

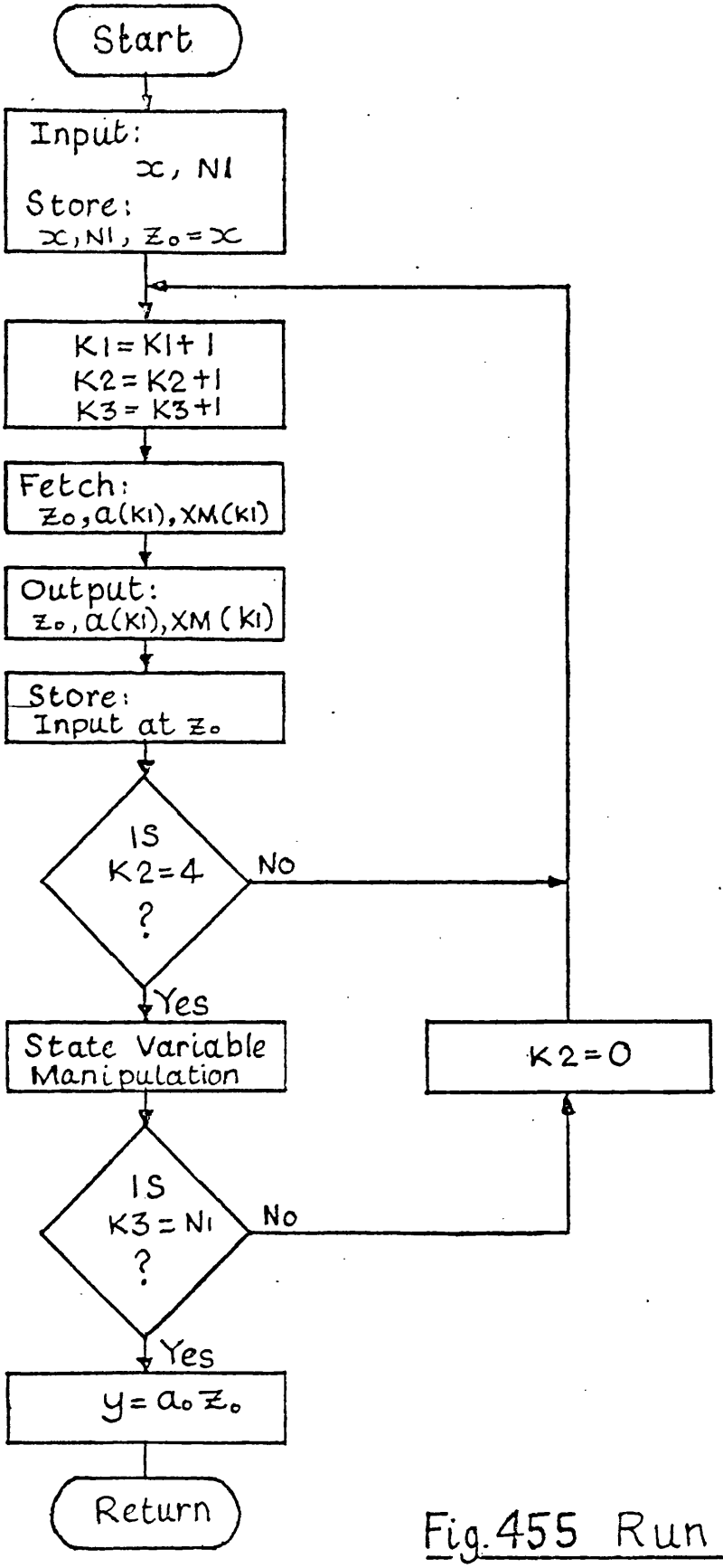


Fig. 455 Run Mode

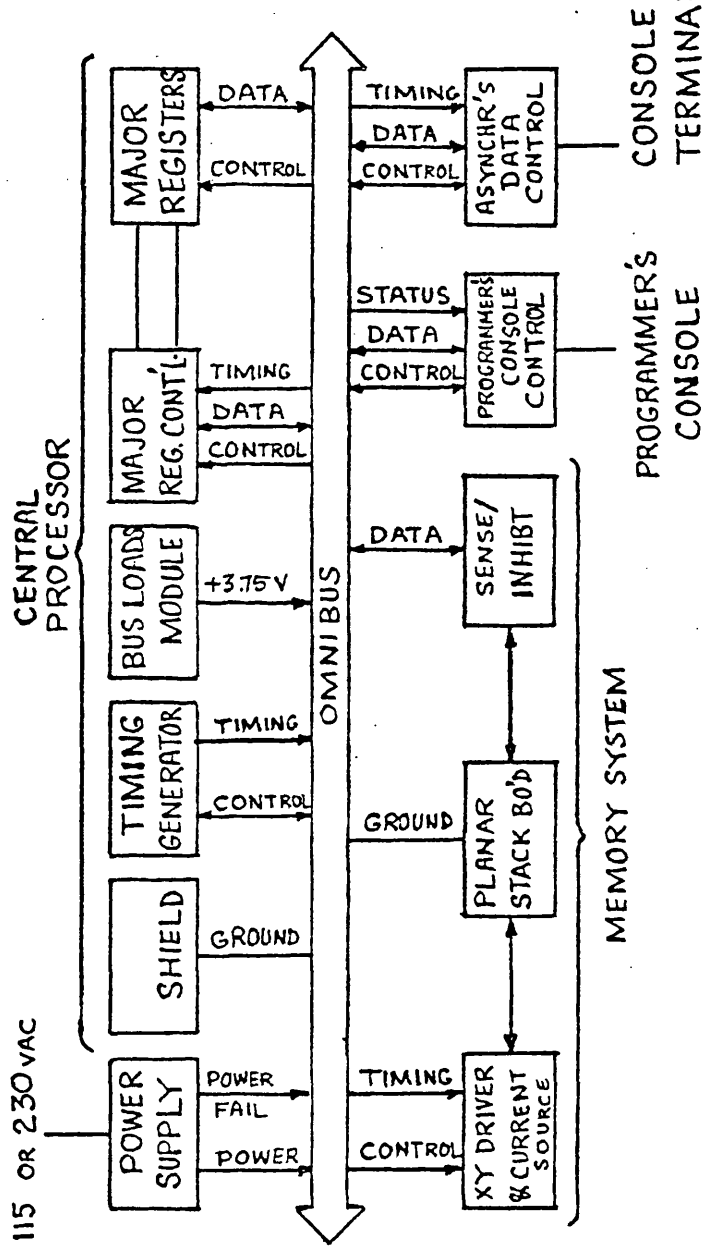


Fig. 461: PDP-8/E Basic System Block Diagram

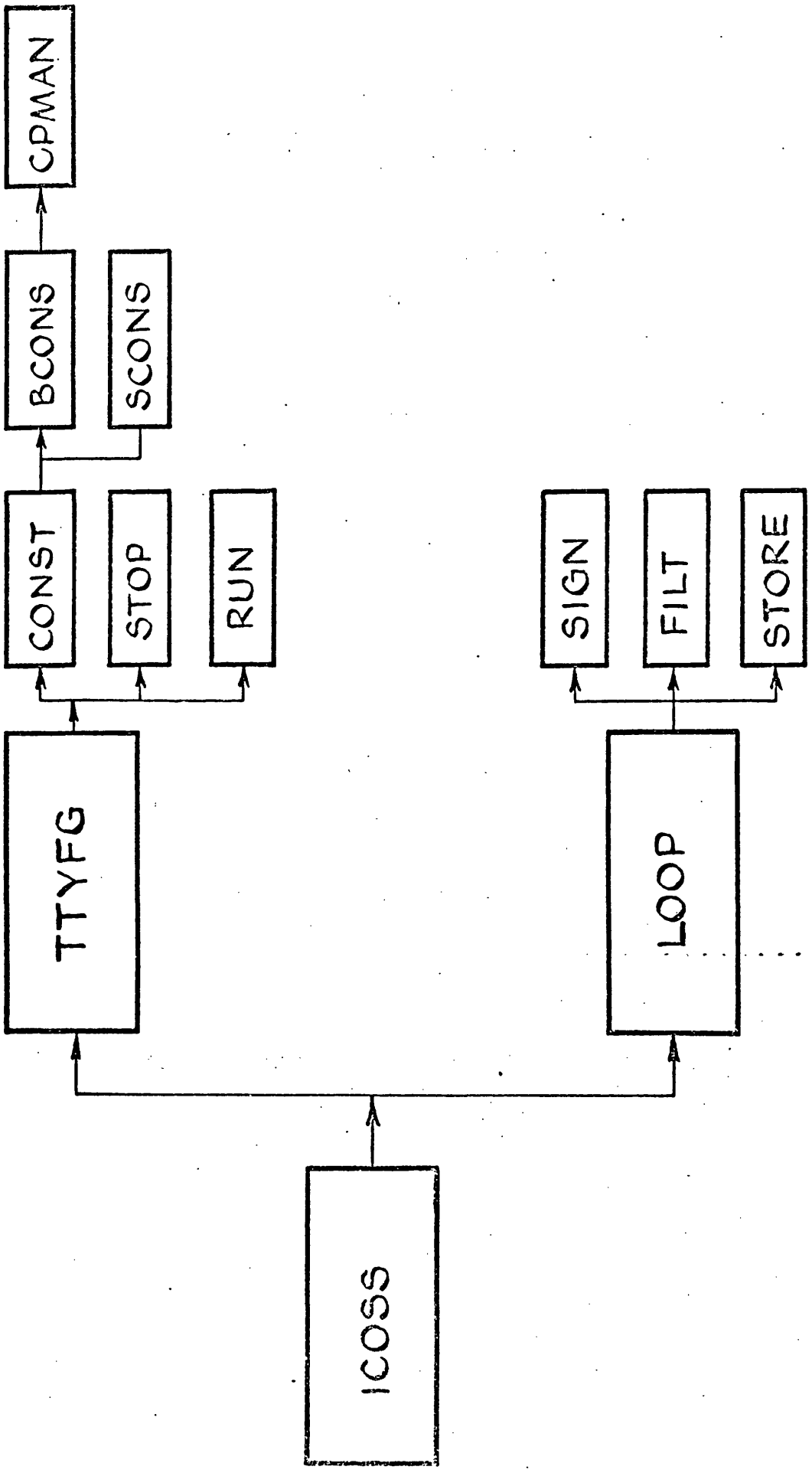


Fig. 4.6.2 Program Structure

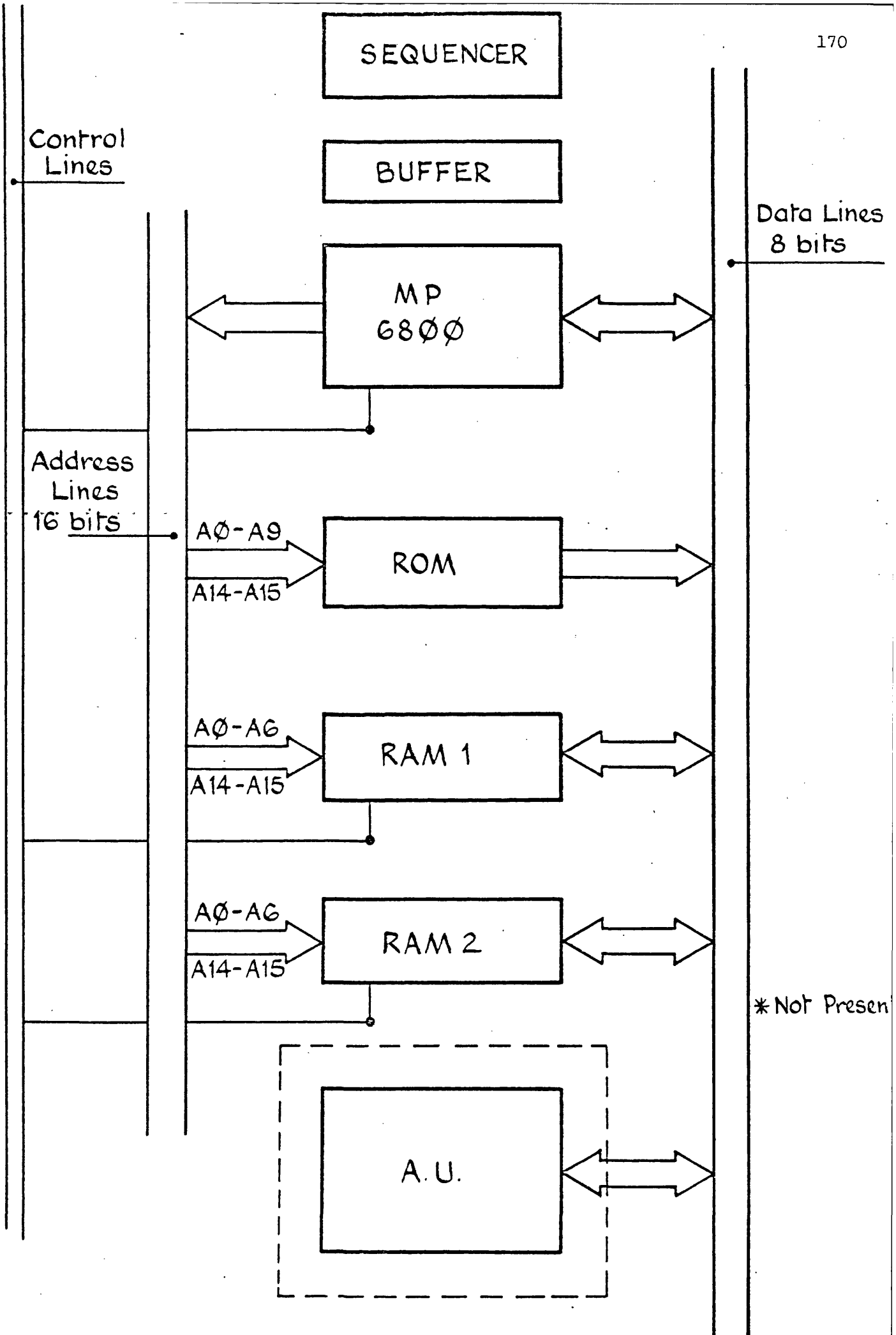

SEQUENCER
BUFFER
Control Lines
Data Lines 8 bits
MP 68ØØ
Address Lines 16 bits
AØ-A9
A14-A15
ROM
AØ-A6
A14-A15
RAM 1
AØ-A6
A14-A15
RAM 2
* Not Presen
A.U.

Fig.471

CHAPTER 5

APPLICATION PROBLEMS

5.1 INTRODUCTION

Discussion in developing the new communication system simulator (ICOSS) so far, has covered ICOSS structure and function, and the future feasibility and developments, (Chapters 3 and 4 respectively). The third and final aspect of developing a simulator is its novelty in engineering and research work, to be discussed in this Chapter.

Two investigation problems were chosen to test ICOSS behaviour in a communication engineering application; they are

(a) Interferences in phase-locked loop
(b) Fast acquisition phase-lock loop (PLL)

The reasons for choosing those application problems are:

(i) Both contain feedback loops, which ICOSS supposedly is capable of handling.
(ii) The performance of both systems is difficult to assess theoretically.

(iii) Both problems have been investigated experimentally. The first by A Blanchard[7] and the second by J P McGeehan[36] at Bath University. The comparison of results will make a good indication of ICOSS usefulness.

As a complementary introduction to those problems, the analysis, characteristics and behaviour of a simple PLL is made, as well as outlining the capability of ICOSS approach in determining these parameters compared with other conventional methods.

5.2 SIMPLE PLL

5.2.1 Introduction

A simple second order, analogue phase-lock loop (PLL), Fig 5.2.1a, is to be designed and simulated, and whose basic theory and analysis is found in Appendix H. Two approaches are made in the simulation of this PLL using ICOSS, namely:

(a) The direct approach: in which the actual signal as function of time is used in the analysis.
(b) The complex signal approach: in which the real-imaginary components of the signal are considered in the analysis.

The direct apprach is also attempted using a dedicated

simulation package for simple PLL. The comparison between the three approaches is based on:

(i) number of computer runs needed to achieve result.

(ii) User ease.

5.2.2 PLL parameters

With reference to the PLL theory as outlined in Appendix H, the following are the parameters of the PLL used in the tests:

* Phase sensitive detector PHSD parameter:

  $K_d = 0.274$

* Voltage control oscillator VCO:
  + parameter: $K_o = 0.029$ Hz/volt
  + free running frequency: $f_o = 10$ Hz
* Gain amplifier (GAIN):

  A = 1200, 1500, 2000

* Filter:

  Cut-off frequency f_c = 0.1, 0.2, 0.3 Hz

* Calculated d.c. gain of PL:

  K: 59.9, 74.88, 99.87

5.2.3. Direct Approach

(a) PLL special purpose package SPP

The tests carried out using a specially written program for a phase-lock loop, were to determine the lock ranges of the PLL whose parameters outlined in 5.2.2, for the conditions mentioned above. For each set of conditions two tests were carried out, one for determining the upper limit of the locking range, ie F4, Fig H3, and the other for the lower limit, F1. In either case the input frequency was increased in steps of 1 Hz, and the VCO frequency was tested for lock, until the PLL becomes out of lock. In all a total of 18 runs were needed. The results obtained, as compared with the expected values are:

A	f_c (Hz)	L measured (Hz)	L calculated (Hz)
1200	0.1	19	12
"	0.2	21	12
"	0.3	18	12
1500	0.1	24	23.84
"	0.2	23	23.84
"	0.3	22	23.84
2000	0.1	31	31.8
"	0.2	28	31.8
"	0.3	27	31.8

The discrepancy may be due to the number of reasons, see later, section 5.2.5.

(b) ICOSS simulation

(i) Block diagram and ICOSS dialogue are shown in Fig 5.2.1 a,b.

(ii) The locking range of the PLL was determined for the following conditions and sets:

* Input signal frequency (f_c) was increased from 0 → 40 Hz at 2 Hz steps.
* Cut-off frequency of the simple first-order low-pass filter and the gain were varied producing 9 sets of curves.

A = 1200, 1500, 2000

f_c = 0.1 , 0.2 , 0.3 Hz

as before.

(iii) Procedure of (ii) was repeated with input signal frequency decreasing from 40 → 0 Hz at 2 Hz steps, producing 9 curves.

(iv) Method employed:

* Select a fixed input frequency f_i.
* Drive PLL until lock is achieved, if f_i within the range of PLL.

* Measure mean locking frequency and store value in YAXIS(I). This is simply done by using the EDIT interrupt of TTYFG group, and the MEAN interrupt of the FLAG group.
* Increase f_i using the CHCP (change of parameter) interrupt of the TTYFG group, and repeat.
* Plot the curves using the PLTG interrupt of the TTYFG group.
* One run only was enough for the above test, including the graph plotting curves.
* The curves obtained are shown in Fig 5.2.3.

5.2.4 Complex-signal approach

(a) Theory

(i) Basic Operation Analysis (Fig 5.2.4a)

the input signal: $x(t) = m(t) \cos(\omega_o t+\phi(t))$

ie general modulated signal

$$y(t) = \cos((\omega_o+\delta\omega)t)$$

ie frequency offset, constant amplitude signal

thence: $$z(t) = m(t) \cos(\omega_o t+\phi(t)) \cos((\omega_o+\delta\omega)t)$$

$$= m(t) \frac{1}{2} \{\cos\left[(2\omega_o+\delta\omega)t + \phi(t)\right] + \cos\left[-\delta\omega t+\phi(t)\right]\}$$

$$\omega(t) = \frac{m(t)}{2} \cos\left[\phi(t)-\delta\omega t\right] :$$

assuming ideal filter

$$\omega'(t) = \alpha(t) \cos \theta(t) \quad \text{in general}$$

assuming non-ideal filter.

(ii) The model:

1. Phase sensitive detector PSD (Fig 5.2.4b)

$x(t)$, $y(t)$ are r.f. modulated signals

$\omega'(t)$ is a real, base-band signal

$z'(t)$ is $z(t)$ filtered by ideal Lpf so as to exclude the $2\omega_o$ component

or $$z'(t) = \frac{m(t)}{2} \cos\left[\phi(t)-\delta\omega t\right]$$

$$\equiv m(t).\{\exp j\left[\phi(t)-\delta\omega t\right] + \exp -j\left[\phi(t)-\delta\omega t\right]$$

Now complex models of $x(t)$ and $y(t)$ are:

$$x_c(t) = m(t) \exp j\,\phi(t)$$

$$y_c(t) = 1 \exp j\,\delta\omega t$$

thence $$z'_c(t) = x_c(t) \cdot y^*_c(t)$$

$$\omega'_c(t) = z'_c(t) * h(t) \quad \text{in usual way,}$$

where $h(t)$ is the filter impulse response

2. Voltage controlled oscillator VCO

$$\delta\omega \propto |\omega'_c(t)|$$

(iii) Simulation elements

1. Multiplier: (Fig 5.2.4c), which includes ideal Lpf to exclude $2\omega_o$ components

$$z' = x.y^*$$

Taking real and imaginary components:

$$z'_R = x_R y_R + x_I y_I$$

$$z'_I = x_I y_R - x_R y_I$$

but z'_I is not important since z'(t) is a real baseband signal.

2. Filter

$$\omega'_R = z'_{R*} \; h(t)$$

$$\omega'_I = z'_{I*} \; h(t)$$

but ω'_I is not relevant.

3. VCO

$$\delta\omega = k\omega'_R$$

$$y_R = \cos \delta\omega t$$

$$y_I = \sin \delta\omega t$$

(iv) Loop simulation

The final block diagram representation for this simple phase loop is shown in Fig 5.2.4d.

(b) The block diagram as used in ICOSS and its equivalent listing are shown in Fig 5.2.5.

(c) Measurements and tests

The effect of varying sampling frequency on acquisition time of PLL under test.

(d) Method and results

* The transient responses of the locking frequency of PLL are obtained using two interrupts: TRAN of the FLAG group and TRAN of the STOP interrupt of the TTYFG group, as shown in Fig 5.2.5, for number of sampling frequencies, (f_s). The f_s variation was carried out using the change of global parameter (CHGC) interrupt of the TTYFG.
* The values of f_s were chosen to be multiples of the input signal frequency which was chosen to be 8 Hz.
* Numerical values were obtained this time for the response and a closer look to the actual values was made.
* The acquisition times (measured over 1% of the locking frequency) were determined as follows:

Sampling frequency (f_s) Hz	8	16	24	48
Time in sec	No lock	1.75	1.83	1.88

5.2.5 Comment

(a) Locking range of the PLL under test: the reason for the discrepancy between the calculated and the measured locking range of the PLL is due to the fact that the locking range is not symmetrical about the free-running frequency of the VCO, as clearly shown in Fig 5.2.3. This means that the simple equation of Appendix H for lock range does not hold; however ICOSS is capable of analysing completely this type of problem.

(b) The effect of varying f_s:

* Increasing sampling frequency will give better definition of the signals at the expense of computing time.
* Complex-signal approach is related to the modulating signal frequency and therefore requires lower sampling frequency (f_s) than the direct approach which is carrier frequency dependent. eg for the previous work, f_s was 96 Hz for the direct approach, and 24 Hz for the complex-signal approach, for the range of frequencies mentioned, which were centred about 10 Hz (the VCO free running frequency).

(c) Plots: with the ability to store sample values and manipulate them, it is possible to plot any response vs any parameter variations. This was demonstrated in Fig 5.2.3 where a number of curves are plotted for different sets of parameters, all on one graph, within one run. This is a feature of ICOSS which provides better understanding of system behaviour.

(d) Runs: there is a marked reduction in number of runs using ICOSS, as compared with the SPP, vis, 18:2, hence reduction in computer time.

5.3 INTERFERENCES IN PHASE-LOCK LOOPS; STOCHASTIC SIMULATION

5.3.1 Stochastic Simulation[49]

Even in apparently deterministic and well-behaved situations, some averaging must take place. There are two ways of treating this type of problem:

(a) Systematic variation of parameters: for example, a problem with two parameters; the first parameter having six values and the second, five, 30 simulation runs are required in order to examine the full effect of those two parameters on the system under test. Yet averaging process has to be carried out to arrive at the final result. In practice, it is possible to reduce drastically the number of runs by examining the sensitivity of the final result to individual parameter. Notice that in a practical test the relative phases of the signals are continually moving, so that an indicating instrument takes automatic averaging.

(b) Monte-Carlo technique[22] : where the parameters are varied by choosing random values, according to certain statistical criteria. That means simulation must record the results of a number of observations, considering the sequence in which they occur and not the values assumed by the variable . This technique

usually requires a large number of estimates to ensure adequate convergence.

5.3.2 The problem of intereferences in PLL

As an application to this type of problem, tone interferences in PLL which may be encountered in PLL receivers, is chosen. The reasons for this choice are:

* Co-channel interference is common occurrence.
* The theoretical analysis is complicated and the simulation approach is ideal for this type of problem.
* The effect of an interfering sinusoidal signal on another PLL situation has already been investigated theoretically, and practically by A Blanchard[7] as mentioned earlier, and a comparison between the two approaches can be made.

There are two cases to consider:

(a) Influence of an unwanted signal on PLL initially in lock: in this case the loop initially is in lock with the wanted signal f_i and then the unwanted signal f_u is applied, and the PLL behaviour is investigated.

(b) Influence of an unwanted signal on PLL initially out of lock: the two signals are simultaneously

applied to the PLL input, and the acquisition behaviour, etc is then investigated.

These two cases were investigated by Blanchard experimentally, and number of graphs are obtained. The attempts are now made to confirm these results using ICOSS and simulation approach.

5.3.3 Problem formulation

(a) General

* Block diagram and its equivalent ICOSS listing are shown in Fig 5.3.1. a,b.
* The PLL used for these tests is the same as the one already used in Section 5.2.
* The simple PLL structure, its parameters, and theoretical formulations are discussed in Appendix H, whereas the theory of the interferences in PLL is fully described in Blanchard paper[7].
* The input signals under tests are of the form:

  wanted signal $y_i = A_i \cos(\omega_i t + B_i)$

  unwanted signal $y_u = A_u \cos(\omega_u t + B_u)$

  For f_i = 8 Hz fixed

  f_u : varied from 5 Hz → 15 Hz

  A_i = 1.0 fixed

  A_u : 0.3, 1.0, 2.0

  giving rise to

  $\frac{A_i}{A_u}$: - 10 dB, 0 dB, + 6 dB.

(b) Measurements

(i) For unwanted signal f_u = 11.0 Hz, and amplitude A_u = 0.1, set of waveforms are to be obtained for the nodes within the block diagram, which will show the variation of signal at each node, for the case PLL initially out of lock (5.3.2b).

(ii) Parameter variations: for the case of PLL initially out of lock, unwanted signal = 10 Hz and amplitude: 0.1, 0.3, 0.6, 0.8, the locking frequency responses and the RMS values variation vs amplitude, for the steady state situations are to be determined.

(iii) Confirmation of the Blanchard formulation, by plotting points on the curves which he formulated theoretically, and approved experimentally. With use of the EDIT interrupt of TTYFG group, the test for (5.3.2a) can easily be carried out, ie running PLL until lock is achieved on the wanted signal channel and then edit the system structure to add the unwanted signal, preserving the system state, and continue with the test.

5.3.4 Results and Comments

(i) Signal Flow

The pictorial representation of the signals (wanted and unwanted) as accessed at the inner nodes of the block diagram for f_u = 11.0 Hz, A_u = 1.0 and f_i = 8.0 Hz, A_i = 1.0, are plotted on one graph, Fig 5.3.2. It can be deduced from the test and the graphs:

* The ease with which many signal responses are obtained (ICOSS feature), within one run.
* The influence of small interfering tone is clearly shown in the behaviour of PLL.
* The FM effect on the VCO.

(ii) Parameter variations

Two graphs are plotted, Fig 5.3.3.

(1) The VCO frequency response
(2) The RMS of the steady state locking frequency ripple vs amplitude variation.

It can be seen from the second graph that a derived variable can be plotted with the same ease as plotting the response of the first graph. This is another feature of ICOSS.

(iii) Blanchards Results

To show the actual variations and actions within PLL for the conditions under tests as mentioned earlier, the locking frequency response of case (a) subsection 5.3.2, for the situation when PLL locking on the unwanted signal, and then the unwanted signal is injected, and the transient response resulted, Fig 5.3.4.

The parameters for Blanchard curves, ie $m = (\frac{\Delta f}{f})$ and $(\frac{\Omega}{2\zeta\omega_n})$ are determined from these graphs and then plotted on those curves which Blanchard formulated theoretically and approved experimentally. These curves, Fig 5.3.5, have been approved using this method. However, more runs need to be made, in order to confirm all the results Blanchard formulated.

5.4 FAST ACQUISITION PLL

5.4.1 Description

The considered technique has been proved to give both improved acquisition and tracking performance for second order PLL used in narrow-band communication systems. The new technique is such that:

* it could be easily applied to all second-order phase-lock loops.
* it would not be necessary to redesign the loop filter (active or passive).
* the loop gain must remain unaltered at the design value.
* on attaining phase-lock the loop must revert to its conventional form.

The technique is based upon a consideration of the non-linear loop equation describing the pull-in behaviour of a second-order phase-locked loop shown in Fig 5.4.1:[36]

$$\ddot{\theta}_e(t) + 2\eta\,\omega_n \cos\theta_e(t)\,\dot{\theta}_e(t) + \omega_n^2 \sin\theta_e(t) = 0$$

where $\sin\theta_e$ is the error signal from p.s.d.1,

η is the loop damping factor

and ω_n is the natural frequency of the loop.

By expressing the solutions to this equation in the form

of phase-plane plots[36] a relationship can be established between the instantaneous frequency error, $\dot{\theta}_e$, and cos θ_e, a signal which is readily available in most second-order loops. If the error signal from p.s.d.1 is differentiated by a simple RC network the signal from p.s.d.2 can be used to pass only those half-cycles driving the VCO towards synchronisation. A schematic diagram of this techniques is shown in Fig 5.4.2. It will be observed that the cos θ_e electronic switch, S1, is used to pass the appropriate half-periods into a suitable summing amplifier configuration. By adjusting the gain of the differential path the smoothed voltage leaving the loop low pass filter is such that the input and VCO frequencies are automatically synchronised. At this point the differential error signal leaving the RC network is small and the error signal from p.s.d.1 (which is now large) drives the loop into phase synchronisation. Once in lock the cos θ_e switch, S1, remains open and the loop reverts to its normal form.

5.4.2. Parameters

With reference to the circuit diagram shown in Fig 5.4.1:

(a) The actual hardware parameters:

These are the actual values used in the original circuit:

* VCO $K_o = 5.75 \times 10^5$ radians/sec/volt

f_o = 200 KHz

* PHSD K_d = 0.065 volts/radian

* Filter R_1 = 430 Ω

R_2 = 1500 Ω

C_1 = 0.47 μF

* Differentiator

C_2 = 0.1 μF

R_3 = 20 KΩ

* PL Loop z = 0.707

$\omega_n = 2\pi \times 10^3$ rad/sec

(b) Parameter used in ICOSS model

With the use of the formulation deduced in Appendix H, the parameters for the PLL model as formulated by ICOSS, Fig 5.4.2 are:

* VCO K_o = 91.5141 KHz/volt

f_o = 200 KHz

* PHSD K_d = 1.0

* LPass filter f_c = 1.414 KHz

This is the same first order Butterworth type low pass filter already used earlier

* GAIN A = 0.1189

inter d.c. gain (loss)

5.4.3 Method and Measurements

(a) The method adopted is the complex signal technique, producing the block diagram shown in Fig 5.4.3, with the ICOSS listing in Fig 5.4.4. The differentiator is a 2 sections type, for the same performance, Appendix B.

(b) The measurements were:

(i) Comparison of acquisition time with and without the technique.

(ii) The effect of varying the technique phase shift for 0°, 60° and 90° on the performance of the loop.

5.4.4 Observations

(a) 90° phase-shift case (Fig 5.4.5)

Observation

* Input on VCO centre frequency (200 kHz): new technique produces no difference in acquisition time.
* Input off centre frequency: lock to one side.
* Capture range ± 5 Hz, so:
  - at 194 kHz - no lock
  - at 195 kHz - just locks

Comment

With the limited test carried out, the PLL transient responses seem to tie-up with the experimental results for the cases with/without the technique[53]. The Fig 5.4.6 shows that the fast acquisition technique clearly improves acquisition time.

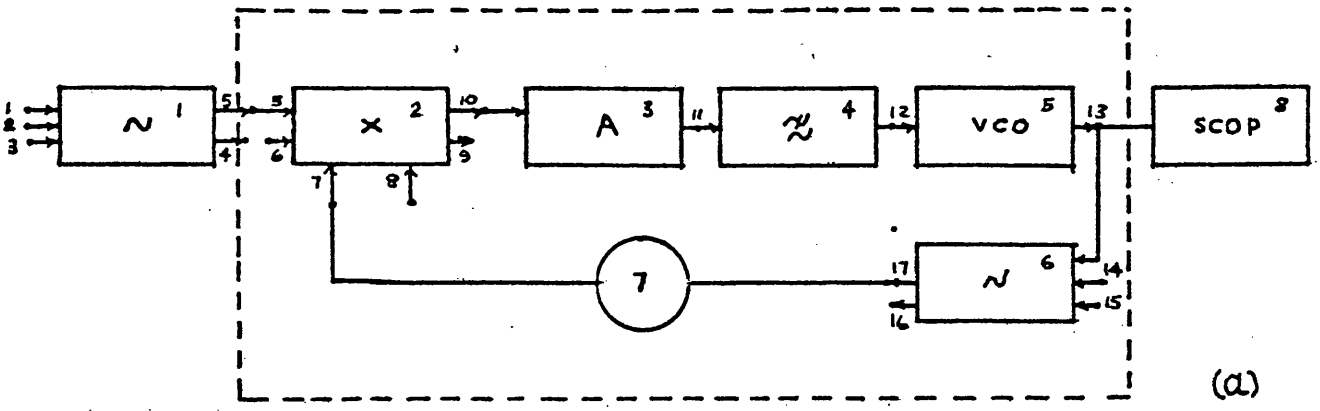
(b) Varying phase shift (Fig 5.4.6)

Observation

* At the edge of normal capture range - very little difference from above.
* Outside the capture range, ie at 194,, 193 KhZ, the new technique appears to extend the capture range.

Comment

Varying the phase shift, seems initially to have some effect on the capture range, probably due to cycle slipping. This result was not anticipated from previous experimental studies and requires more thorough investigation[53].



(a)

```
ENTR
100.0                    fs
SIGN                     ①
5.0
1.0
0.0
-1.0
PHSD                     ②
0.5
GAIN                     ③
1500.0
FILT                     ④
0.1
VCO                      ⑤
10.0
0.029
SIGN                     ⑥
0.0
1.0
0.0
-1.0
BRCH                     ⑦
1
17
7
SCOP                     ⑧
TRAN
Y
500  13
END
1   0.0   1
2   0.0   1
3   0.0   1
14  0.0   1
15  0.0   0
RUN
```

(b)

Fig. 521 PLL BLOCK DIAGRAM & ICOSS LISTING

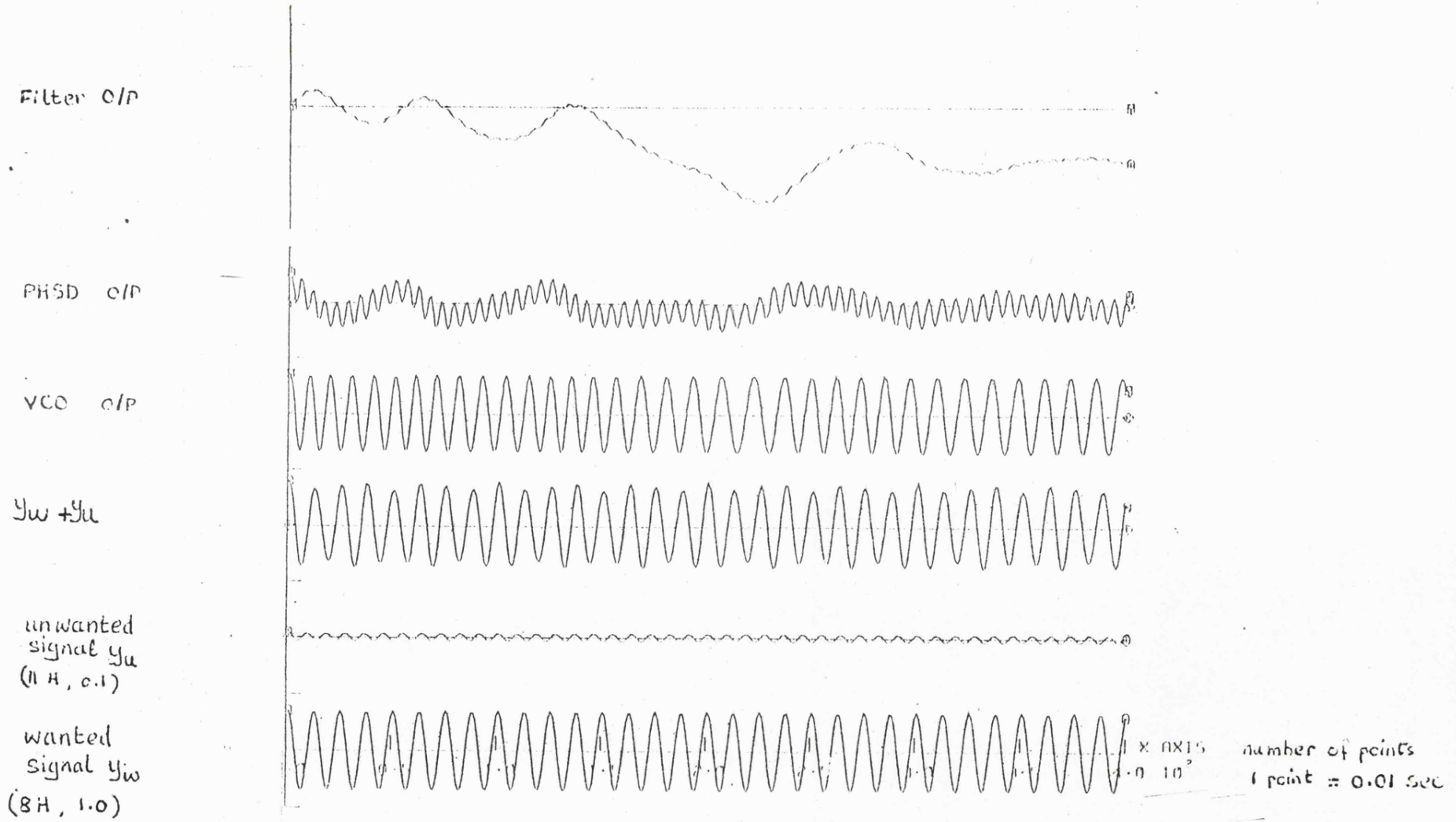


Fig. 532. Interference in PLL - nodes responses

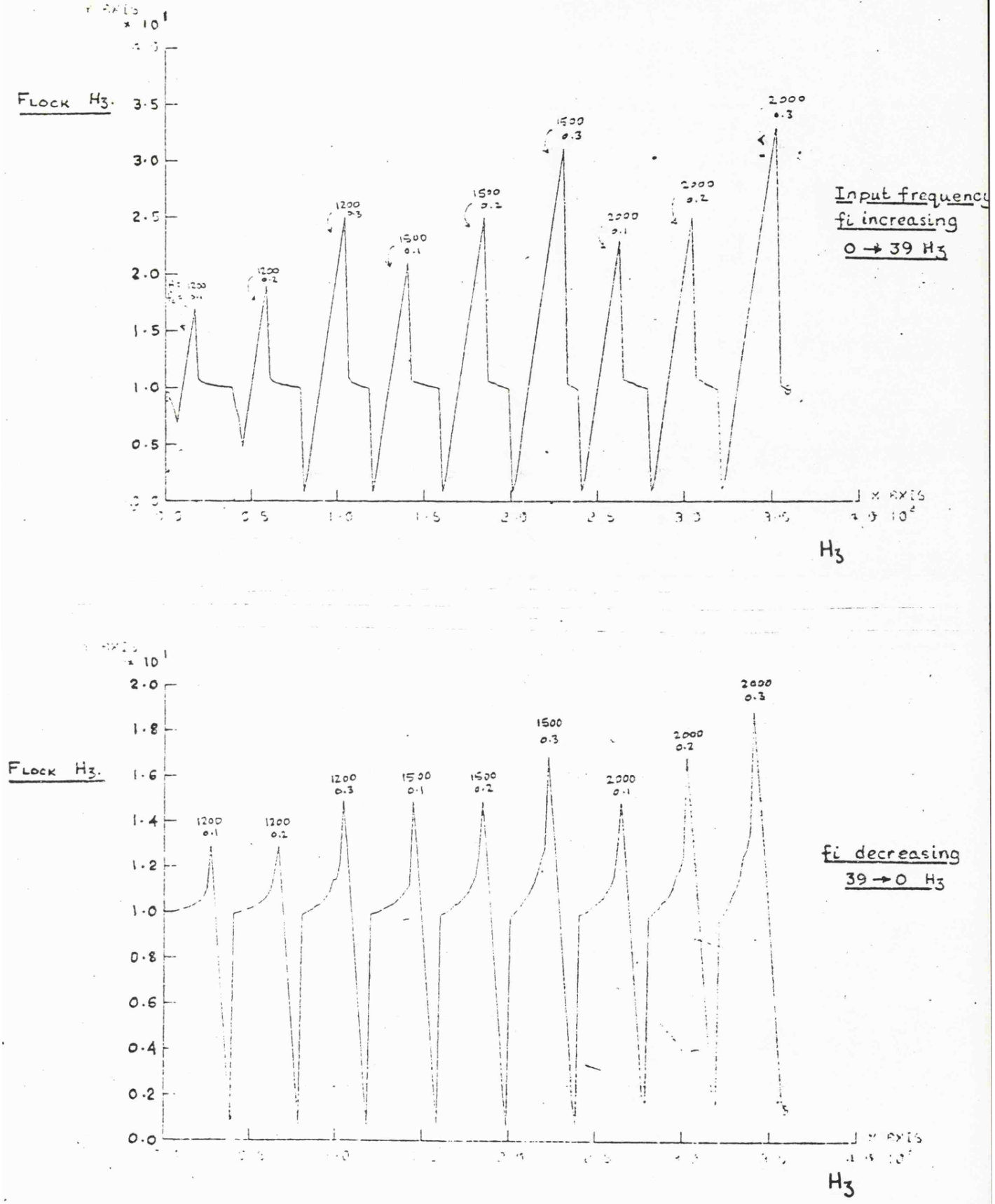
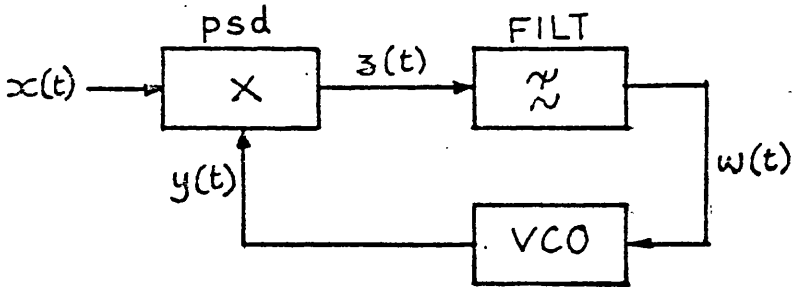
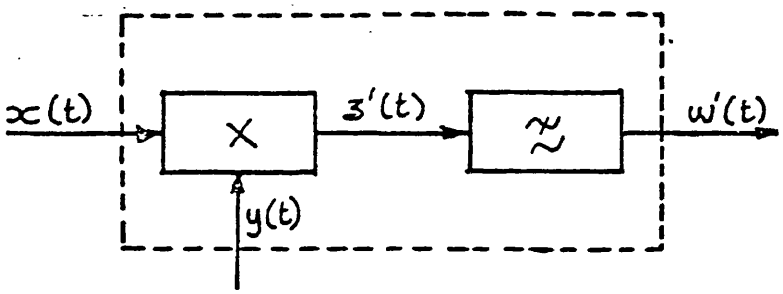


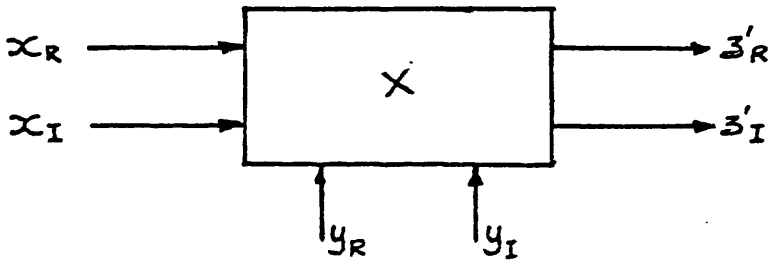
Fig. 523. Simple PLL . Locking range response



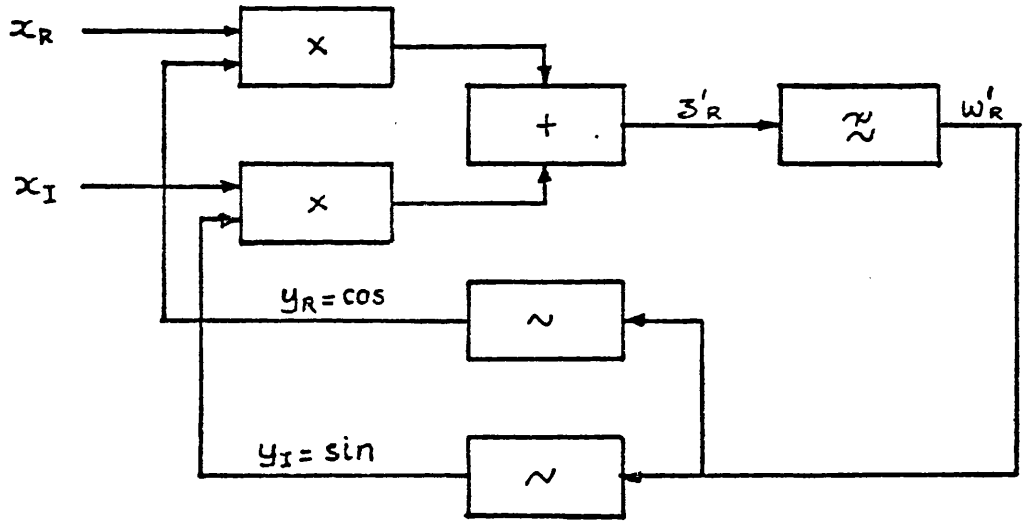
(a)



(b)

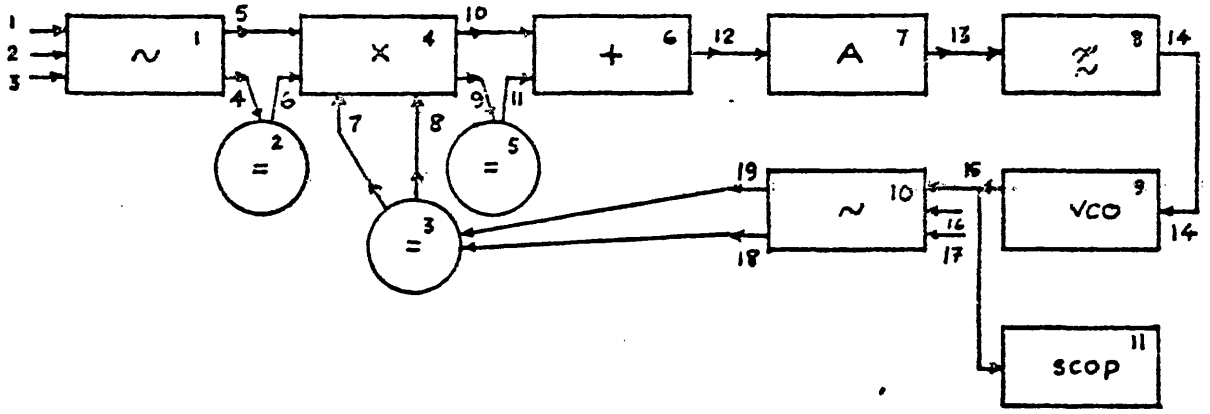


(c)



(d)

Fig 524 : Complex signal approach



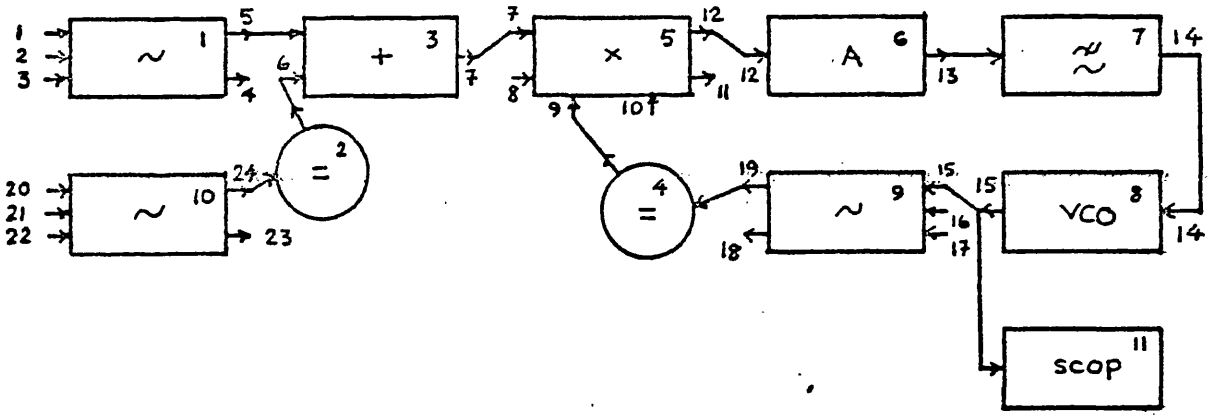
Block diagram

```
ENTR
24.0
SIGN
8.0
1.0
0.0
BRCH
1
4
6
BRCH
2
19   18
 7    8
PHSD
0.5
BRCH
1
9
11
ADDR
GAIN
1500.0
FILT
0.1
VCO
10.0
0.029
SIGN
0.0
1.0
0.0
0.0
SCOP
TRAN
Y
500  15
END
```

ICOSS LISTING

```
1  0.00.0  1
2     0.0  1
3     0.0  1
16    0.0  1
17    0.0  0
RUN
```

Fig.525 Simple PLL - complex signal approach



Block diagram

```
ENTR
100.0
SIGN
8.0
1.0
0.0
0.0
BRCH
1
24
 6
ADDR
BRCH
1
19
9
PHSD
0.5
GAIN
1500.0
FIL1
0.1
VCO
10.0
0.029
SIGN
0.0
 1.0
0.0
0.0
NSTR
SIGN
11.0
 0.1
 0.0
 0.0
```

```
SCOP
TRAN
500
500   15
END
1     0.0   1
2     0.0   1
3     0.0   1
8     0.0   1
10    0.0   1
16    0.0   1
17    0.0   1
20    0.0   1
21    0.0   1
20    0.0   0
RUN
```

ICOSS listing

Fig.531. Interference in PLL_
ICOSS block diagram and listing

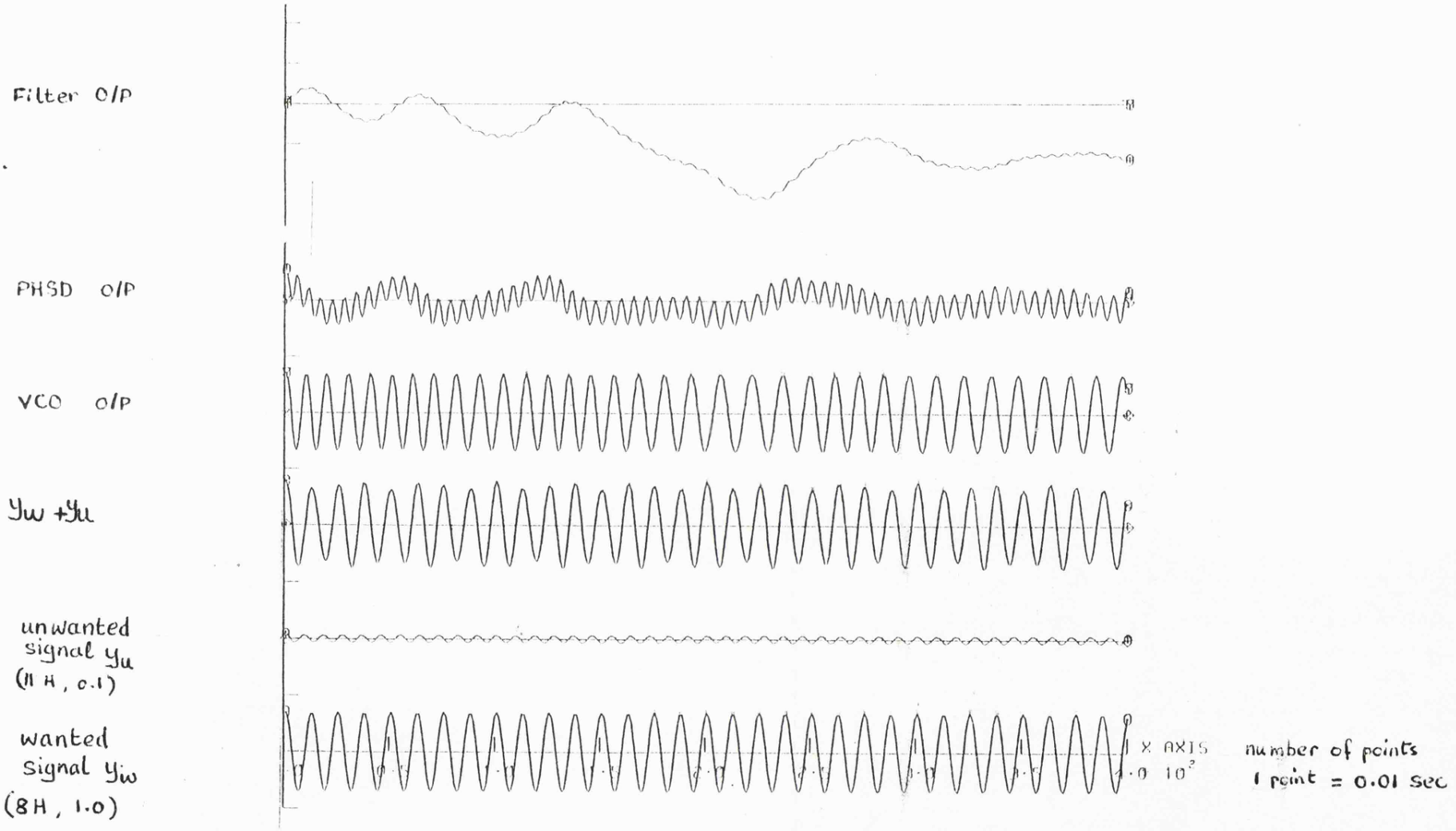
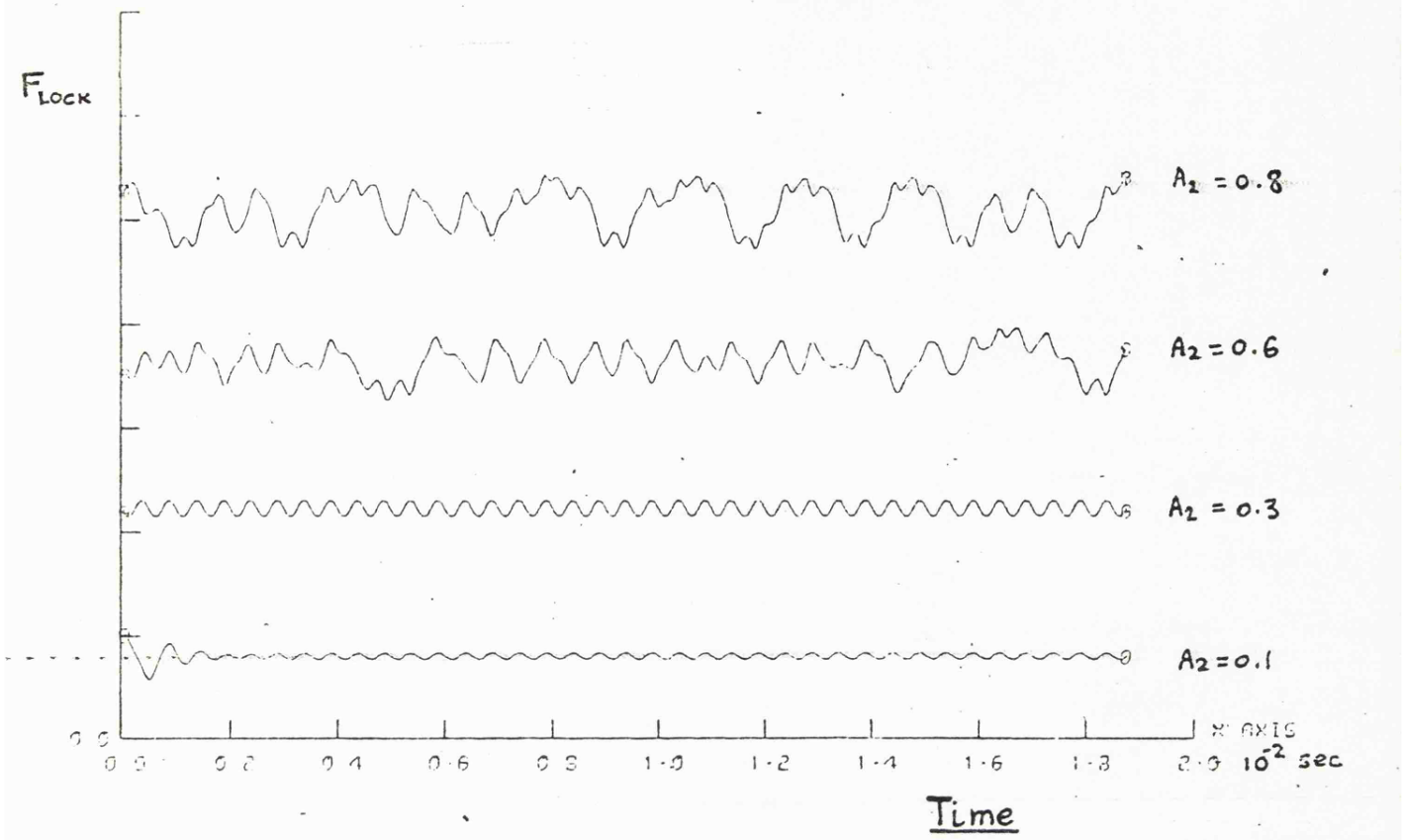


Fig. 532. Interference in PLL - nodes responses



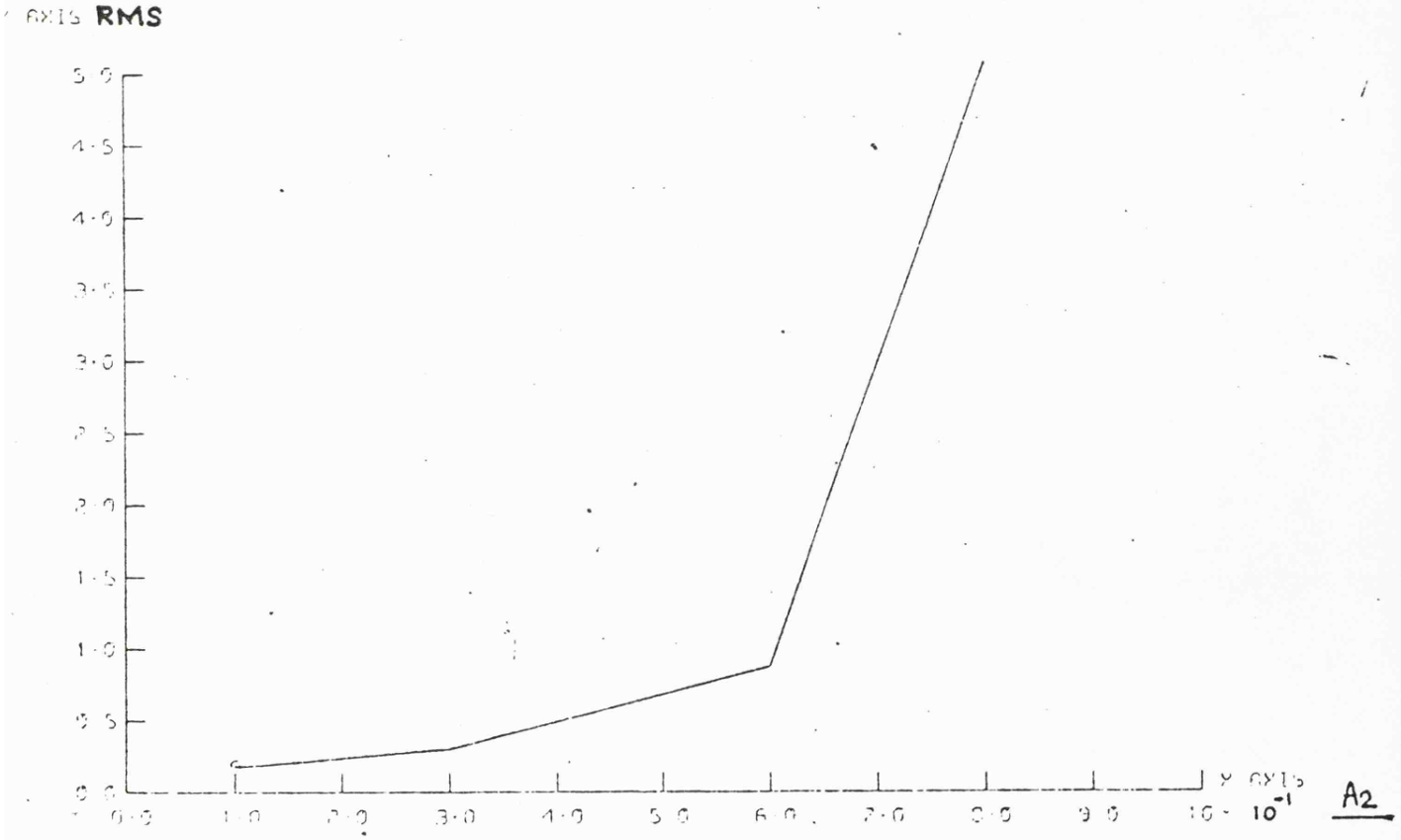


Fig. 533 Interference in PLL

(i) F_{LOC} vs. Time (ii) RMS vs. A_2

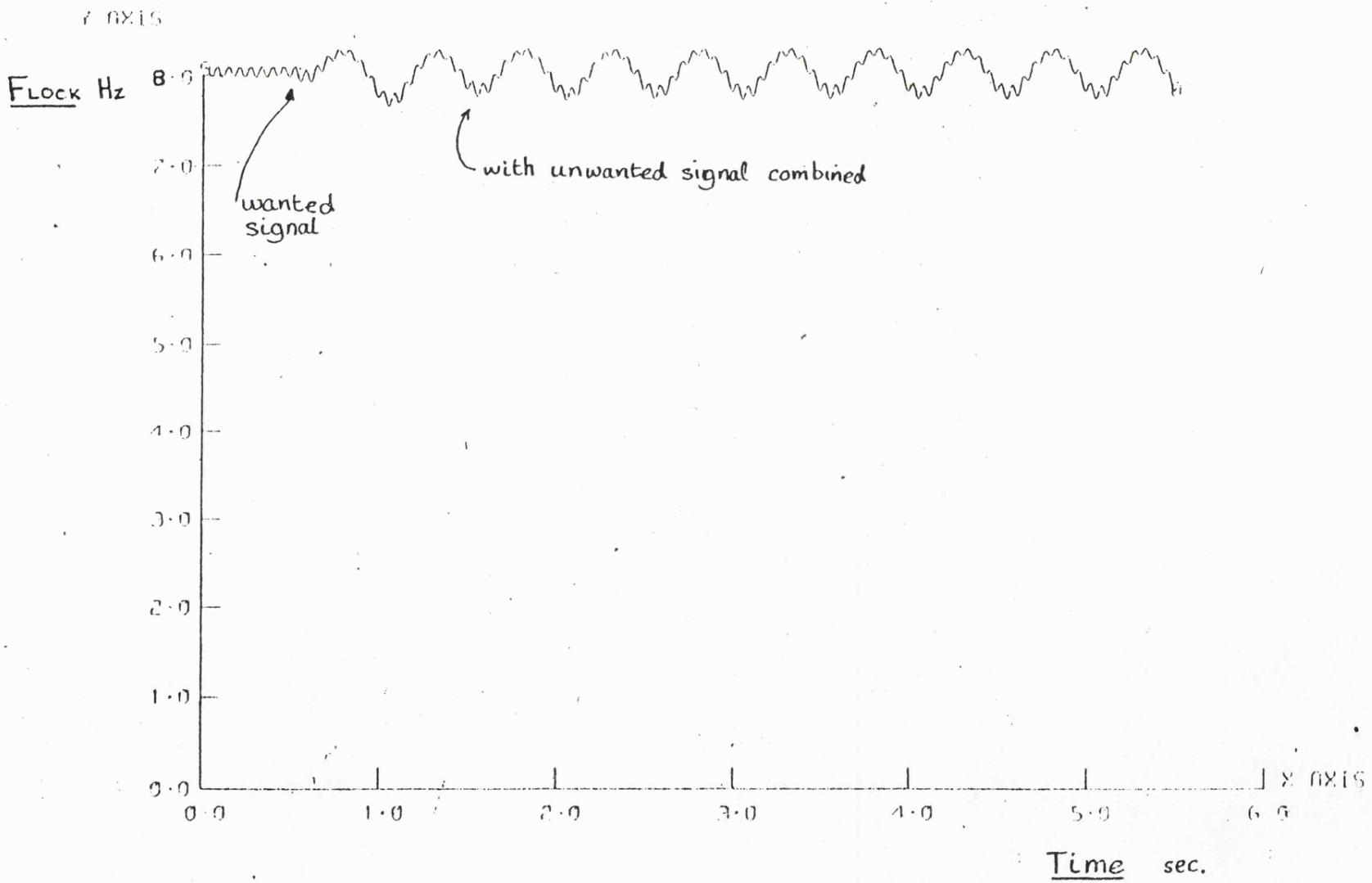
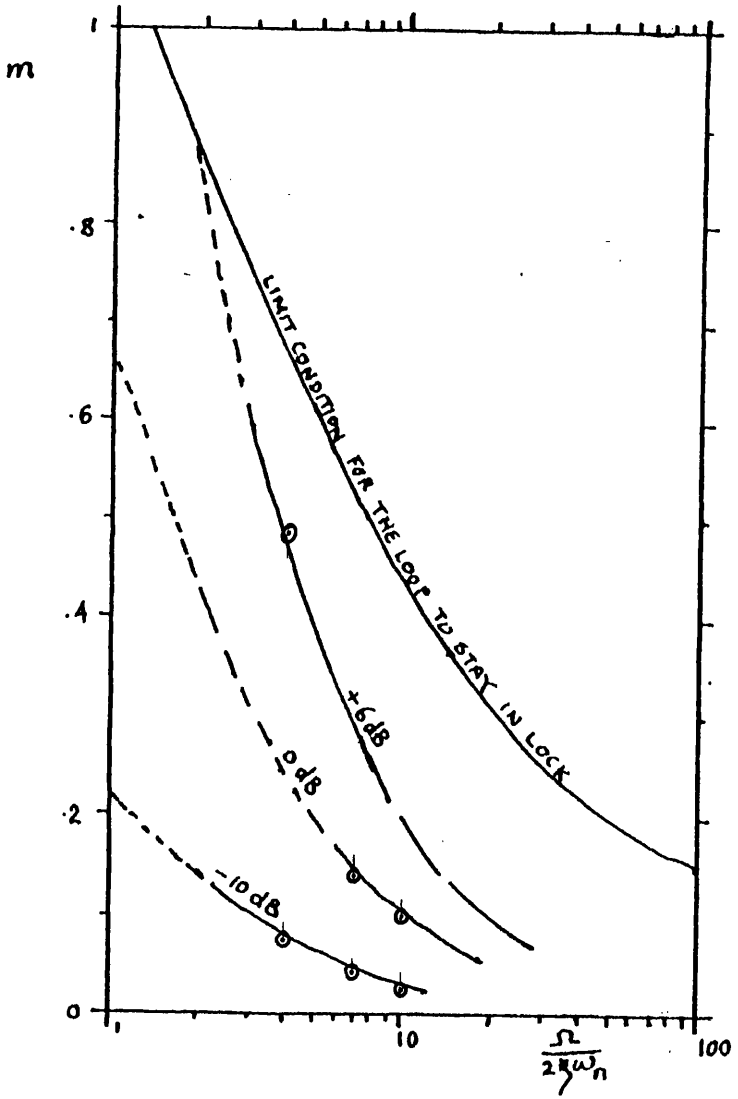
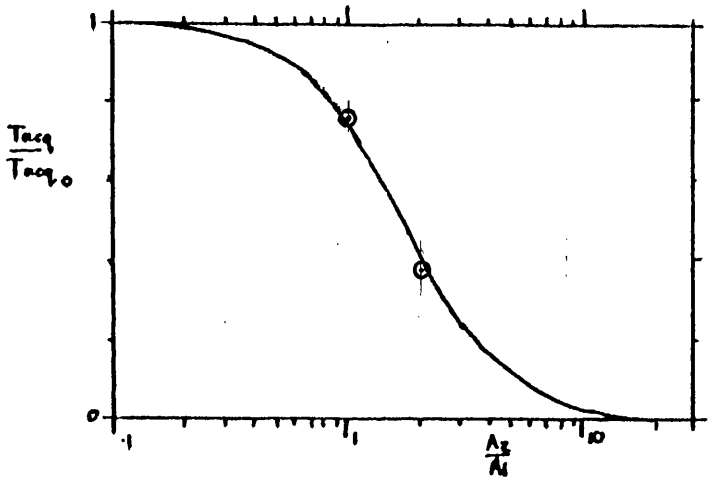


Fig.534. Typical response - PLL initially in Lock



case a



case b

$Tacq_0 = \frac{(\omega_0 - \omega_1)^2}{2\zeta\omega_n^3}$

: pull-in time without interference

Tacq : pull-in time with interference

Fig 535. Interference in PLL - Blanchard plots

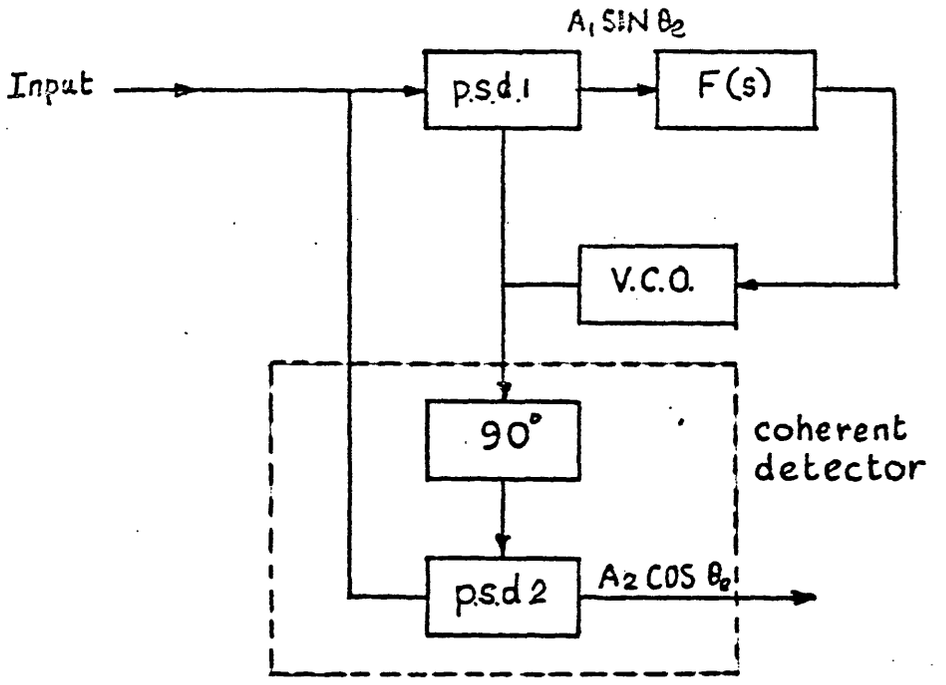


Fig. 541: Basic phase-locked loop with coherent detector

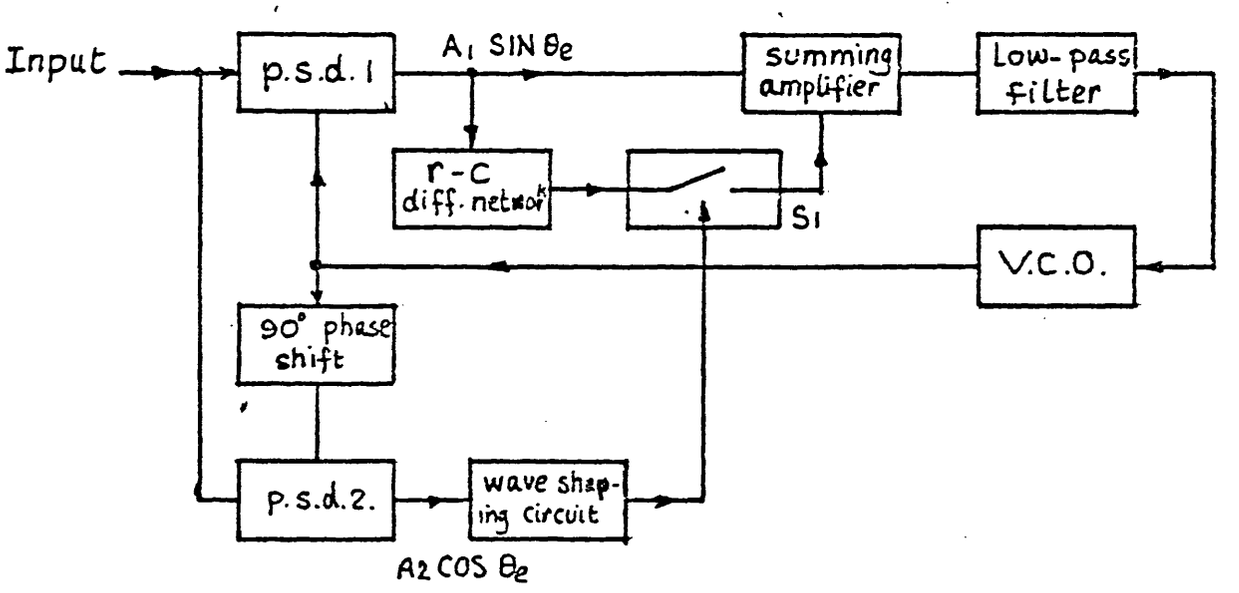


Fig. 542. Schematic diagram of modified phase-locked loop

switch S_1 open for $\cos\theta_e > 0$

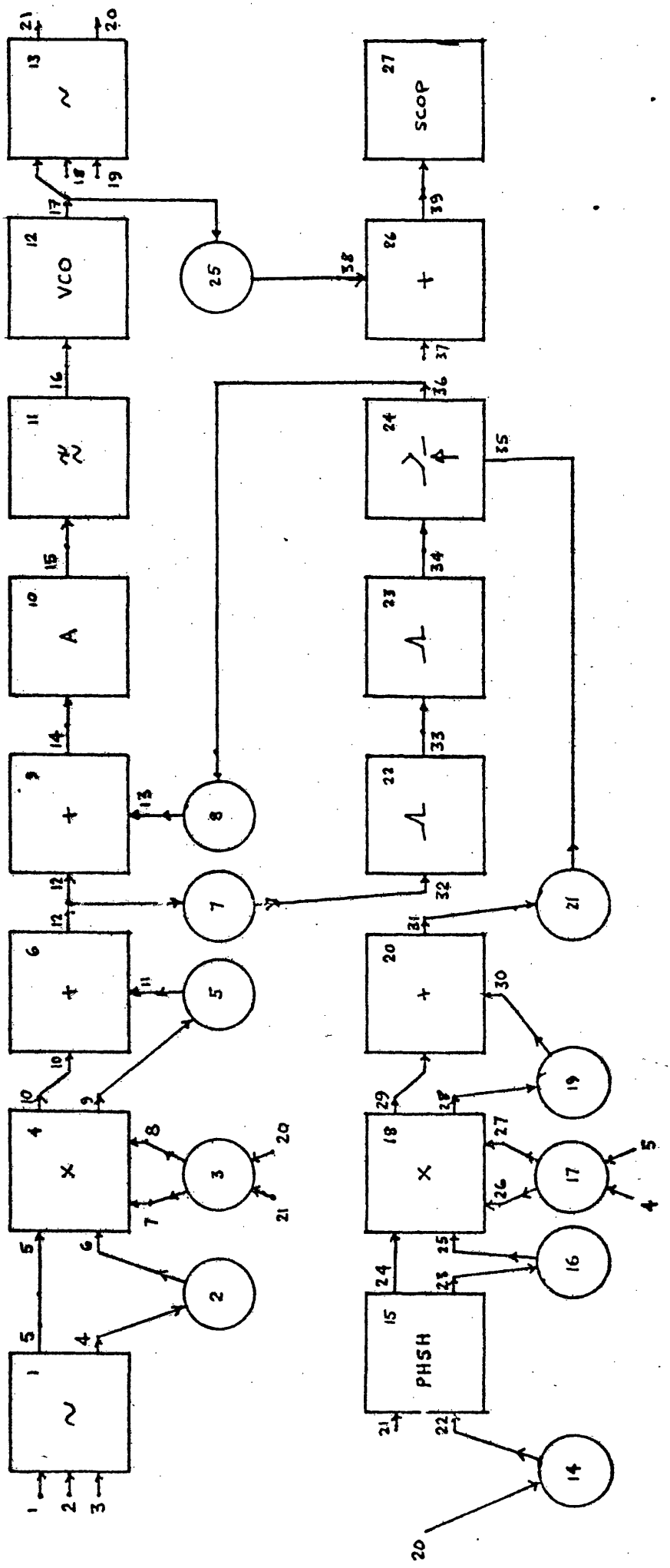


Fig.543. Fast acquisition PLL_ Block Diagram

```
RATE                                  Sampling frequency  fs  Hz
    200000.0
SIGN                                  Block one : Sinusoidal generator
    195000.0
          1.0
          0.0
         -1.0
BRCH                                  Block Two : branching
1
          4
          6
BRCH                                  Block Three : branching
2
         21          20
          7           8
PHSD                                  Block Four : phase-sensitive detector
          1.0
BRCH                                  Block Five : branching
1
          9
         11
ADDR                                  Block Six : adder
BRCH                                  Block Seven : branching
1
         12
         32
BRCH                                  Block Eight : branching
1
         36
         13
ADDR                                  Block Nine : adder
GAIN                                  Block Ten : gain
          0.118891076
FIL1                                  Block Eleven : filter
       1414.0
VCO                                   Block Twelve : voltage control
     200000.0
      91514.0
SIGN          --                      Block Thirteen : oscillator
          0.0
          1.0
          0.0
         -1.0
BRCH          ____                    Block Fourteen : branching
1
         20
         22
PHSH          ______________          Block Fifteen : phase-shifter - degrees
        -90.0
BRCH          ______________          Block Sixteen : branching
1
         23
         25
BRCH          ______________          Block Seventeen : branching
2
          4           5
         26          27
PHSD          ______________          Block Eighteen : phase-sensitive detector
          1.0
BRCH          ______________          Block Nineteen : branching
1
         28
         30
ADDR          ______________          Block Twenty : adder
BRCH          ______________          Block Twenty one : branching
1
         31
         35
NSTR          ______________          New start
DIFR          ______________          Block Twenty two : differentiator
          0.99999949
         -0.86810806
         -0.10779165
         -0.87602073
          1.0
DIFR          ______________          Block Twenty three : differentiator
          0.32672838
         -0.44183252
          0.33494085
         -0.51312758
          0.36804011
SWCH          ______________          Block Twenty four : switching
          0.000001
BRCH          ______________          Block Twenty five : branching
1
         17
         38
NSTR          ______________          New start
ADDR          ______________          Block Twenty six : adder
SCOP          ______________          Block Twenty seven : scope
TRAN                                  Transient
Y             ______________          Y axis
        500          39
END
          1         0.0          1  ┐
          2         0.0          1  │
          3         0.0          1  │
         18         0.0          1  │
         13         0.0          1  ├─ Initial conditions and terminators.
         36         0.0          1  │
         37   -200000.0          1  │
         17    200000.0          1  │
         19         0.0          0  ┘
```

Fig. 544. Fast acquisition PLL _ ICOSS listing

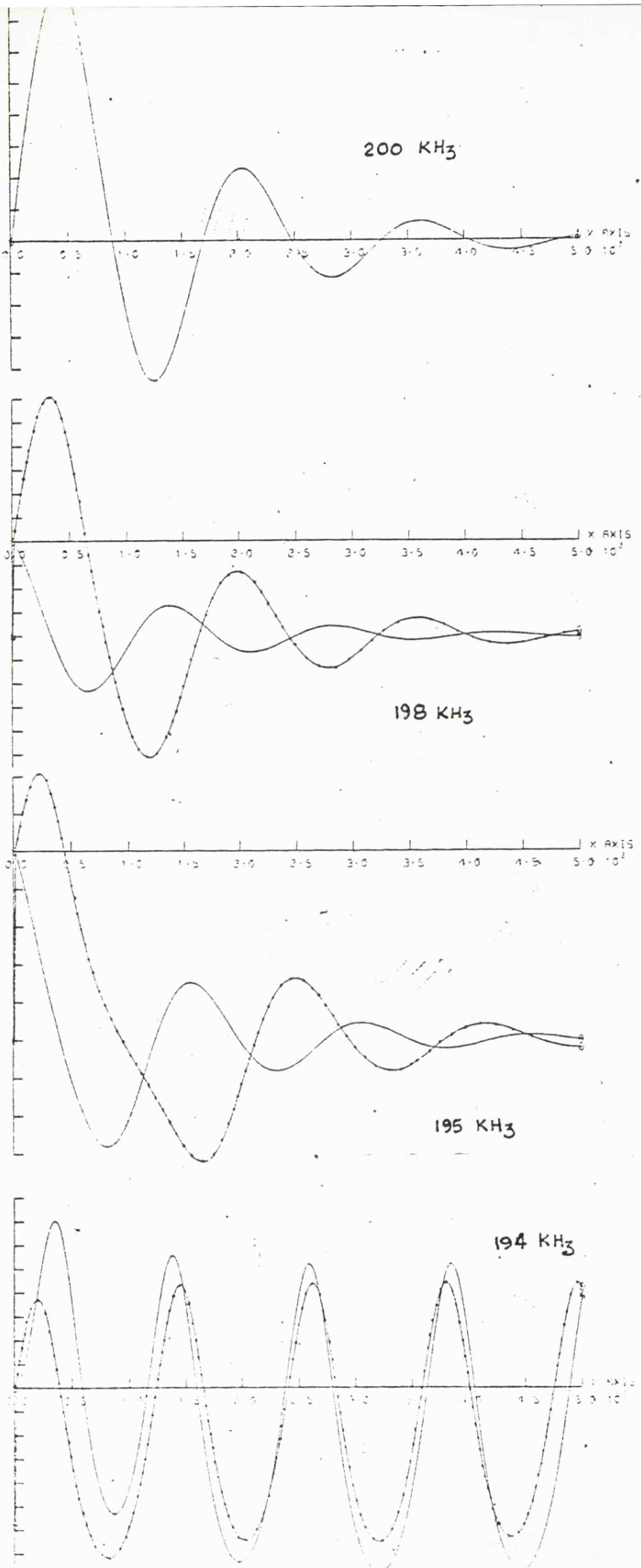
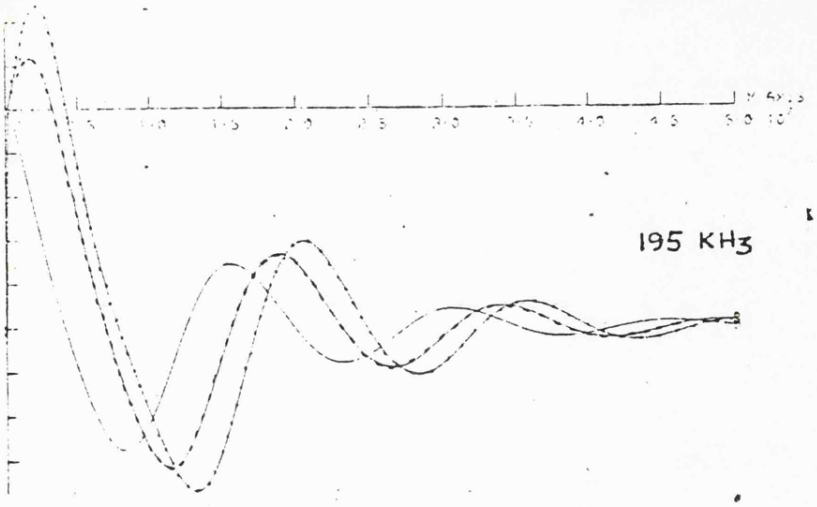
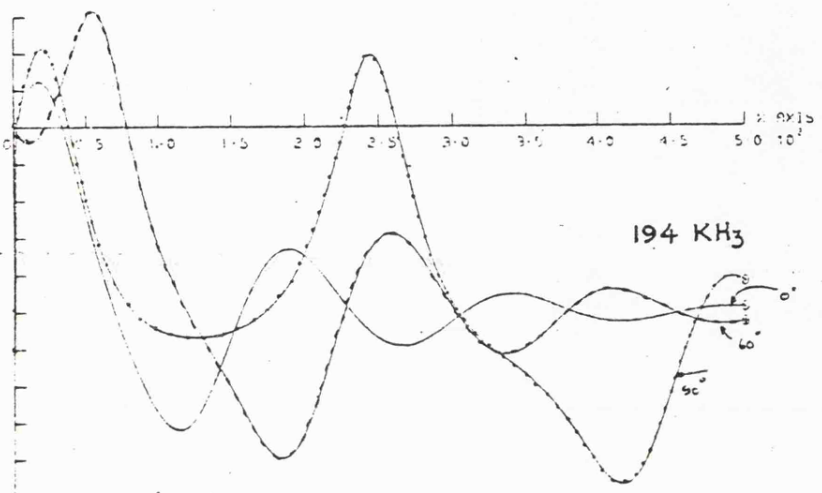


Fig. 545. Fast acquisitio
PLL with/without

with _____

without .....





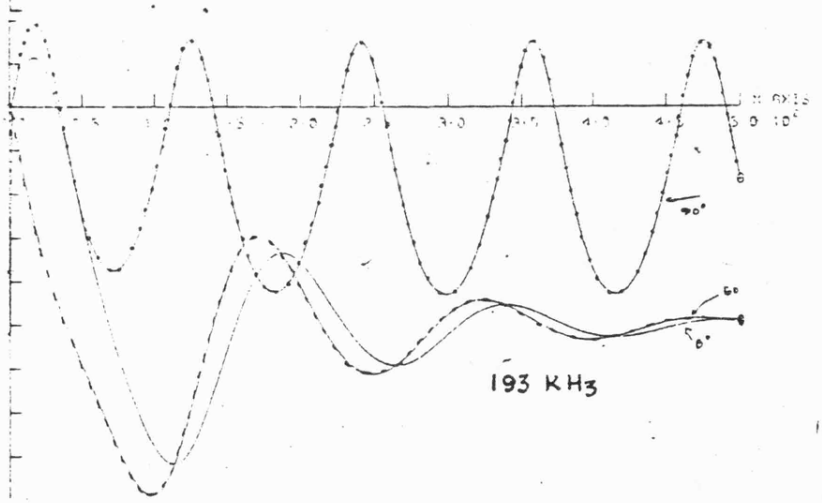
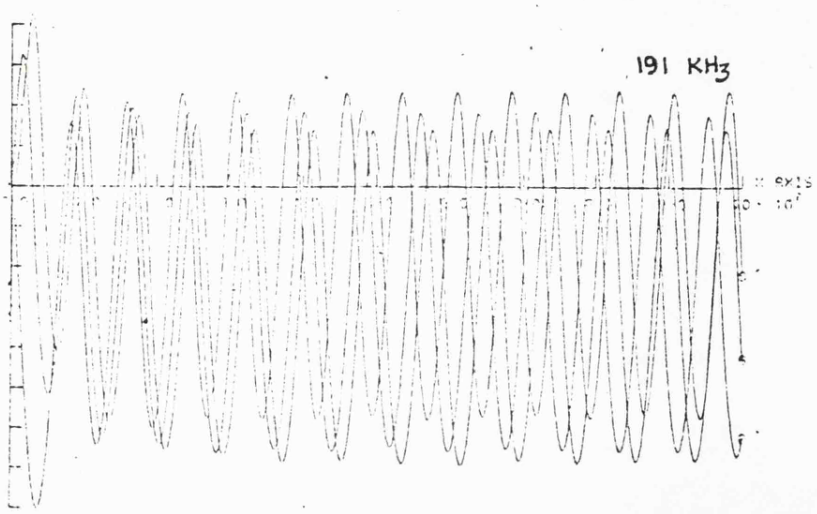


Fig. 546. Fast acquisiti
PLL - phase shift respon

90° ●●●●●●●

60° ———————

0° ———————



CHAPTER 6

DISCUSSION

6.1 DISCUSSION OF COMMUNICATION SYSTEM SIMULATION IN GENERAL

Before discussing ICOSS system and its applications in detail, a summary of the features of communication system simulations and their advantages will tie up with the discussion to follow. There are two main areas where communication system simulation provides a valuable aid:

(i) In research: the main advantages in simulation techniques are[35]:

(*) Simulation is excellent for quick assessment of a new proposal, and very good for obtaining a feel for the first order effects of various parameters, some of which may remain implicit in the actual hardware.

(*) Carefully used, communication system simulation can place proposals in ranking order of feasibility, and thus avoid unprofitable proposals. A simulation study may not necessarily indicate what is to be built in hardware, but it can usually eliminate those

concepts which should not be attempted.

(*) The discipline of modelling increases the understanding of the real system.

(*) Optimum parameters of a system may be obtained, by multiple runs.

(ii) In teaching[35]: students can be left to experiment with systems of their devising, without being hampered by the lack of suitable hardware, and quickly come to appreciate the basic limitation imposed by bandwidth restraints, group delay distortion, spectral manipulation etc. Of course, unless this is backed up by routine experiment work, the student may cling to the idea that ideal systems components are both possible and desirable. Simulation must always be kept in close relationship to reality.

In the absence of a general-purpose communication system simulator, the researcher or the student, will tend to simulate a special-purpose simulation package for the particular problem at hand. Their attempts will mean, inevitably, spending some time for studying computer programming, some computer science and computer runs to establish that this simulator is giving the correct results. This also means, and more so in research, an appreciable percentage of the overall time spent by a researcher could be saved if an already established general purpose simulator with a known efficiency is available.

6.2 DISCUSSION OF ICOSS

(a) ICOSS main features

In the pursuit of a general-purpose communication system simulation, the ICOSS was designed and implemented. It was a time-domain based, and on-line oriented simulator with interactive facility, which has number of important features. These features are discussed first:

(*) Simulate any signal processing system due to its unlimited flexibility in the signal processing modules, see below.

(*) The user is capable of changing parameters of individual signal processing (SP) modules or the overall system control parameter (GCP) while the system is still running, through a buffer, which makes the simulation as close to the real system as possible. This avoids the troublesome procedure of (stopping a run-initialise system-change parameters - rerunning) approach of other communication system simulators. The alternative approach of other simulators has the following apparent faults:

(i) Repetition of runs means great increase in computer time and programming inconvenience.

(ii) The transient and system response during a change of control parameter of a module will be lost.

(iii) The true resemblance of actual running of a system is also lost.

(*) Editing: the editing capability of ICOSS is another facility for the simulation of communication system in a practical way. The insertion or deletion of a module will:

(i) Give a realistic assessment of the behaviour of the tested system.

(ii) Avoid the problem of other simulator procedures in (stopping a run-editing-rerun), which involves the disadvantages mentioned above.

(*) Interactive action: The interactive action of ICOSS provides the user with the following:

(i) With the interactive dialogue, the construction of the simulated model becomes an easy task.

(ii) Accidental errors are greatly avoided, and automatically deducted.

(iii) The simulator becomes under the user's complete control.

(*) ICOSS allows the user to write the block diagram statements of the model in any order, or with the minimum constraints. With the internal structuring facility of the block diagram, the signal flows and is processed in the correct way, according to the system state analysis. This avoids a major source of error which the user may unintentionally make in the block diagram statement sequence, especially for those systems having feedback links.

(*) Portability: ICOSS is written in Fortran IV, a language widely used in engineering and scientific applications. Therefore it can be loaded onto most (if not all) computer systems. In fact, ICOSS was loaded and tested in 3 different computer situations:

(i) The Digital Equipment PDP8/e mini-computer: because of the limitation in size of computer memory, only a modified version of ICOSS was loaded. Also because of the limitation in speed of the computer interfacing (55 KHz) and computer speed, only the multiprocessing activity of ICOSS with a microprocessor sub-system was investigated, see Chapter 4.

(ii) The ICL 4-50: the interactive facility

of ICOSS was fully tested.

(iii) The ICL 2980: the overall operation and function of ICOSS were investigated on patch basis.

(*) Software multiprocessing: It is feasible to couple ICOSS to another simulator using a fourth group of interrupts. With a teletype command, ICOSS can be halted and the guest simulator operated until the required signal values are accummulated. ICOSS could then be resumed to process with the new set of signal values just obtained. As an example, an RF transmitter could be simulated using the original ICOSS, and RF transmission using a guest simulator. This provides a wider scope of application, but will need further investigation.

(*) On-line operation: The full benefit of ICOSS is the real on-line operation. Unfortunately the necessary computing power and processing equipment are rarely available. For this reason, the investigation of ICOSS features were made on off-line operation, with the interactive dialogue stored on a data file. Although this method was simple, the quick response which an on-line system would give, and which ICOSS is designed for, were not achieved.

(b) Discussion of ICOSS internal structure

(with reference to Chapter 3)

(*) Signal processing module treatment:

(i) The simulator investigation was mainly concerned with the simulator structure rather than the modules detailed function. However, the principles adopted, ie with unlimited number of inputs, outputs, local control parameters or state variables, it should be possible to simulate any signal processing module.

(ii) In the final form of ICOSS, an SP module library will be established. It will be possible then to simulate most types of communication system. However if there is a system using a very special and rare type of SP module, a situation will arise in which a compromise must be reached:

1. Either write this unique and special purpose module, and keep it permanently in ICOSS. This is a very inconvenient and uneconomical situation, since it means occupying a space in ICOSS which will rarely be utilised. However, the module may be deleted as soon as it is not needed, utilising the dummy module

approach, see Appendix (G). The process of adding and deleting a module involves compilation and composition of ICOSS, which must be avoided as much as possible. Alternatively, a transient file for the version of ICOSS which includes the new module is generated, run for the particular simulation, and then deleted; leaving the original version of ICOSS untouched.

2. Or supplying the function as data, in similar way as supplying the control parameters, to a special purpose dummy module. Further investigation is required for the implementation of this approach.

(iii) Dummy modules: These can be utilised more extensively for the expansion of ICOSS.

(*) Standard Unit (SU): In communication system problems, there is often a standard unit SU, say a receiver, which is made of a number of basic SP modules. The SU may be a sub-system in a bigger system in which SU internal function does not need to be observed. The running procedure described in (3.6) involves a number of substitutions and directives. This means that for multiple runs, an appreciable time could be

saved if some reduction of these substitutions for the SU could be devised. However, there are a number of points which must be remembered:

(i) The state variables calculation and signal flow has still need to be performed with the SU.

(ii) Only CPs calls need further investigation.

(*) SU further: The facility provided by the teletype interrupt ENTR, see Chapter 3, can be further developed. Instead of inputting complete model block diagram only, as the case at the moment, it can be made to accept SU models as well. This additional facility will require SU library, some delimiter, and the utilisation of the EDIT facility, as shown in Fig 6.4.1.

(*) I/O peripherals:

(i) Graph plotting arrangement: A main feature of ICOSS structure is its modularity. Each module can be replaced or modified semi-independently. In the case of the plotting routine, which represents a teletype interrupt PLTG, the procedure for employing alternative types of plotting devices is a simple one. The data for Y, X etc

are supplied automatically to PLTG in either accummulated form or directly. The plotting management interrupt PLTM can be used if other requirements are needed by the new plotting device. Of course, the ultimate aim is to have PLTG interrupt constructed as a slave (peripheral) processor, similar to BOF processor described in Chapter 4. The plotting procedure is then carried out outside ICOSS semi-independently. Another advantage with this method is that the processor will have its own storage, which enables the user to arrange the graph display independently of time. This arrangement needs full investigation.

(ii) Input buffer unit: Inputting data related to change of parameters require a special buffer unit, in which the new CPs, as supplied by the user, wait for the interrupt to take place and are automatically submitted into ICOSS. During the course of this work, the buffer action was simulated. Therefore, this facility should be added and overall operation investigated.

(*) Loading ICOSS on computers: The flexibility in ICOSS makes the simulator completely portable.

When loading ICOSS on a new computer, the minimum sub-set of programs, ie ICOSS, TTYFG, LOOP and FLAG, see Appendix (D), are loaded first. Then, there are no limits on the number of interrupts one requires; the only limitation imposed is the size of the recipient computer memory. Of course interrupts of the various groups are added according to their order of importance, which means that some of the facilities may either be increased or decreased. However, some programming techniques can be utilised, such as overlaying and segmentations. Inspecting ICOSS structure, Fig 3.2.3 , the program execution is performed as lines of sub-routines, and only 10% of those subroutines operate at once, ie need loading. This fact, which was utilised in the running of the present version of ICOSS, can be employed in other computer systems.

(*) Hardware multiprocessing: The capability of software multiprocessing can be further developed, by real hardware multiprocessing. In this, two mini-computers are used: one holding ICOSS and the other holding the new simulator, and working in a similar way as BOF, Chapter 4. The exception is that in the latter case the separated interrupt was within a group of other interrupts, whereas in the new concept, a whole group of interrupts are contained within ICOSS. Further, it is

feasible that any group of interrupts of ICOSS, say TTYFG, may be separated from the rest of ICOSS and loaded onto another mini-computer or a dedicated processor, and allowing the system to run in parallel. The advantages of this technique are:

(i) Faster execution time, which is important for real-time operation.
(ii) Save on memory locations.
(iii) Additional facilities can be loaded.
(iv) The fixed procedures are loaded on dedicated processor.
(v) Increased improvement in the computer utilisation.

However, the programming of multiprocessing interfacing and timing control (on the system operation) needs great care. This approach needs further investigation, if the necessary equipment are available.

(*) Segmentation technique: The time sharing arrangement of ICOSS, Section 3.3, made possible time-independent operation to be processed alongside the time-dependent operation of ICOSS. This involves slicing the time-independent operation into segments on well defined boundaries within it, and making the function of each segment an independent interrupt, to be called as many times

as needed by the FLAG group of interrupts, until the various controlling factors become ready for the action of the next interrupts, and so on. The time allowed for each interrupt is small, and hence the division of the segments must be made accordingly. This technique was proved successful in the FFT operation as explained in Appendix (B).

(*) FFT: The importance of the frequency response of a system is well known in communication system analysis. The segmentation technique above, provided the means of obtaining a quick frequency response using the FFT at any moment of time during the running of ICOSS (which is a time domain simulator). The importance of this technique lies in the fact that: a system could be modified, ie change some of its parameters or delete or add some modules, and have its frequency response examined for each set of parameters and modules, all in one run.

The alternative approach, in which a fixed number of signal points are taken out from the simulator to be processed by another dedicated simulator for FFT, is extremely cumbersome, and does not assist the user easily, as well as not coming as near to the real situation in the practical system. Also the points raised in (a) above, for the change of parameters will apply to this situation too.

6.3 DISCUSSION OF APPLICATION PROBLEMS

In testing the validity of ICOSS in simulating practical problems, there are four areas of discussion covering the design targets:

(a) Comparison between ICOSS vs special-purpose package (SPP):

* Convenience: the gap in convenience between the two is enormous in favour of ICOSS, but the outstanding feature the flexibility of ICOSS by which the module structure can be constructed. Even the fixed structure of SPP for the phase-lock loop (PLL) which supposedly easier to run is overcome by the use of ENTR interrupt in ICOSS which meant testing a fixed system structure again and again without the need for constructing it for every run.
* Difficult tasks: the control parameters (local and global) variation capability as well as editing facility provided the means of tackling difficult tasks.

* Time reduction: the ability to modify the system and change its parameter meant a vast reduction in the number of runs required. In fact with one run only any

test can be performed on the system as opposed to the multiple runs situation of the SPP. This in turn means reduction in computing time.

* Transients: during changes in system parameters, the transient effect is preserved in the ICOSS situation, and provided valuable understanding to the system behaviour.

(b) Comparison between ICOSS, Direct/Complex signal approaches:

* Application: although both techniques produce similar results, as expected, the complex technique may find useful application for the system which requires phase-shifting, say, as the case for the fast acquisition PLL problem.
* The complex signal approach eases the problem of high sampling frequency for slow variation of amplitude and phase in a high frequency carrier system.
* The direct approach however provides better understanding of signal processing in time domain.

(c) Interference in PLL:

The tests for this type of problem showed:

* Parameter variations, and their effect on the system response (transient and steady) are easy to achieve.
* Multiple plots at any node or instant of time can be achieved within one run.
* Derived variables can be plotted relative to any other variable.

(d) Fast acquistion PLL:

* Achievement: the behaviour of single loop PLL is difficult to analyse by its own, especially the transient response, but with additional loops as in this case, the understanding of behaviour at different nodes of the loop is even more difficult. However, with the capability of accessing any node, this understanding becomes easy to achieve.

6.4 FUTURE WORK

The prototype structure of ICOSS so far discussed, together with complementary investigations of BOF etc, lay the foundation to the establishment of the full system. There are five main areas of development required in that direction:

- (*) The expansion, and completion of the present ICOSS library etc.
- (*) Improvement in the speed of execution, for real-time operation.
- (*) The utilisation of some of the peripheral equipment.
- (*) The coupling with other simulator systems.
- (*) Further development toward full stochastic simulation.

These developments can be divided into 2 groups according to the facilities available:

(a) Immediate: With the existing computer and equipment facilities available in the University laboratory:

(*) Write up ICOSS in its final form:

(i) Extend the signal processing (SP) module library, with each made as general and to include as many functions as possible

within one fundamental operation. This can be made in similar line to the FILT module, as described earlier - Appendix B , which was a general-purpose module containing most of the important types of digital filter, eg BPass, LP, HP, BStop, Chebychev, Butterworth, for unlimited order. As a suggestion, the following SP modules should be added:

1. Signal generation: saw-tooth, square (rectangular) wave, pseudo-random etc.
2. Demodulators: envelope, amplitude, frequency, phase.
3. Filtering: non-recursive, linear phase, arbitrary specification.
4. Non-linear functions: clippers, power series.

Also, in order not to confuse the users of ICOSS in adopting misleading mnemonics, the existing and new SP modules should be grouped as follows:

1. Signal generation: sine-wave, saw-tooth, square (rectangular) wave, pseudo-random etc.
2. Modulators: amplitude, frequency, phase.
3. Demodulators: amplitude, frequency, phase.

4. Filtering: non-recursive, linear phase, arbitrary specification.
5. Differentiators: integrators.
6. Non-linear functions: eg clippers, power series.

The modules are for the deterministic type of system; more modules are to be added for the stochastic simulation purposes, as outlined below.

(ii) Modify the interrupt ENTR and construct SU library in the way explained earlier, Section (6.2b) and indicated by the flow chart of Fig 6.4.1.

(*) Up-date the dummy module concept:

(i) The general-purpose dummy module: Leading from section (6.2b) with regard to special SP module, it is feasible that set of equations to include addition, multiplication, of variable and a simple zero-pole module, etc are constructed in this special dummy module. These sets of equations can be controlled by set of parameters which the user can supply in order to control the flow of calculation

for the input signal. The idea of this method is that any special (rare) function which is not in the SP library can be constructed temporarily and used by supply extra set of parameters. This is only a first hand suggestion, which must be updated with tests and application.

(ii) Rename the module mnemonic (of the subroutines in the LOOP group) and replace with SUB1, SUB2,...,SUBN. The particular SP module function will then be identified by the mnemonic/module number conversion table which must be constructed.

(iii) In up-dating the dummy module concepts, special care must be taken with the controls of the LOOP program.

(*) Investigation into the following:

(i) Coupling with other simulator system, ie software multiprocessing, eq QUASIM (quasi-synchronous simulator) currently being developed at Bath University[19].

(ii) The standard unit (SU) concept: the alternative approach to ENTR mentioned above. The investigation must include the possibility of reducing the inter-links between the internal modules and the

possibility of control parameters (CPs) reduction. A receiver module will be a good testing model.

(*) Stochastic Simulation: Further developments are needed for this type of simulation. They include:

(i) Additional SP modules to be included in the SP library, eg

1. Noise generation (stochastic)
2. Random signal generation
3. Rayleigh fading

(ii) Additional measuring devices (output display routines) eg

1. Averages
2. Variance
3. Correlator

(*) Dialogue: The present form of interactive dialogue of the teletype interrupts has been tested only by the author. The experience of other users of ICOSS will provide a source of improvement to the dialogue.

(b) Future: Given the supporting equipment, such as a microprocessor chip, arithmetic unit (AU), interfacing etc, and an appropriate computer power:

(*) The first and most important objective is to develop ICOSS system with its full real-time on-line operation. This will require a solution to the problem of speed of program execution. The method outlined in Chapter 4 must be fully utilised. However, the system must be built up in stages, and the following suggested procedure followed:

(i) Load the modified version of ICOSS into a minicomputer, say the Digital Equipment PDP8/e or PDP11. This version of ICOSS will include the minimum subset of subroutines as outlined earlier, Section 6.2.

(ii) Build-up the master/slave system with the BOF module in its final form, see below.

(iii) The BOF interfacing must be treated so that the data transfer rate is higher than the slave module processing speed.

(iv) Trials with this configuration will show the region which most needs improvement, and more modules may be taken out of the host computer and stored in a slave processor as in BOF above.

(v) Utilise the time-scaled signals, ie

record signal on a disc or tape, say speech, and play at slower speed.

(*) BOF: This module can be speeded up greatly by using higher speed microprocessor chip, coupled with the arithmetic unit as suggested in Chapter 4. Therefore number of alternative trials must be made to achieve an efficient bank of filter processor. These alternatives include 16 or 32 bit words, for higher dynamic range. Finally, an investigation must be made to have a single chip containing this facility.

(*) SP programmable board: Leading further from above, it will be extremely beneficial if a programmable microprocessor controlled unit could be constructed so that SP functions are programmed directly. With the aid of the dummy module concept, any number of additional SP modules could be added to the system with the minimum interference in the main body of ICOSS.

(*) Establish the peripheral equipment for:

(i) input buffer: where the new control parameters are placed waiting for the appropriate space of time to be inserted into the system, as explained earlier.

(ii) I/O display for:

1. Graph plotting and display, as has

been explained in Section 6.2.

2. Block diagram construction and display.

(*) Having hardware multiprocessing with one group of interrupts or more, say the TTYFG, on separate processor (mini-computer). This means that the function of ICOSS will simply be a controlling program for a number of dedicated processors, each having special function (or group of interrupts as defined in Chapter 3).

(*) Finally: the ultimate aim is to have ICOSS side by side to the hardware apparatus of the communication engineer in the research and development laboratory, complementary to each other in the investigation of new ideas and practical problems in communication engineering. Therefore, the size of the equipment holding ICOSS is important, and the above suggestion will make this a reasonable size (small).

6.5 CONCLUSION

A time-domain interactive communication-system simulator (ICOSS) was designed, and put into test. The main objectives for this simulator were achieved, that is

* being able to change the simulated model control parameters while in the running mode.
* being able to edit its module structure (block diagram).
* be portable, and
* operate interactively for real-time simulation.

Unfortunately, due to lack of some equipment and computer power, it was not possible to demonstrate the full features of ICOSS. However, an investigation was made towards a possible solution to the inherent problem of on-line simulation, ie the need for fast processing and high sampling frequency. A dedicated (slave) processor containing a signal processing module, the bank of digital filters, and working in parallel with a host computer where ICOSS resides was studied. With further developments, and with the introduction of some fast and dedicated processors, the demands for making the communication system simulation as another bench tool for research and development to complement the hardware apparatus, could be met by ICOSS. However, there is still more work needed to be done to fully develop the system, so that further problems other than the deterministic

type, which was discussed in the previous section, ie stochastic simulation, may be solved in order to assess the full impact of this simulator.

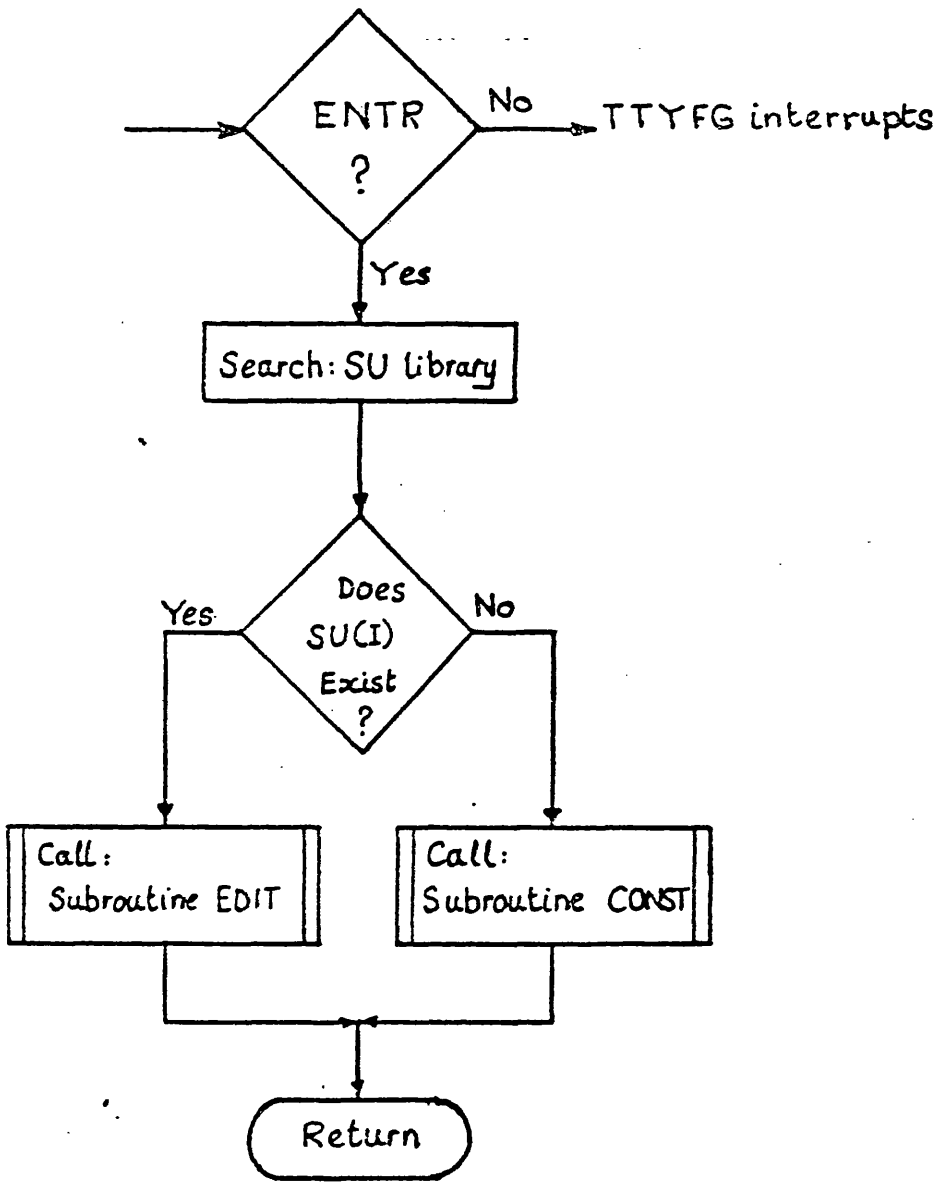


Fig.641: SU call using ENTR interrupt

ACKNOWLEDGEMENTS

The author wishes to acknowledge the following who have enabled this work to be completed:

(i) Professor W Gosling, Head of School of Electrical Engineering at the University of Bath for the facilities provided.

(ii) Mr J D Martin, for his great help, advice, understanding and patience when acting as project supervisor.

(iii) The Iraqi Government for their financial support.

(iv) Dr J P McGeehan for helpful discussions.

(v) My family for their support.

(vi) Mr M Al-Douboni for his assistance.

(vii) Mrs F Williams for her assistance and hard work in producing this thesis.

REFERENCES

1. Ackroyd M H: 'Digital Filters', Butterworths, 1973.

2. Altman L: 'Microprocessors', Electronic Book Series, McGraw-Hill, 1975.

3. Andrews M: 'Minicomputer CSSL simulation of PLL', IEEE Region Six Conf on Minicomputers and their applications, pp 21-24, May 1973.

4. Aspinall D and Dagless E L: 'Introduction to microprocessors', Pitman, 1977.

5. Beauchamp K G: 'Signal processing - using analog and digital technique', Unwin Ltd, 1973.

6. Bergland G D: 'A guided tour of the Fast Fourier Transform', IEEE spectrum, Vol 5, pp 41-52, July 1969.

7. Blanchard A: 'Interferences in phase-locked loops', IEEE Trans on Aerospace and Electronic Systems, Vol AES-10, No 5, pp 686-697, Sept 1974.

8. Bogner R E and Constantinides A G: 'Introduction to digital filtering', J Wiley, 1975.

9. Coates R: 'Approaches to the simulation of communication systems', Proc 8th AICA Congress on Simulation Systems, pp 233-242, 1976.

10. Coates R and Kwok K Y: 'Communication system evaluation program COSEP', Colloquium on computer simulation of communication systems, IEE Electronics Division, pp 3/1-3/3, 12 May 1975.

11. Cullyer W J: 'Application of Fourier techniques to computer-aided design of electronic systems', Proc IEE, Vol 118, No 3/4, pp 437-448, March/April 1971.

12. Davis B R, Beare C T, Coutts R P and Le N H: 'Computer simulation of communication systems', Electrical Engineering Department, University of Adelaide, South Australia, Report No 2-75, 1975.

13. Enslow Jr P H: 'Microprocessor organisation - a survey', Computing surveys, Vol 9, No 1, pp 103-129, March 1977.

14. Epstein P L: 'Small master station with video display implemented with multiple microprocessors', Conf proceeding, Remote Supervisory and Control 77, Remscon, pp 67-79, 27-29 April 1977.

15. Flake P L, Musgrave, G and White I J: 'A digital systems simulator: HILO', Digital Processes, 1, pp 39-53, 1975.

16. Freeman E and Fashano M: 'System time-domain simulation program', NASA report No R71-014, Feb 1971.

17. Gabel R A and Roberts R A: 'Signal and linear systems', J Wiley, 1973.

18. Gardner F M: 'Phase lock techniques', J Wiley, 1967.

19. Gladstone K J and McGeehan J P: 'A computer simulation of the sideband diversity scheme for mobile radio', to be read at IERE Conference, International Conference

on computer aided design and manufacture of electronic components, The University of Sussex, 3-5 July 1979.

20. Gold B and Rader C M: 'Digital processing of signals', McGraw-Hill, 1969.

21. Golden R M: 'Block diagram compiler B', Bell STJ, XLV, 3, pp 345-358, March 1966.

22. Gordon G: 'System Simulation', Prentice-Hall, 1969.

23. Holding D J, Jacovides D and Mamdani E H: 'On the use of conventional computers to provide parallel execution of simulated software', Proc 8th Congress on Simulation Systems, pp 523-526, 1976.

24. Kadokawa Y et al: 'Computer simulation of the constant net loss single sideband system for the land mobile communications CNL-SSB simulation', Review of Radio res labs (Japan), Vol 19, No 101, March 1973.

25. Karafin B J: 'The new block diagram compiler for simulation of sampled-data systems', AFIPS Conf Proc 27: part 1, Fall joint computer conference spartan books, Washington DC, pp 55-61, 1965.

26. Kelly Jr J L, Lochbaum C and Vyssotsky V A: 'Block diagram compiler', Bell STJ, 40, pp 669-676, May 1961.

27. Kingsbury K G and Kelly L C: 'A digital filter bank of real-time speech analysis and synthesis using logarithmic quantised signals', Conf on Digital

Processing of signals in communication, IERE Conf Proc, No 37, pp 81-96, Sept 1977.

28. Kingsbury K G and Rayner P J W: 'Logarithmic arithmetic for digital filters', Proc of symposium on digital filtering', Imperial College, London, Sept 1971.

29. Lockhart G B and Cheetham B M G: 'Minicomputer simulator for digital communication systems', Colloquium on computer simulation of communication systems, IEE Electric division, pp 5/1-5/2, 12 May 1975.

30. Lockhart G B, Cheetham B M G and Mansson B M: 'A real-time simulator for digital signal processing', Loughborough Conference on digital processing of signals in communications, Sept 1972.

31. Manasswitsch V: 'Frequency synthesisers theory and design', J Wiley, 1976.

32. Manufacturer Manual: 'Programmable read-only memories', Fairchild Semiconductors, July 1977.

33. Manufacturer Manual: 'M6800 microprocessor programming manual', Motorola Co, 1975.

34. Manufacturer Manual: 'M6800 microprocessor application manual', Motorola Co, 1975.

35. Martin J D: 'Sigsim: a general-purpose signal-processing simulation program', School of Electrical Engineering, University of Bath Publication, March 1975.

36. McGeehan J P: 'A new technique for improving the acquisition performance of second-order phase-locked loops', Electronic Letters, Vol 14, No 2, pp 42-43, 19 January 1978.

37. McGlynn D R: 'Microprocessor: technology, architecture and applications', J Wiley, 1976.

38. Metcalfe J: 'An investigation into the use of digital filtering methods applied to the simulation of continuous communication systems', PhD Thesis, University of Bath, 1976.

39. Morris R: 'Scatter storage technique', CACM, Vol 11, No 1, pp 38-44, Jan 1968.

40. Morrow R J and Warren C S: 'The application of hybrid computers to the design of digital communication systems', Conf on Digital Processing of signals in communication, IERE, Proc No 23, April 1972.

41. Musson J T B and West B G: 'Program for the simulation of modulated communication systems - MODSIM', Marconi Survey Series, MSS 75/8, April 1975.

42. NASA Report: 'Computer for real-time flight simulation', No CR2885, pp 4.1-4.10, Nov 1977.

43. Peatman J B: 'Microcomputer based design', McGraw-Hill, 1977.

44. Rabiner L R and Steiglitz K: 'The design of wide-band recursive and non-recursive digital differentia-

tors', IEEE Trans audio electroacoustic, Vol AU-18, pp 204-209, June 1970.

45. Ramamoorthy C V and Li H F: 'Pipeline architecture', Computing Surveys, Vol 9, No 1, pp 61-102, March 1977.

46. Sakai T and Numi Y: 'Programming system for Block diagram simulation and its application', Electronic and communication in Japan, Vol 51-C, No 9, pp 133-142, 1968.

47. Seynaeve R, Hug E and Vettori G: 'Interactive time series analysis', Signal Processing - Proc of NATO, advanced study institute on signal processing, pp 183-203, 1973.

48. Skwirzynski J K: 'Simulation techniques for the study of modulated communications channels', Proc IEE, Int Symp on Circuits and Systems, San Francisco, pp 22-25, April 1974.

49. Smith D G and Martin J D: 'Digital simulation survey', School of Electrical Engineering, University of Bath, Internal University Publication, January 1978.

50. Ulrickson R: 'Software modules are the building blocks', Electronic Design, No 3, pp 62-66, 1 Feb 1977.

51. Viterbi A J: 'Principles of coherent communication', McGraw-Hill, 1966.

52. Voelcker H: 'FFT program', Private Communication, University of Rochester, New York State.

53. McGeehan J P: Private Communication, School of Electrical Engineering, Bath University 1979.

APPENDIX A

The following table contains the main features and characteristics of a number of important communication system simulators (three time-domain, and two frequency-domain). However there are many other communication system simulators, which can be found in literature as mentioned in Chapter 2.

Simulator	BLODI: Block Diagram Compiler	BLODIB: Block Diagram Compiler, B	SYSTID: System time domain simulation program	WASP: Waveform and Spectral-analysis program	SIGSIM: Signal processing simulation program
Author	J L Kelly, C Lochbaum and V A Vyssotsky	R H Golden	E Freeman and M Fashano	J Cullyer	J D Martin
Domain	Time	Time	Time	Frequency	Frequency
Application	As below	Study of sampled data system such as vocoders and data transmission systems. Analogue systems could also be simulated, but the bi-linear z-transforms of their filters may cause errors.	Study of analogue and digital signals with/without noise interference, and optimisation of channel and signal parameters.	Study of analogue and digital signals in noise and inference, and optimisation of channel and signal parameters.	The study of analogue and digital signals or the optimisation of channel and signal parameters in communication systems.
Philosophy	Early approach in which BLODI program accepts input program written in BLODI language. The latter corresponds closely to an engineer's Block diagram of a circuit. The input code consists essentially designing the connectivity of a number of boxes drawn from an alphabet of about 30 types.	The language is designed for programming sampled-data systems which may be represented either in block-diagram form or in the mathematical representation of the z-transform calculus. The structure of BLODIB allows convenient specification of the system parameters as well as permitting these parameters to be varied in order to study changes in system performance.	The idea is to allow the user to achieve time-domain simulation of telecommunication links. The SYSTID program characteristic is achieved through the use of a language processor which translates simple English language user command and link element descriptions and topology into the Fortran code necessary to establish a digital filter equivalent of each link. It employs bilinear transform. Frequency domain responses are computed for time responses by post-processing routines.	To establish a fixed structure for the analysis of waveform and spectra in these systems (communication) and then to build onto this framework a library of electronic or electro-mechanical modules of the types needed for specific application.	a) To provide a facility for simulating many communication type sub-system and signal processing arrangements b) This facility must be capable of being used by persons familiar with signal-processing but not necessarily with computing c) While retaining the utmost flexibility the input data and specification of operations must be in common-sense form.
Capabilities	as above	40 basic building blocks for communication systems (mainly digital). Approximations to analogue systems are possible. Parameter changes can be conveniently dealt with-in the program. 20 kHz max input sampled to 4096 bits (including sign).	All common modulation modes, noise, filters, non-linearities. (including hard-limiters) and adjacent channel interference evaluation; in fact its capabilities are very similar to SIGSIM.	Most common modulation modes, gaussian noise, adjacent channel interference signals, non-linearities and filters. Does not cater for low-pass equivalent signals or filters. Can handle off-set transmitter situations for all standard channel spacings at present in use.	All modulation modes, gaussian noise, an extensive library of common time, waveforms, filters specified by arbitrary frequency responses or pole-zero locations and frequency spectrum/time waveform monitoring facilities. Adjacent channel situations can be handled at up to 100 kHz channel spacing with better than 500 Hz resolution-reduced channel spacing gives increased resolution.
Procedure	Object program produced by the compiler consists of 3 parts: a) the prefix which sets up the logic for the main loop b) the main loop which is executed once for each sample processed c) the suffix, which causes the main loop to be repeated in proper number of times, empties output buffers, fills input buffer etc.	3 pre-requisites: a) the determination of an appropriate digital representation for the system to be studied b) the preparation and compilation of a BLODIB computer program which causes the computer to perform the same operations as would be performed by the actual system c) the digitisation of a real speech for processing by the computer.	2 phase procedure: a) Language processor b) Library The functional make of the system can be described as follows: * system parameters: start time, run time, sample time, specification etc * Input data specification: parameters to be read in at execution time * Output specification (data to be printed or plotted) * Post processing (optional) * Statements defining a module (topological description).	a) Draw communication system as block diagram (each block must be one of the library module) b) number of each node and decide on the display routine nodes c) Input data	a) Heading, graph identifier, setting parameters of the run b) Input signal detail(prime signal source) c) Processing module detail (each with its own parameters), presented in the order in which they are to be executed d) Sequence terminated by END e) Modify, followed by detail of system module changes f) Input signal details, as before or a new one g) Run terminated by END
Data Structure	See sub-section (2.2.1)	See sub-section (2.2.1)	* System parameters: start-time, run time sample time, specification etc * Input data specification statement: parameters to be read in at execution time * Output specification (data to be printed or plotted) * Post-processing (optional) * Statement defining a modal (topological description) * see sub-section (2.2.1).	See sub-section (2.2.1)	Format are normally free, except for certain specifications under SIGNAL GENERATOR module. But all punched card information must start on column 1. Comment may be placed on each card, anywhere after the proper Labels: Example: See fig (A.1)
HOST machine		IBM 709 mod II	Univac 1108	ICL 4130	ICL4-70
Language	SAP	Fortran	Fortran	Algol 60	Fortran
References	26	21	16	11	35

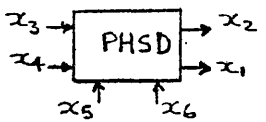
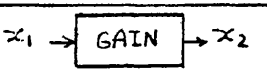
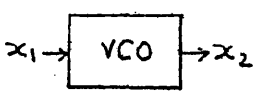
To generate a 1 kHz square wave, magnitude ±0.5; then NT = 1 ms and T = 3.91 μS for N = 256.

The fundamental has a magnitude of $2/\pi$ at frequency 1 kHz. Represent this by components of magnitude $1/\pi$ at frequencies ± 1 kHz. Hence to generate and observe the waveform:

```
COMMENT FOURIER SERIES TEST
TIME
3.91 E - 6
POINTS
256
SOURCE DATA
8 -2                    Generates seven harmonics
1 0.3183     90.0       Note conjugate symmetry about
                        zero frequency, giving real
-1 0.3183    90.0       resultant.
 3 0.1061    90.0
-3 0.1061    90.0
 5 0.0637    90.0
-5 0.0637    90.0
 7 0.0455    90.0
-7 0.0455    90.0
 SCOPE
 LP                     Displays time waveform of
 1 1                    one period on line-printer graph
 END
 END
```

Fig A.1 Example for SIGSIM

APPENDIX B The modules of the loop group of interrupts as implemented in the prototype model of ICOSS are summarised:

	Module	Code	Function	Input/Output	LCP	GCP	Comments
1	Signal generator	SIGN	X1=(A1+A2)sin(2π(f_1+f_2)t+(B1+B2)) X2= .....cos.....		CP_1 = f_2 CP_2 = A2 CP_3 = B2 CP4	f_s	CP_4: internal parameter for initiation and timing = (-1)
2	Multiplier, or phase sensitive detector	PHSD	X1=A1.X3.X5 X2=A1.X4.X6		CP_1 = A1	0	-
3	Gain (Amplifier)	GAIN	X2=A1.X1		CP_1 = A1	0	-
4	Filter 1 lowpass filter 1st order	FIL1	A=tan(π f_c/f_s) $G = \frac{1}{1+A}$ $H = \frac{1-A}{1+A}$ X2= G.X1 + G.XM1-H.XM2 XM2= X2 XM1=X1		CP_1 = f_c	f_s	Secondary CP_1=G " CP_2=H Theory see ref 5
5	Voltage control	VCO	X2=A1+A2.X1		CP_1 = A1 d.c. offset CP_2 = A2 d.c. gain	0	-

6	Filter 2: digital filter of any order type, or condition	FILT	See below(*)		see below (*)	f_s	-
7	Adder/summation	ADDR	X3=X1+X2		0	0	-
8	Switching	SWCH	X2=X1 if X1 < X3			0	-
9	Differentiator	DIFR	See below (+)		see below (+)	-	-
10	Branching, joining 2 nodes	BRCH	X2=X1		0	0	-
11	Double branching	JOIN	X2=X1 X4=X3		0	0	-

(*) Filter:

(a) This is a digital filter of any order (n), any type (Butterworth, Chebyshev), and any condition (high pass HP, low pass LP, bandpass BP, and band stop BS). The design procedure executed internally, utilises number of primary parameters supplied by the user to derive the coefficients of second order segments of a low pass filter, of the form

$$H(z) = A_o \prod_{i=1}^{i=n} \frac{1 + a_{i1} z^{-1} + a_{i2} z^{-2}}{1 + b_{i1} z^{-1} + b_{i2} z^{-2}}$$

For the other filter conditions, ie HP, BP and BS, a frequency shifting procedure[1] is applied and the second order coefficients are modified accordingly.

The primary control parameters as supplied by the user are:

CP1 = filter order (n)

CP2 = cut-off frequency (f_c)

CP3 = condition: 1 = BPASS, 2 = BSTOP, 3 = HPASS, and 4 = LPASS

CP4 = type: Butterworth, or Chebyshev (1 = Chebyshev, 2 = Butterworth)

CP5 = Centre frequency (F_{CEN})

CP6 = Delta for Chebyshev filter only

It should be noticed that the number of secondary control parameters is never equal to the number of primary control parameters, they may be less for second order filter but

always more.

Also a global control parameter (the sampling frequency f_s) is automatically utilised in the calculations.

(+) Differentiator

A wide band differentiator, as derived by Rabiner and Steiglitz [44] is adopted, which is made up of a series of second order segments, and having a transfer function:

$$H(z) = A \prod_{i=1}^{i=n} \frac{(1-z^{-1} a_{i1})(1-z^{-1} a_{i2})}{(1-z^{-1} b_{i1})(z-z^{-1} b_{i2})}$$

where n is the number of segments in the differentiator (or the order of it).

The characteristic of the differentiator depends on the order of the differentiator (n), and hence its choice dependent on the application.

In the present work only one segment is simulated, and if a higher order differentiator is required then the segments making up the differentiator will be treated separately, having in mind only one of these segments is supplied with the value of A, and the rest of the segments are supplied with 1.0 for their A's. Therefore, the primary control parameters for one segment are:

$$CP1 = A$$

$$CP2 = a_{i1}$$

$CP3 = a_{i2}$

$CP4 = b_{i1}$

$CP5 = b_{i2}$

APPENDIX C

The output display routines (ODR) as implemented in the present form of the flag group of interrupts, are summarised as follows:

ODR	Type	Function	Output
FFT	See below	(*)	
MEAN	a/b	$X_{MO}=\frac{X_1+X_2+\ldots+X_n}{n}$	X_{MO}/X_{MO} (I)

Notes

(i) a: the calculated value will be stored as a single value in (RESULT), which will be printed out and deleted in the TTY interrupt STOP

(ii) b: the calculated value will be stored in (YAXIS) in which case it can either be printed out as in (i) or plotted on a graph

AMPL	a/b	$X_{AO} = M_{ax}\ (X-X_M)$	X_{AO}/X_{AO} (I)
RMS	a/b	$X_{RO} = \sqrt{\frac{X_1^2+X_2^2+\ldots+X_n^2}{n}}$	X_{RO}/X_{RO} (I)

(iii) X: Result for condition (i)

(iv) X(I): Result for condition (ii)

(*) The FFT

(i) A limited number (n) of signal values (points) were accessed from the node (D) at which the FFT measurement is to be made. Only the number of points (n), and node number (D) are supplied by the user when the FFT request is made. These signal values, which represent the time domain response of the system at that node, are operated on by the FFT over a number of cycles. This is done in the following way:

(a) The FFT program package [52] was segmented into 5 portions.

(b) Each dedicated portion is treated as a separate "inner" flag interrupt.

(c) During the flag interrupt, and in the time allowed within the sampling period as outlined in the time domain, subsection (3.3) earlier, the ICOSS processor may execute the particular inner flag interrupt (segment) only once, depending on the time slot available. This is repeated for the same segment in the next

clocking periods until this segment is completely executed.

The method above gives rise to interactive execution of FFT, which is most suitable to real time simulation. This is the prime preason for its choice in ICOSS, over the alternative approach in which signal processing comes to a halt as soon as the correct number of samples are accummulated in order to perform FFT, as a time dependent TTY interrupt.

The FFT procedure's flow-chart is shown in Fig C1.

(ii) The result is given in type b only.

(iii) Theory: this can be found in numerous text books and papers

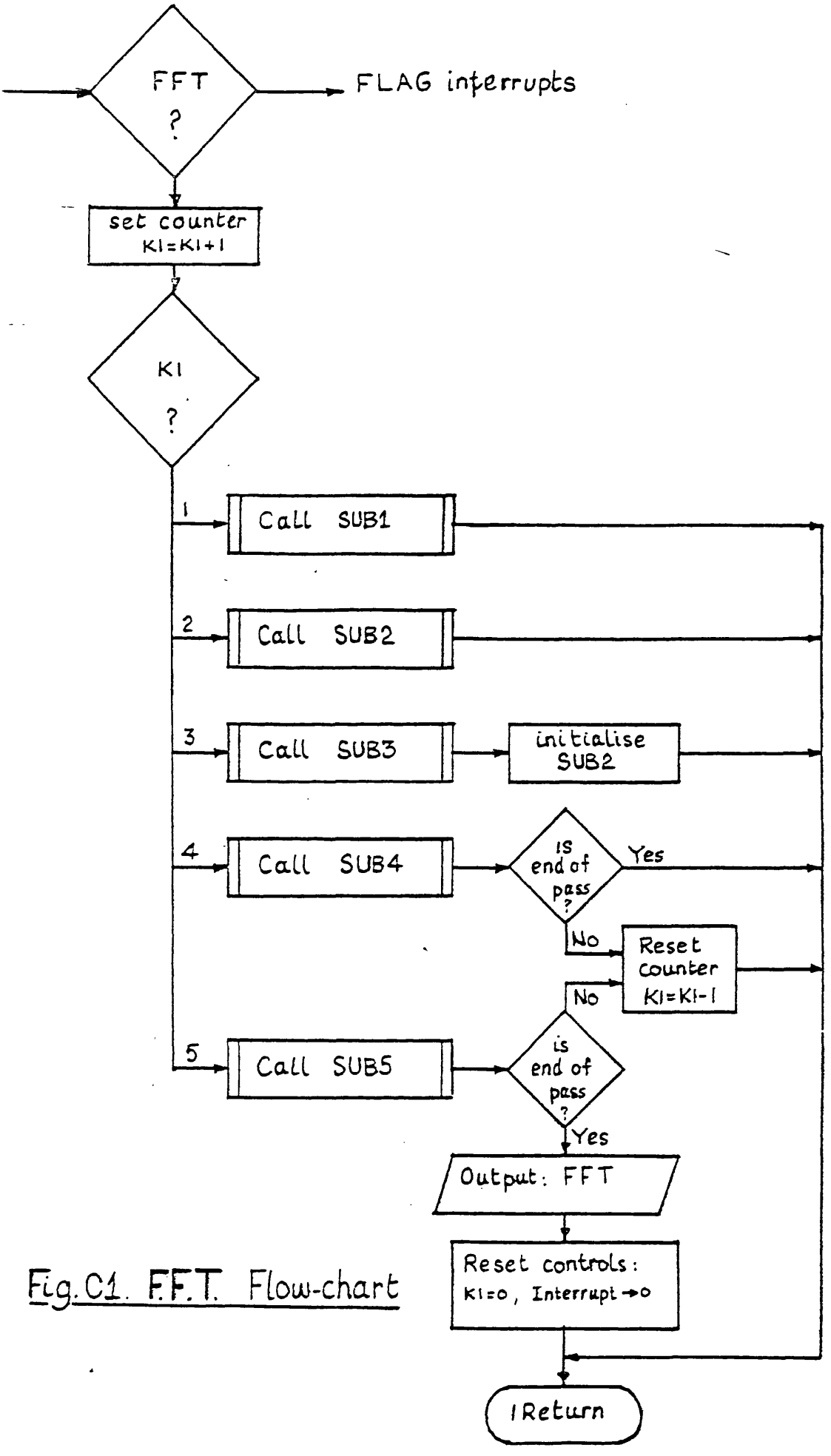


Fig. C1. F.F.T. Flow-chart

APPENDIX D

Subroutines and their descriptions used by the present prototype ICOSS. Fig 3.23 shows the inter-relationship structure of the various subroutines.

Subroutine	Description L loop interrupts T teletype interrupts F flag interrupts S supplementary routines
ICOSS	Main controlling program
LOOP	L_0 = The loop group of interrupts controlling program
SIGN	L_1 = Sinusoidal wave generator, complex signal output: $x = A_1\cos(\omega_1 t+B_1)+jA_2\sin(\omega_2 t+B_2)$
PHSD	L_2 = Phase sensitive detector (multiplier)
FILT 1	L_3 = Low pass digital filter of the first order (recursive)
GAIN	L_4 = Gain Amplifier
VCO	L_5 = Voltage offset controller for the voltage control oscillator
FILT	L_6 = Digital filter of any order, any type (LP,BP,HP and B stop), and any kind (Butterworth, Chebychev)
ADDR	L_7 = Adder module

SWCH	L_8 = Switching module
DIFR	L_9 = Differentiator
BRCH	L_{10} = Branching, joining two nodes
JOIN	L_{11} = Double branching
PHSH	L_{12} = Phase shifter - degrees
STORE	L_{13} = A signal values managements and storage routine for service of the flag group of interrupts at later stage
TTYFG	T_0 = Teletype group of interrupts controller
CONST	T_1 = Block diagram construction of the simulated system
EDIT	T_2 = Editing the block diagram structure of the simulated system
CHCP	T_3 = Change of parameters (local) - stage one
STOP	T_4 = The logging out, pausing, and results outputting interrupt routine
ENTR	T_5 = Supplying an already constructed block diagram of a simulated system to ICOSS via the TTYFG interrupt
COMSTR	T_6 = Block diagram presentation of the simulated system
CHGCP	T_7 = Change of global control parameters
PLTM	T_8 = Management of graphs plotting
INXM	T_9 = System initialisation

FLAG	F_0 = Flag group of interrupts controller
MANUP	F_1 = For the manipulation and management of the various flag interrupts
PROCES	F_2 = The Flag interrupts sequencer
FFT	F_3 = Fast Fourier Transform
SUB1	F_4 = Auxiliary routine for FFT
SUB2	F_5 = " " " "
SUB3	F_6 = " " " "
SUB4	F_7 = " " " "
SUB5	F_8 = " " " "
RMS	F_9 = Root mean squares determination
AMPL	F_{10} = Mean amplitude value
MEAN	F_{11} = Calculation of the mean value
CHCPFL	F_{12} = Changing of local control parameters - second stage
BCONST	S_0 = Block diagram construction: signal processing modules only
MOBRCH	S_1 = " " " : branching modules only
SCONST	S_2 = " " " : output display routines only
INSRT	S_3 = Insertion of new modules controlling routine, into an existing system
BCONSI	S_4 = Insertion of new modules: signal processing modules only

DELT	S_5 = Deletion of existing modules controlling routine
DELTØ1	S_6 = Deletion of existing signal processing modules
MODIF	S_7 = Array modification routine
SEARCH	S_8 = Table searching routine (characters)
SRCH	S_9 = " " " (numbers)
SHIFT	S_{10} = Shifting up (+1), and down (-1), of contents of an array or matrix
CPMAN	S_{11} = Local control parameters manipulations: directing ICOSS to accept the exact number of primary CPs, deducing secondary CPs, and constructing the pointer to the exact positions of these secondary CPs in their store (table)
FCONVR	S_{12} = Controlling routine for forward conversions of local control parameters from primary to secondary
FCONV1	S_{13} = Forward conversion of local control parameter for module 1 (FILT1 above)
FCNV2	S_{14} = " " " " " " " " 2 (FILT above)
CHEBY	S_{15} = Chebychev low pass digital filter: it computes the poles and zeros from order of filter, cut-off frequency and ripple[1]
BUTTER	S_{16} = Butterworth low pass digital filter: it computes the poles and zeros from order of the filter and cut off frequency [1]
BPASS	S_{17} = Band pass filter: frequency shifting operation to compute the new poles and zeros [1]

BSTOP	S_{18} = Band stop filter: frequency shifting operation to compute the new poles and zeros[1]
HPASS	S_{19} = High pass filter: frequency shifting operation to compute the new poles and zeros[1]
COEFF	S_{20} = The coefficients of the equivalent second order segment of the filter as deduced from the poles and zeros of it [1]
BCONVR	S_{21} = Controlling routine for backward conversion of local control parameters from secondary to primary
BCNV1	S_{22} = Backward conversion of local control parameters for module 1
BCNV2	S_{23} = " " " " " " " " 2
LOPDIR	S_{24} = Loop directive routine

APPENDIX E : Tables, stacks, arrays and matrices

(a) The following are the tables, stacks, arrays and matrices and their descriptions which have been used by ICOSS. The number of these tables can be reduced, but makes programming less flexible, and the reduction in the number of tables will not result in any significant reduction in memory size, since eliminating one table means enlarging another by the same amount.

Mnemonic	Type	Dimension	Description
SPMLIB	Characters	9	Signal processing modules library (modules definitions) directory.
SPCLIB	Integer	9	Signal processing modules local control parameters library (number of parameters which are required by the corresponding module).
SPMNOD	Integer	9,2	Signal processing modules input/output relationships (ie number of input nodes to number of output nodes of the modules).
SM	Integer	30,7	System matrix (as described in detail previously - subsection 3.5.3.

Mnemonic	Type	Dimension	Description
ODRLIB	Characters	6	Output display routine library (directory).
AXMLIB	Characters	4	Auxiliary modules library (directory).
TTYCOM	Characters	11	TTY interrupts commands (CONS, EDIT, etc) directory.
MODCON	Integer	2	Module numbers which require conversion of control parameters to secondary parameters.
CP	Real	100	A store, where the actual values of the local control parameters used in the system are kept.
GCP	Real	2	A store, where the actual values of global control parameters used in the system are kept.
X	Real	100,2	A store, where the processed signal at each node within the system is kept (temporarily for one sampling period).
XM	Real	32	State variable local storage.
LDR	Integer	30	Array containing the loop directive of the system.
STNLCP	Real	4	New LCP temporary storage from stage one to stage two in CHCP teletype interrupt.
STOSAM	Real	15 000	A store, where signal values (points) are accumulated ready for further processing in the flag group of interrupts.
FLM	Integer	3,5	Flag group of interrupts management matrix.

Mnemonic	Type	Dimension	Description
XAXIS	Real	512	Store for x-axis points for graph plotting procedure.
YAXIS	Real	5000	Store for y-axis points for graph plotting procedure.
Y	Real	1000,10	Store for y-axis points for multiple graph plotting procedures.
RESULT	Real	5	Temporary storage for the final results of the output display routines (mean value, amplitude etc).
ITEX	Integer	2	CPU execution time measurements; ITEX (1) = time so far taken by the program, ITEX(2) = time allowed by the CPU.
ALPHA	Real (complex)	30	Second order segments coefficients (the a's - see theory)
BETA	Real (complex)	30	Second order segments coefficients (the b's - see theory)
H	Real (complex)	200	The FFT procedure: complex signals storage area.
MFT	Real	20	The FFT procedure: internal coefficients storage area.

(b) The present content

For the present prototype version of ICOSS, the permanent (built-in) content of tables are:

```
SPMLIB: SIGN, PHSD, FIL1, GAIN, VCO, FILT, ADDR, SWCH, DIFR
AXMLIB: BRCH, SCOP, END, NSTR
ODRLIB: FFT, RMS, AMPL, MEAN, CHCP, TRAN
TTYCOM: CONS, EDIT, CHCP, STOP, RUN, ENTR, CMST, CHGC, PLTM, PLTG, INXM
SPCLIB: 4, 1, 1, 1, 2, 5, O, 1, 5
SPMNOD: 3, 4, 1, 1, 1, 1, 2, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1
```

APPENDIX F

LOGARITHMIC ARITHMETIC UNIT - MODIFIED

F1 Comparison between linear and logarithmic numbers[28]

Consider a signal value in digital filters being represented by a word having 16 bits, Fig F1. Assuming the word has one sign bit, five characteristic bits, and ten mantissa bits, then

* The dynamic range is: $2^{32} \equiv 192.66^{dB} \approx 193$ dB
* Since the smallest difference between two logs is $2^{-10} = 1/1024$, then each signal could be stored to an accuracy of $2^{1/1024} - 1 = 0.071$ %.

The absence of an extra zero in log encoding will seldom be important because of the large dynamic range; and, if necessary, a small amount of extra logic can be included in the arithmetic unit of the filter so as to treat the smallest encodable signal as if it were zero. Alternatively, the characteristic bits are reduced to four and remaining fifth bit is allocated for the exact zero. This will reduce the dynamic range to: $2^{16} \equiv 96.23^{dB} \approx 100$ dB.

Comparing a 16 bit logarithmic system with 0.071% accuracy, with a conventional 16-bit floating-point system with 0.05 to 0.1% accuracy, the log system has 36 dB greater dynamic range, coupled with the fact that the accuracy of the

floating-point system becomes progrssively worse over the lower 60 dB of its dynamic range.

Further consider having 11 bits word, with: one sign bit, five characteristic bits and five mantissa bits, then the dynamic range will still be 193 dB, but the accuracy will be reduced to $2^{1/32} - 1 = 2.19\%$.

Summing up:

(i) Advantages of logarithmic system over linear system:

* Greater dynamic range
* Multiplication of two numbers is easy (adding two logs)

(ii) Disadvantage

* Summation (or subtraction) is a complex operation; but can be simplified by approximation method.

F2 Logarithmic Arithmetic Unit LAU[27]

The purpose of LAU as proposed by Kingsbury and Kelly is to calculate $R = A.K + B$, where all of these numbers are encoded logarithmically to base 2. The LAU calculation is centered on:

$$\log_2 |R| = \log_2 |A.K + B|$$

and sign (R) = sign (A.K. + B)

The inputs to LAU are therefore:

$$a = \log_2 |A| \quad , \text{ sign (A)}$$
$$b = \log_2 |B| \quad , \text{ sign (B)}$$
$$k = \log_2 |K| \quad , \text{ sign (K)}$$

and the outputs from LAU are:

$$r = \log_2 |R| \quad , \text{ sign (R)}$$

as shown in Fig F2, and the detailed LAU diagram for 11 bit word is shown in Fig F3.

Note that each arithmetic operation amounts to single programming instruction.

F3 Bank of digital filters (BOF) requirements

The proposed BOF as described in Chapter 4, was based upon cascaded second order segments whose arithmetic operation on signals is of the form: a = b.c + d. Therefore the logarithmic arithmetic unit LAU is most suitable for this type of arrangement. However, a number of problems have to be resolved:

(a) The LAU as it stands is a 11 bits word unit, which may be satisfactory for the intended speech analysis;

however the word length may need to be increased to 16 bits say for higher accuracy as mentioned earlier.

(b) The input / output numbers of LAU are logarithmic numbers which must be encoded and decoded in a "buffer" zone between the microprocessor unit and the LAU, which amounts to a subset of the floating point in a representation in the host computer.

(c) The LAU operation must be adapted to the BOF microprocessor unit, in which

(i) each arithmetic operation is considered as a programming instruction

(ii) the interfacing problem between the two units is solved.

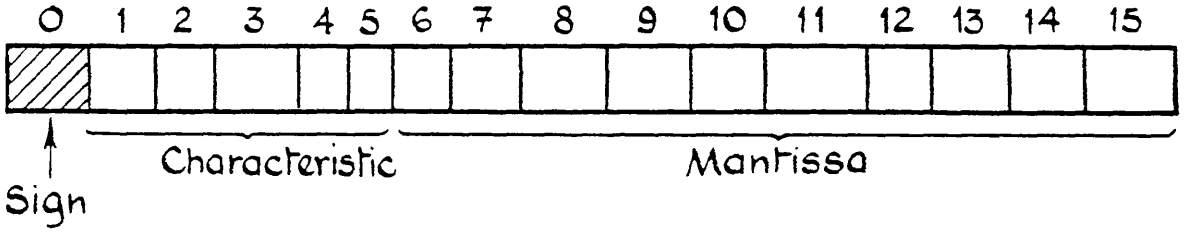

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Sign
Characteristic
Mantissa

Fig. F1

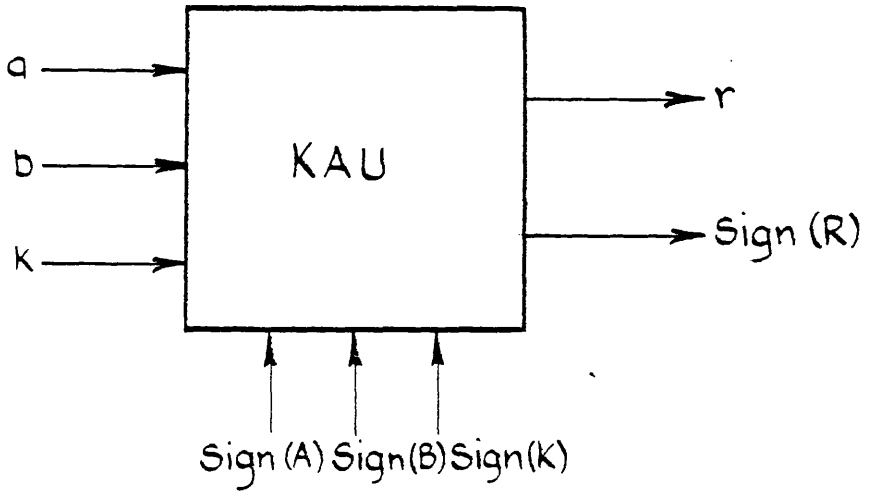

a
b
k
KAU
r
Sign (R)
Sign (A)
Sign (B)
Sign (K)

Fig. F2

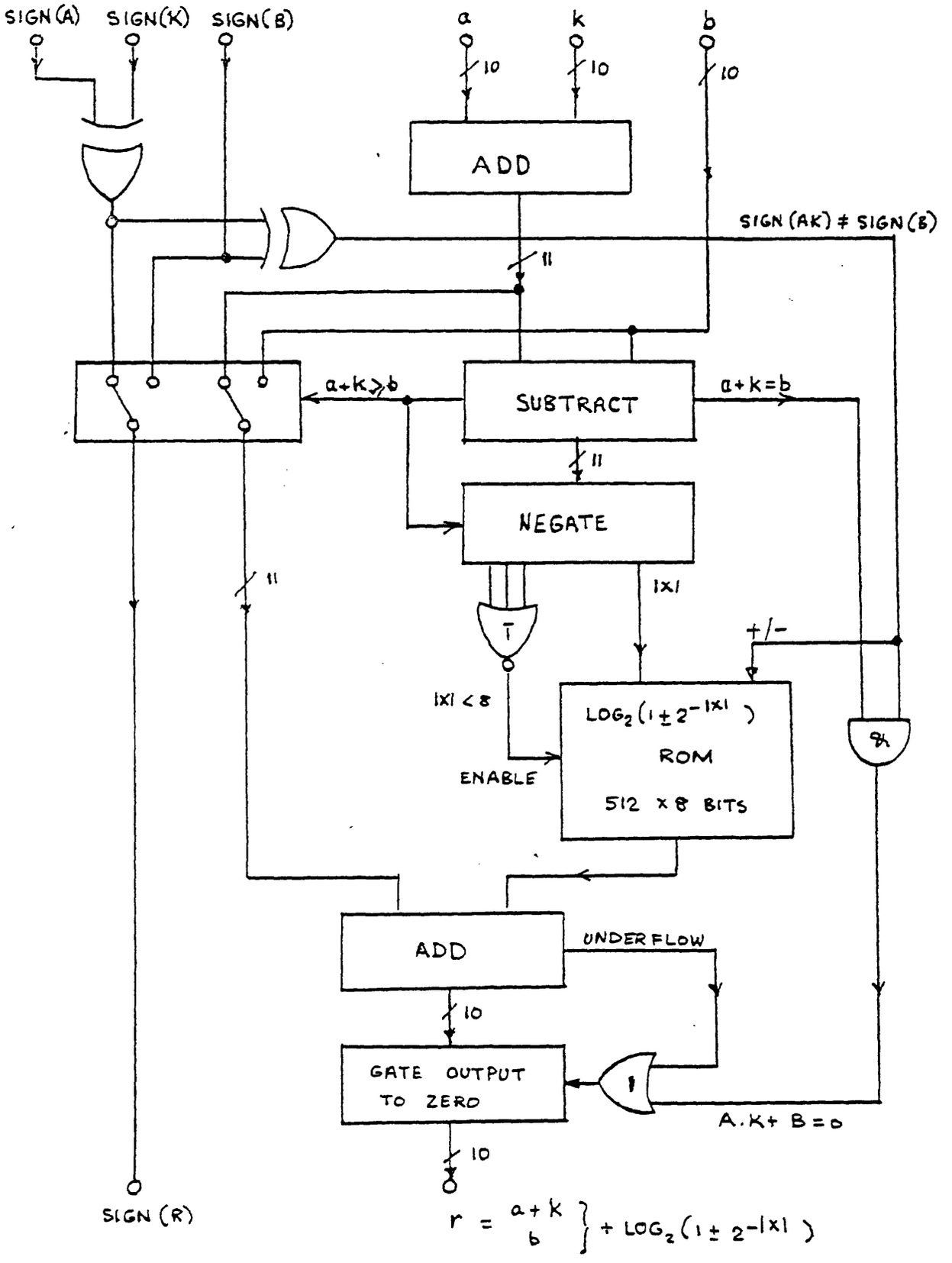


Fig. F3. Logarithmic arithmetic unit[27]

APPENDIX G Procedure for adding a new LOOP/TTYFG/FLAG interrupt

G.1 Preparations

* Choose an appropriate mnemonic for the new interrupt, using alphanumeric characters not exceeding four.
* Write a subroutine which performs this interrupt function.
* In the case of secondary control parameters are required in some of the LOOP interrupts (basic signal processing modules), two more subroutines have to be prepared, as follows:

  (a) Formulate the forward conversion and write subroutine FCNVN in the usual way

  (b) Formulate the backward conversion and write subroutine BCNVN.

G.2 Implementation

(a) In main controlling program (ICOSS):

Changes to be made as described in the table below, for all types of interrupts:

LOOP	TTYFG	FLAG
* Increase dimension by one for: SPMLIB(n_1); CPLIB(n_1); SPMNOD(n_1) * Add mnemonic at end of SPMLIB * Add number of LCPs as required by the new SP module, at end of CP * Add input / output node relationship as allocated by the user in module formulation, at end of SPMNOD * Increase NZ1 by one.	* Increase dimension by one for: TTYCOM(n_2) * Add mnemonic at end of TTYCOM * Increase NZ11	* Increase dimension by one for: ODRLIB(n_3) * Add mnemonic at end of ODRLIB * Increase NZ5

(b) For LOOP interrupt:

(i) In subroutine LOOP

```
      GO TO (1,2,....,NZ1), K1
      - - -
      - - -
NZ1   CALL SUBN (-------)
      GO TO 21
      - - -
      - - -
```

(ii) For the modules which need conversions to secondary control parameters

* In FCONVR:

```
      GO TO (1,2,....,NZ), IPOS1
      - - -
      - - -
```

```
NZ    CALL FCNVN (-------)
      GO TO 4
      - - -
      - - -
* In BCONVR:

      GO TO (1,2,....,NZ), IPOS
      - - -
      - - -
NZ    CALL BCNVN (------)
      GO TO 4
      - - -
      - - -
```

(c) For TTYFG interrupt

In subroutine TTYFG

```
        GO TO (1,2,....,NZ11)
        - - -
        - - -
NZ11    CALL TTYN (------)
        GO TO
        - - -
        - - -
```

(d) For FLAG interrupt

In subroutine PROCES

```
        GO TO (1,2,....,NZ5)
        - - -
        - - -
```

```
NZ5   CALL FLAN (------)
      GO TO
      - - -
      - - -
```

G.3 Comments

It is clear from the above discussions that the use of dummy interrupts will save some labour when adding a new ICOSS interrupt. However great care must be taken in the translation of these dummy interrupts into actual modules.

APPENDIX H

Phase-lock loop

1 Basic Structure[18,31]

The basic components of a phase-lock loop, shown schematically in Fig H.1, are the voltage controlled oscillator (VCO), and phase detector (PHSD), usually with a low pass filter. However, the attentuator is included in order to control the d.c. gain of the loop. The system works as follows: in the PHSD the phases of the input signal and of the VCO are compared, the output voltage of the PHSD passes through the loop filter (F), where eventually only the h.f. components are suppressed, to the control element of the VCO, and change its frequency, in such a way that the phase difference between the input signal and the local oscillator is reduced. When the loop is locked the average output frequency of the VCO is exactly equal to the average frequency of the input signal, ie for such output cycle there is one and only one cycle at the oscillator input.

2. Basic Equations

Input signal $V_i = \sin \theta_i$

VCO signal $V_o = \text{Sin } \theta_o$

PHSD output $V_d = K_d (\theta_i - \theta_o)$ the steady state situation, where

K_d = the phase detector constant (volts/radians)

Filter equation $\bar{V}_c = F(s)\,\bar{V}_d$

For a first order digital filter, having frequency cut-off (f_c) and an equivalent analogue cut-off frequency (angular ω_a)[5], the transfer function, using the digital filtering concept is

$$H(z) = \frac{z + 1}{z(1+1/\omega_a) + (1-1/\omega_a)}$$

Giving a difference equation:

$$Y_i = G\,x_i + G\,x_{i-1} - H\,y_{i-1}$$

$$\left.\begin{aligned} \text{where } G &= \frac{1}{1 + A} \\ H &= \frac{1 - A}{1 + A} \end{aligned}\right\} \text{ where } A = \tan\left(2\pi \frac{f_c}{f_s}\right)$$

The VCO operation $\frac{d\theta}{dt} = \omega_c + K_o\,V_c$

where ω_c is the free running frequency of the VCO

K_o is the VCO constant (radian/sec/volt)

Then the following parameters and properties can be deduced.

The filter transfer function:

$$H(s) = \frac{\bar{\theta}_o}{\bar{\theta}_i} = \frac{K\,F(s)}{s + K\,F(s)}$$

where K = loop d.c. gain in $\sec^{-1}$, or Hz = $K_o.K_d$

The natural frequency of the loop:

$$\omega_n = \sqrt{\frac{K}{\tau}}$$

The damping ratio of the loop:

$$\delta = \frac{1}{2}\sqrt{\frac{1}{\tau K}} = \frac{\omega}{2\ T\ \omega_n}$$

where τ is the LPF time constant

3. The PLL constants and loop parameters [18]

The PLL constants and parameters which are considered in this discussion, are those which have an important role in the behaviour of the PLL operation. They are the PHSD constant K_d, the VCO constant K_o, loop d.c., d.c. gain, loop BW, damping ratio, and the natural frequency.

3.1 Determination of the Phase sensitive detector constant K_d

If 2 sinusoidal signals of the same frequencies but differing phases are applied to the phase detector, then the output will contain a d.c. gain component, given by the following relationship:

$$E_d = K\ \frac{A_i \cdot A_o}{2}\ \cos\ (\theta_i - \theta_o)$$

where $e_i(t) = A_1 \cos\ (\omega_c t + \theta_i)$

$$e_o(t) = A_o \cos(\omega_c t + \theta_o)$$

and K is the gain (loss) of the phase detector.

Let $(\theta_i - \theta_o) = \theta$, then $K_d = K \frac{A_i A_o}{2} \cos(\theta)$.

The phase detector constant is defined as:

$$K_d = \left.\frac{dE_d}{d\theta}\right|_{\text{for } \theta = \frac{\pi}{2}}$$

$$= -K \frac{A_i A_o}{2} \sin\left(\frac{\pi}{2}\right) = K \frac{A_i A_o}{2}$$

If the frequency of on lock input to the phase detector is changed, then the low frequency component of the output is given by:

$$e_d(t) = K \frac{A_i A_o}{2} \cos((\omega_i - \omega_o)t + (\theta_i - \theta_o))$$

It can be seen that the amplitude of this component is equal to the phase detector constant K_d.

3.2 Determination of the VCO constant K_o

The VCO constant is determined, by plotting "the output voltage" vs applied voltage (d.c.) and taking the slope of the curve at the required centre frequency, as shown in Fig H.2.

3.3 PLL range of operations

(a) Lock range (or tracking range)

The frequency range over which phase lock is maintained

$$F_L = \frac{K}{\pi}$$

where $K = K_o K_d A$ = d.c. loop gain

where A = the gain (loss) of the attentuator.

(b) Capture range (or acquisition range)

The difference between the maximum and minimum frequencies at which the loop just comes into lock

$$F_{ca} = \frac{1}{\pi} \sqrt{\frac{K}{\tau}}$$

F_L and F_{ca} are related as shown in Fig H.3.

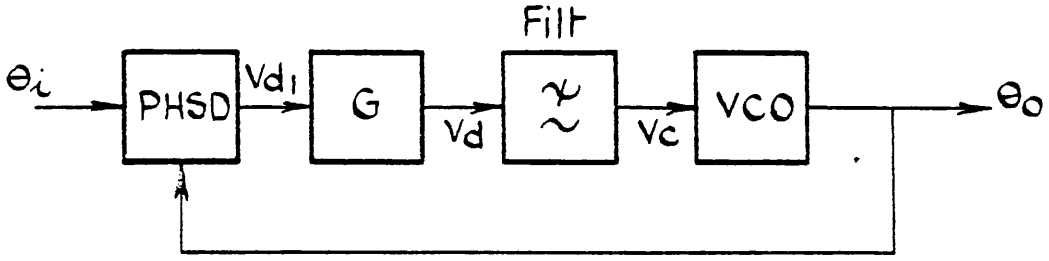


Fig. H1. PLL Basic Structure

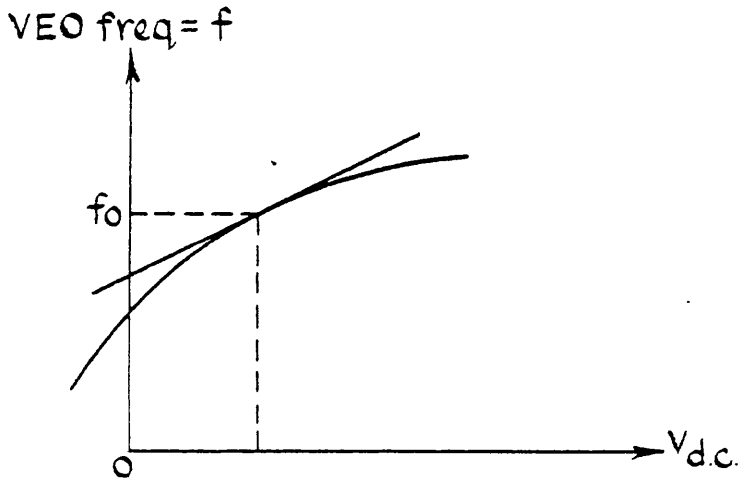


Fig H2 Determination of Ko

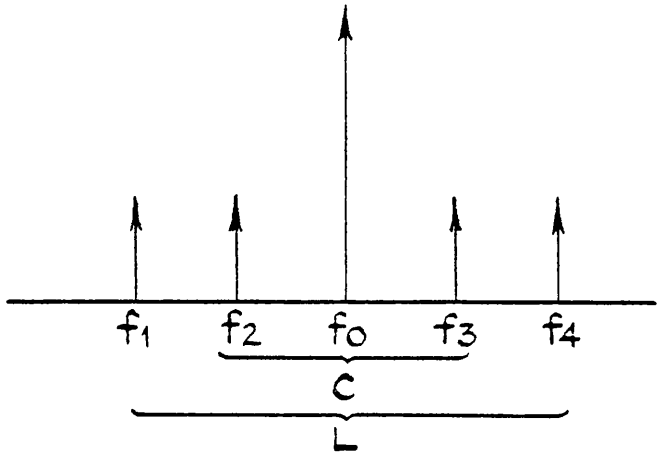


Fig H3. Lock & Capture Range - Relationship

APPENDIX I

COMPUTER RUN TIMES

During the process of testing the simulator, some measurements were made for the computer run times of some of the problems already mentioned, which may be found useful for future work.

I.1 On-line operation, using PDP8

For a system configuration shown in Fig (2.8.1), with the signal processing module being a second order low pass digital filter Chebychev type, the maximum sampling frequency achieved was 15Hz. Due to the slow execution of arithmetic functions (multiplications etc) in Fortran II, which may be improved using a floating point processor.

I.2 Off-line operation using large size computer ICL 2980

(a) For the co-channel interference problem: Section 5.3.
In a test to obtain the graph of Fig 5.3.2, the total run time was 29.5 seconds. The run involves 5 graphs, each with 500 points. The time is sub-divided as follows:

(*) 10 seconds for managements (TTYFG operations) eg System construction, change of parameters etc.

(*) 19.5/2500 = 7.8 msec, the time taken for one sample to be processed.

Therefore for real time situation using this type of computer, the maximum clocking frequency is 128.2 Hz, for this problem.

(b) Fast acquisition problem: section 5.4.
In a test to obtain the graph of Fig 5.4.5, the

total run time was 35 sec. The run involves 2 graphs, each with 500 points. The time is subdivided as follows:

(*) 10 sec for managements (TTYFG operations)

(*) 25/1000 = 25 msec, the time taken for one sample to be processed.

Therefore for real-time on-line situation using the 2980 computer, the maximum clocking frequency = 40 Hz.