Embedded Call Billing for EPABX Systems

PROJECT REPORT 1996 - 97

Submitted By

C. SWAMINATHAN

S.N. RAMESHKUMAR

C.S. SRINIVASAN

MATHEW GEORGE

P-1321

Under the guidance of

Mr. N. SANJEEVI RAMANATHAN, M.E.,

Submitted in partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF ENGINEERING

in ELECTRONICS AND COMMUNICATION ENGINEERING

of the Bharathiar University, Coimbatore

Department of Electronics and Communication Engineering

Kumaraguru College of Technology

Coimbatore - 641 006.

Department of Electronics & Communication Engineering

Kumaraguru College of Technology

Coimbatore - 641 006.

CERTIFICATE

This is to certify that the Report entitled

EMBEDDED CALL BILLING FOR EPABX SYSTEMS

has been submitted by

Mr. C. [illegible], [illegible]
C. S. SRINIVASAN, [illegible]

In partial fulfillment of the requirements for the award of Bachelor of Engineering in Electronics and Communication Engineering Branch of Bharathiar University, Coimbatore - 641046 during the academic year 1996 - '97 .

[illegible]

GUIDE

[illegible]

HEAD OF THE DEPARTMENT

Certified that the candidate was examined by us in the project work Viva - Voce examination held on [illegible] and the University Register Number was 9327D0253 [illegible]

[illegible]

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We express our gratitude to our respected Principal ***Dr.S.Subramanian*** B.E., M.Sc(Eng.)., Ph.D. for providing the encouragement and for the facilities provided for our project.

We are thankful to our beloved ***Prof.M.Ramasamy*** M.E.,MISTE,MIEEE(USA),MIE,C(Eng.), Head of the Department, Electronics & Communication Engineering for being a paragon of inspiration and unflagging help.

We express our heartfelt gratitude to our guide ***Mr.Sanjeevi Ramanathan*** M.E., who was the leading light behind the success of the project.

We are extremely grateful to ***Mr.R.G.K.Pai,*** AGM(R&D) BPL TELECOM LTD., for his assistance.

We are greatly indebted to ***Mr.C.N.Ramakrishnan*** ,R&D Engineer, our guide at BPL TELECOM LTD., for all his encouragement, support and guidance.

We reserve our special thanks to our Parents, classmates and all other faculty members, non-teaching staff for their perseverance and co-operation.

CONTENTS

CHAPTER - 1

ABSTRACT

ABSTRACT

The tireless pursuit for better communication has led to the evolution of Private Telephone Networks. Large organisations have installed EPABXs requiring them to have the highest degree of sophistication. Provision for call management is an absolute necessity in these systems. Efficient call management calls for effective call metering. If these were to be accomplished with minimum expenditure, then it would be certainly appreciated. Our project by incorporating a call billing procedure inside the EPABX controller aims at this pinnacle of satisfaction.

CHAPTER - 2

INTRODUCTION

INTRODUCTION

Introduction

Rapid growth in technology especially in the field of telecommunications has spawned a host of advancements. The concept of an electronic telecom exchange was always a necessity keeping in mind the intricacies involved in a mechanical switching centre.

Communication within an organisation is pivotal for its growth. To facilitate this, a revolutionary concept called the EPABX was introduced. This helped the people within the organisation to effectively communicate not only among themselves but also with the outside world.

Modern EPABX s possess a variety of features like call forwarding, fax homing, and call logging. Due to the heavy volume of voice traffic in a typical organisation there is a need to record the traffic of private exchanges. Hence call management becomes necessary.

Call management should provide the exchange operator with the information on the dialled number, extension number and trunk number. It is expected to give the duration of the call and also the cost of the call. Billing can be done either on line or off line though the later is widely followed in organisations. A record of calls is available to the operator when he wishes to retrieve it.

This project aims at providing call billing as a part of the whole EPABX system. The usage of separate add on cards or of computer based software is avoided completely. The elimination of cumbersome hardware and expensive application software has been a primary objective of this project.

SPECIFICATIONS

The specifications of this project are induced by the customers' needs and the limitations implied by the currently used EPABX system. PRODIGY x0y, manufactured by BPL TELECOM , is used by us.

Call billing as such would require the duration of the call and the corresponding pulse rate. For this a list of STD & ISD codes and their corresponding pulse rates is needed. This would be taxing the limited memory space available inside the microcontroller. Hence an optimum number of code details are made available so that they are neither a burden on the system memory nor are they insufficient to make the billing inaccurate.

The software should be made available inside the controller. The High Level Language application software is ruled out since the user may not be able to afford the service of a PC or be able to get the application software without loosening his purse strings. The object code should be available in the micro controller . The usage of cross assembler which converts the HLL program to assembly is also not considered as it takes roughly triple the memory the assembly program written directly would occupy .The only option left is to write directly in the assembly language .

This project is written to satisfy the user needs . The user is allowed to program his selection of frequently used codes . We provide a list of codes which the user can retain or he can change them on his preference .

Pulse rates are not the same for each code at all places. These differ according to the proximity of the destination . The program leaves this option to the end user. This feature rather being a limitation adds to the versatility of the software used .

The programmer has to ensure that his software should adjust for the condition when the code the subscriber dials does not fall under the 180 codes he has stored .The software should provide the maximum closest approximation for the cost in this case.

SCOPE

The project is the basic step in providing prolific call management services. It is the foundation for services like call budgeting, call limiting and similar other services. These offshoots of call billing can be implemented with little cost in the further developments of EPABX systems.

CHAPTER - 3

EPABX - A GLANCE INTO THE SYSTEM

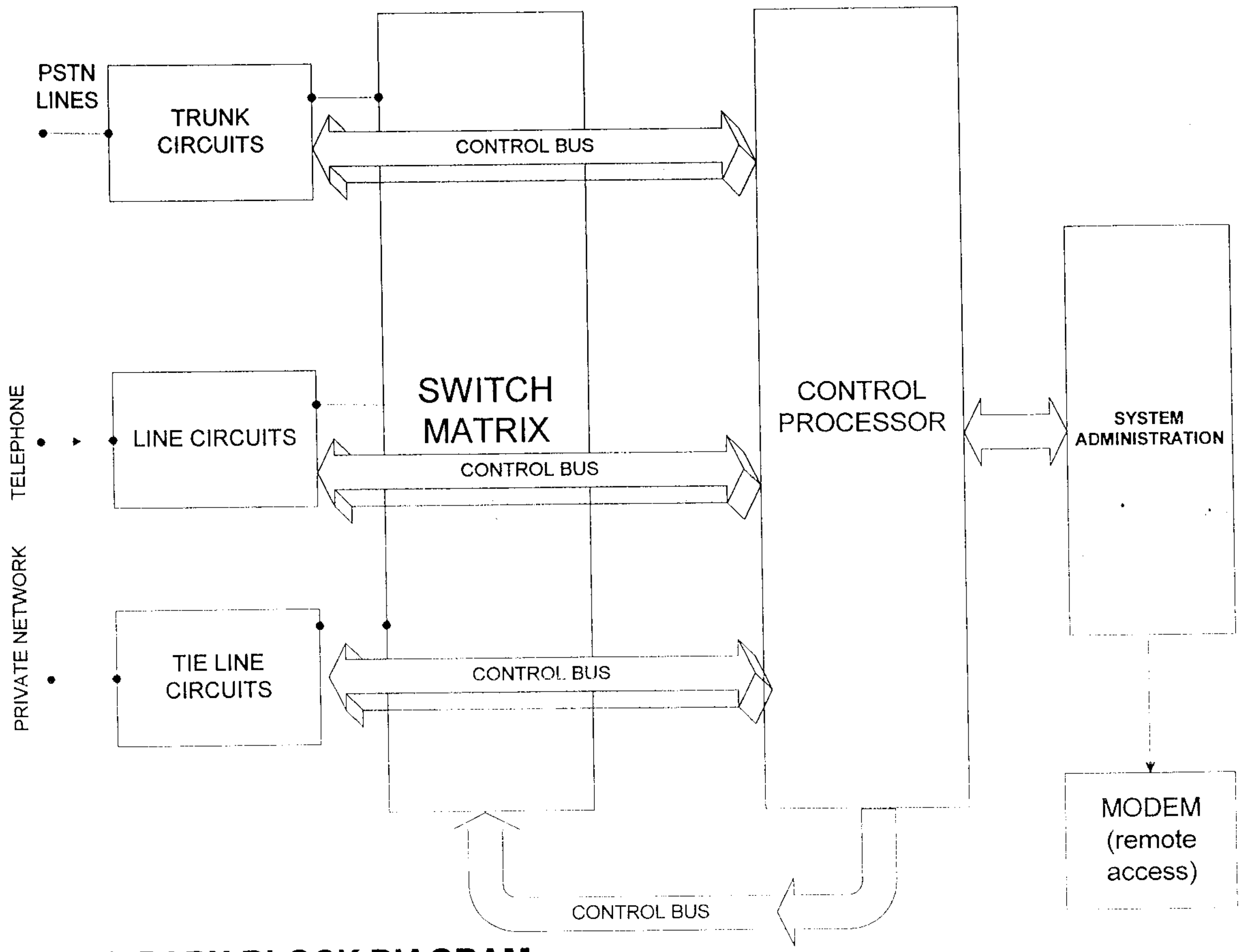


FIGURE-1 PABX BLOCK DIAGRAM

THE EPABX - A GLANCE INTO THE SYSTEM

PURPOSE OF EPABX SYSTEMS

The EPABX is an acronym for Electronic Private Automatic Branching Exchange. Prior to the existence of PABX s , a company would have a separate telephone line from the public exchange for each employee requiring the telephone. If a call came into the wrong telephone then either the caller would have to call back to the correct telephone or the correct person would have to physically come to the called telephone. Also the company would have to pay for every telephone line used. But with an EPABX, incoming calls can be answered by one person and then the call transferred to the correct person.

ARCHITECTURE

The figure shows the main elements of an PABX system. The basic block include

1. Control system
2. Line and Trunk circuits
3. Switching matrix
4. Modems
5. Cross connect frames
6. Telephones

The system control provides all the operational software/control functions to interconnect the various functions.

The trunk circuits provide the basic access to and from the public network and can be analog or digital connections.

SWITCHING SYSTEMS IN EPABX

Early switching systems were manual and operator oriented. Their limitations led to the development of automatic exchanges. The most common among them being the Strowger switching system. Here the control functions are performed by circuits associated with the switching elements in the systems. Crossbar systems have hardware control subsystems using latches and relays. The main problem was the modification of it's functions. Hence electronic switching systems were developed where the control functions are performed by a processor or a computer. These were called stored program control systems. The advantage was that new utilities could be easily incorporated by changing the control program.

Stored Program Control(SPC), a method adopted in EPABX is centralised i.e., all the control equipment is replaced by a single processor capable of a high processing rate. The processor has access to all the exchange resources like scanners, distribution points and executes all the control functions.

FUNCTIONS OF A CONTROL SUBSYSTEM

Event monitoring, Call prócessing, charging and operation & maintenance are the four vital functions of control subsystem in an exchange. Event monitoring possesses the highest real time constraint while O & M and charging, the least. To maintain priority, vectored interrupts are provided so that interrupting source supplies the branch address information to the processor.

OPERATION & MAINTENANCE (O&M) FUNCTIONS

1. Administration of exchange S/w and H/w.

2. Changing information in translation tables.
3. Charge according to subscriber class of service.
4 Introduce a new line/trunk into operation.
5. Supervise exchange facilities.
6. Traffic monitoring.
7. Locating faults and errors.

P - 32

These complex functions are usually handled by a dedicated processor.

S/W ARCHTECTURE IN AN EXCHANGE

Processing a call is an event oriented processing function. An event occurring at a subscriber line or trunk activates an event. Setting up a call is not performed in one continuous processing sequence. Rather it involves several elementary processing actions separated by a waiting duration for external events. In effect, many calls are processed simultaneously with each call being handled by a different process which is in effect a program execution.

To prevent any process from dominating the CPU, a timer is set and if it runs out, then the process is forced to the ready state and joins at the end of the order of the process list depending on its priority.

For example, If a calling subscriber upon picking his handset does not dial within a particular period of time, an error tone is given thus preventing the process from taking control of the CPU for a long time.

Each process has a control block containing information about the processes like,

- Current state of the process.
- Process priority and processor scheduling parameters.
- Register area to save the contents of CPU registers when an
- Interrupt occurs
- Memory allotted to the process.
- CPU time limits, process number etc.
- Event status and I/O resources associated with the process.

Information about the resources of the exchange and their current utilisation is kept in the form of tables. For each subscriber line, a subscriber line state table contains information about whether it is ON hook or OFF hook.

Temporary data are created and modified during call processing thus defining the dynamic state of resources and links. They formulate a call related data stored in the working area of the process.

The Main state description tables are:

1. Subscriber line state table.
2. Terminal circuit state table.
3. Switching n/w link state table.
4. Working area of a process.

Working area of a process holds call progress information, digit dialled, class of device information for calling and called subscriber, data from routing tables etc. The working area's life is synonymous with that of the process.

Call processing is handled by many S/W processes that are initiated and terminated for every call by a main system control process. Periodically scheduled processes like scanning of all the subscriber lines every few ms, checking for OFF hook conditions are related to call processing. When a telephone is OFF hook, main control process senses it and orders another

process to handle all the functions associated with calling line ,like checking whether subscriber has a decadic, DTMF, push-button or a rotary dial telephone, allot a digit receiver and monitoring the dialled digit. When all the digits are collected a process performing digit translation and routing is initiated. Then switching n/w links are set-up. In addition information for call billing is also gathered.

FUNCTIONS OF A SWITCHING SYSTEM

1. Event monitoring
2. Call processing
3. Charging
4. O and M

Events occurring at the line units, trunk junctions inter-exchange signalling are monitored by control subsystem. The occurrences of the events are signalled by relays. Control subsystem operates relays in the junctors and line units commanding them to perform special functions.

When a subscriber goes OFF hook, the event is sensed through the hookstatus (HS-active low) line. The calling location is determined and marked for dialtone and register finder seizes a free register. Depending on a pulse or multi-frequency dialling ,a register is chosen which sends out the dialtone to the subscriber. When the initial digits are received, they are passed onto the initial translator for processing and the register receives remaining digits.

The line circuits provide access to the private network interconnecting a number of PABX s directly or indirectly via tandem private exchanges. Line circuits provide the basic BORSCHT functions for analog telephones.

The switching matrix can be either CMOS analog switches for the smaller systems and digital switching is the most cost effective in nearly all other designs.

METER PULSE DETECTION

On any two wire analog signaling systems the PABX may also be required to detect meter pulses sent from the exchange. These are normally only used on systems where it is necessary to provide call accounting within the PABX.

Two signaling systems are in use for meter pulse detection and both use out of band signaling as it is necessary to send the meter pulses to the PABX during the conversation phase of the call and the talkers should not be able to detect the pulses. 50Hz longitudinal pulses are applied to the speech pair and the trunk circuit of the PABX must therefore be equipped with a detector which senses the meter pulses and then passes them onto the system controller for assimilation with the call record. The other method uses 12kHz or 16kHz signals.

THE BPL PRODIGY EPABX

The BPL model PRODIGY x0y is a mini EPABX catering upto y c.o lines and x subscriber extensions. It has all the productivity improving features comparable to any higher end system, but with optimum hardware costs to ideally suit offices having small to medium capacity.

The system is housed in an attractive ABS plastic cabinet designed for wall mounting. The system operates on 50Hz, 220V AC power and has an in built fuse, varistor gas arrestor protection at various circuit levels.

The system consumes less power than a 25W bulb and employs an SMPS for generating the different voltages required for system operation.

The system uses the 8051 microcontroller as the CPU, with stored program control and employs CMOS space division switching.

The system has an in built health monitoring software to check the sanity of the system which is indicated through an RUN LED. The system utilizes HC & HCT series digital IC's for better reliability and low power consumption.

The system is designed to operate with or without the need of operator console to suit varying customer requirements.

The system accepts rotary as well as push button telephones of decadic or DTMF type. It provides for data entry a standard telephone instrument so that programming of important features can be done at site by users themselves.

The system is provided with SMDR facility to keep a track on the outgoing calls. It weighs 2.5 kgs without cables, extension instruments and packaging.

FEATURES AT A GLANCE

SYSTEM FEATURES

1. y c.o lines and x extension lines.
2. Music on hold.
3. Distinctive ringing for trunk and extension calls.
4. DTMF/PULSE dialing.
5. Incoming trunk c.o line queuing.
6. Direct outward dialing.
7. Subscriber Message Detail Record (SMDR) with buffer (1500 calls).

8. Day and night service mode - Auto & manual.
9. Customized data entry from telephone.
10. Cyclic hunting of trunk.
11. Diagnostic check for trunks.
12. Programmable flash timings.
13. Parallel printer interface.
14. Trunk classification.
15. Cyclic ringing for incoming trunk calls.
16. Hardware reset for data entry.
17. Power failure transfer.
18. Paging interface.
19. Console interface (serial interface)
20. Faulty receiver isolation.
21. Call overflow to external line.
22. Line and c.o line status display.
23. Wall mountable.

EXTENSION FEATURES OF PRODIGY

1. Intercom
2. C.O line transfer
3. Follow me
4. Intercom call transfer
5. Do not disturb
6. Automatic call back on busy extension
7. Automatic call back on specific trunks
8. Hotline to trunks
9. Hotline to specific trunk
10. Automatic call back on busy C.O line
11. Call pickup (single digit)
12. Dial call pickup
13. Call forward on no answer
14. Call forward on busy
15. Call consult
16. Call camp on
17. Specific trunk access
18. Automatic call back on STD trunk
19. Call privacy
21. Last call recall
22. Pad lock facility for trunk access with 4 digit password

CHAPTER - 4

THE 8051 MICRO CONTROLLER

THE 8051 MICRO-CONTROLLER

The Intel 8051 micro-controller belongs to the MCS-51 family. it is an 8-bit MCU, having a 4K internal ROM. It is also used as an Universal Asynchronous Receiver/Transmitter (UART). The features of 8051 are,

* 8-bit CPU optimized for control applications
* Extensive Boolean processing(single-bit logic) capabilities
* 64k Program memory address space
* 64k Data memory address space
* 4kB of on-chip Program memory
* 128 bytes of on-chip data RAM
* 32 bi-directional and individually addressable I/O lines
* Two 16-bit counters/timers
* Full duplex UART
* 6-source/5-vector interrupt structure with two priority levels
* On-chip clock oscillator

8051 - HARDWARE DETAILS

Here we shall discuss in brief about the various features of 8051 architecture ,which we shall be using frequently in this project implementation.

SPECIAL FUNCTION REGISTERS

A part of 8051 memory is allocated for special function registers., not all of which are occupied. The important SFC ,the accumulator is the ACC register. The B register is used for multiplication and division operations or as a scratch pad. The stack pointer is 8-bit wide. The DPTR stands for Data Pointer, consists of DPH & DPL. The function intended is to hold a 16-bit address. It can be manipulated as a 16-bit register or as two 8-bit registers. The serial data buffer consists of transmit and receive buffer registers.

TIMER/COUNTERS

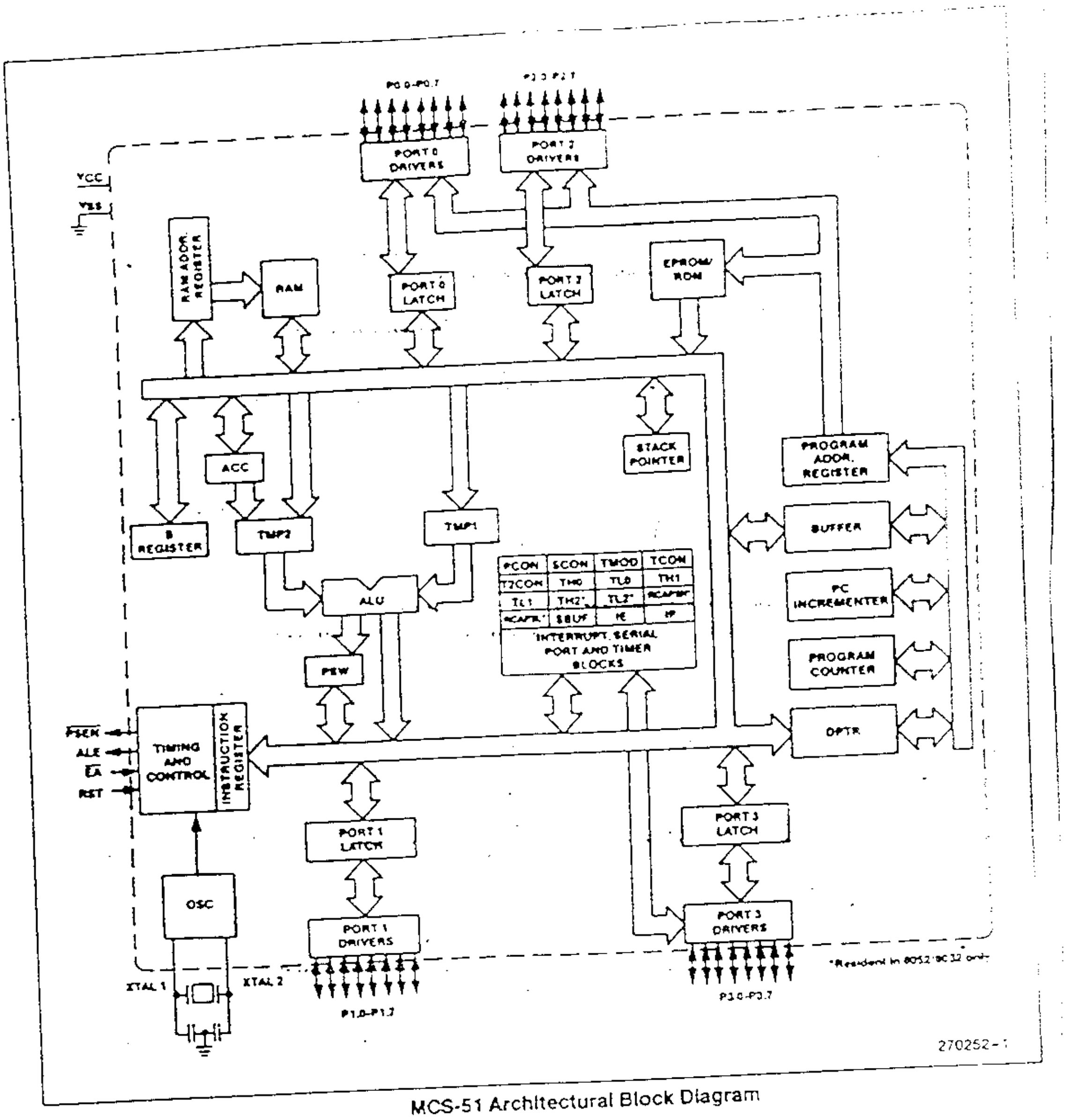
The 8051 has two 16-bit Timer/Counter registers, Timer-0. Timer-1. In timer function the register is incremented every machine cycle i.e., for 1/12th of the oscillator frequency. In the counter function the register is incremented in response to 1 to 0 transition at its corresponding external input pin. These are 4 modes of operation.

INTERRUPTS

There are 5 interrupt inputs to 8051. The external interrupts are INT 0, INT 1 (BOTH ACTIVE LOW). The service routine is vectored to by the hardware, only if the interrupt was transition activated. The Timer-0 and Timer-1 interrupts are generated by TF0 and TF1. When a timer interrupt is generated the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to. Interrupts can be generated or pending interrupts can be cancelled in the software.

REGISTERS

The registers are available as register banks (4 x 8 banks). There are also bit addressable registers. We use both of these in our project apart from the special function registers (SFC).



MCS-51 Architectural Block Diagram

CHAPTER - 5

SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

EXISTING CALL METERING METHOD

In the existing model of the EPABX system used , we get an out put which includes various details of the call made through the exchange. These include the dialed number , the extension number , the duration of the call, and the type of call . The cost of the call is not given.

The dialed number for each of the call made is stored in a specific location which is available for each of the extensions. The number can pertain to a local exchange or a trunk code. The details pertaining to an extension are stored in a HOOKBLOCK. In order to obtain the details about a particular extension we can obtain it by specifying the address of HOOKBLOCK.

The details of calls made from all subscribers of the EPABX are stored inside the memory of the controller used. The **SMDR** (Station Outgoing Message Detail Record) stores all the details of a call transaction.

PROPOSED CALL BILLING METHOD

The main drawback of the existing method is that it requires a PC to perform the metering functions. For economic viability, it is preferred if the codes and pulse rate are stored within the EPABX itself. Our method aims at alleviating the problem in the existing method by implementing the software using 8051 micro controller. Thus access time is shortened leading to a better billing.

In the proposed implementation,150 STD codes and 30 ISD codes can be stored in the program memory. These codes are completely user programmable and can be modified at any time thus giving unrestricted flexibility to the user. A hard copy of the details of the call can be obtained like call duration, pulse rate and the cost of the call. The extension wise print out is also possible, thereby facilitating the monitoring the use of telephone by each extension.

Further intelligence is added to the procedure. If the cost of the calls made by the subscriber exceeds a particular limit as set by the programmer then calls can be disabled to that subscriber. The purpose of call management is envisaged by call limiting. An extension wise printout facility is also provided. Default pulse rates for local calls and trunk calls is included in the software.

```
PRODIGY EPABX VER 1.5, DATED 04/04/1997.
--------------------------------------------------------------------------------------
Ser.|Ex|   DIALLED    |  DATE  |  TIME    | Mts. | Rs.Ps.|  TYPE    |FROM|Jn.
No. |No|   NUMBER     |        |          | Sec. |       |          |EXTN|No.
--------------------------------------------------------------------------------------
0001 48 03341236       08/03/97 10:49:00 001:26  064.50 DIRECT           06
0002 37 049174321      08/03/97 14:43:00 002:44  123.00 DIRECT           06
0003 44 0373623        15/08/97 16:37:00 007:50  119.70 DIRECT           06
0004 37 3512           09/03/97 10:02:00 000:20  002.10 DIRECT           06
0005 48 02267412       15/08/97 06:32:00 001:02  021.00 DIRECT           06
0006 44 03384231       15/08/97 05:22:00 001:00  014.70 DIRECT           06
0007 37 0522716943     06/07/97 16:28:00 001:28  023.10 DIRECT           06
0008 44 05226473       06/07/97 12:00:00 001:27  021.00 DIRECT           06
0009 48 0491533134     09/03/97 07:38:00 000:52  004.00 DIRECT           06
0010 44 0491533134     09/03/97 07:40:00 001:14  006.00 TRANSFER      48 06
0011 37 0491533134     09/03/97 07:41:00 001:44  008.00 TRANSFER      44 06
0012 44 0112346        09/03/97 07:49:00 002:57  029.00 DIRECT           06
0013 44 02269732       09/03/97 07:55:00 003:20  033.00 DIRECT           06
```

SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

1.MAIN OBJECTIVE

Our basic aim is to provide the user with a billing facility which gives him the information about the call duration and the charge for it.

So far in the existing models we can get the information only about call duration and not the call charge. So we have to add this facility. So we have to develop a logic of how to make this cost calculation.

Note that this call calculation must be made for local, STD and also for ISD. So we need make the user have billing of calls to places which is in his list.

So our first step in the development of SRS is to give the user the call charge with duration.

2.PRINTING OF BILL DETAILS

Now our next step is to print the entire bill details which the user would require. We have to decide what are the fields he requires in the bill printout.

Basically the printout of the call bill must have the following,

a. Serial number

b. Extension number

c. The dialed number

d. The date on which call is made

e. Duration

f. Type of call

g. Junction/trunk number

i. The cost of the call

So in our billing output the last field i.e., the cost of the call is very important.

Now it is how we are going to manage all these fields. We have to allow the user to choose the fields of his requirements from the above said fields, as per his choice.

3.ENTERING OF STD CODES, PULSE RATE

Now the important point is how to enter the STD codes, the pulse rate. This is an hectic problem, as we are left with only 150 locations for which we can enter the codes. Added to this, one user may require a particular group of cities and another a different group.

As per statistical study of STD codes ,

* There are about 2700 STD codes in India.

* Most of the STD codes are either 4 or 5(mostly)

* There are also places having 7 or 8 digit codes.
* The starting digits are from 1 to 8.

So we can leave it to the user to enter the codes (through touch telephones) and the corresponding pulse rates. Provisions must be provided to include 7 digits.

4.ENTERING OF ISD & MULTIPLICATION FACTOR

As far as ISD codes are concerned all of them have two leading zeroes. Also all of them have pulse rates between 1.2 to 2.0. So we can round off the pulse rates of all ISD codes as 2.0.

Regarding the multiplication factor we shall leave it to the user to enter as per his wish any factor<10.

5.PRINT OUT OF ENTERED STD/ISD CODES

Now say that the user has entered the STD and ISD codes along with the pulse rates into the PABX. What is the surety that he has entered the correct codes, since there is no monitor to display this.

So we must provide him a facility to have the printout, i.e., a harc copy of the codes and the pulse rates of what he had entered. This would allow him to have a double check. Also this facility will allow him to detect and correct errors in his codes. Also it will avoid unwanted ambiguity in his bill analysis.

6.STD CODES OF IMPORTANT PLACES

Irrespective of any user there may be certain places where all the users will call frequently. We can take into account such 150 cities and store their codes in the EPROM. So these codes will be provided to all the users irrespective of their geographical locations or need.

7.CALL MADE TO AN UNLISTED PLACE

Now let us assume a condition where a user makes a STD call to a place which is not in the given list stored in the database memory. If such a situation occurs i.e., if the has dialed to a place say Alathur with Std code 04922, the system will start searching for the code among the codes which are stored. It checks match with the first code (serial no), the next and so on till the STD code. The number of digits matching for each code is stored. Maximum digit matching is 3 (matching digits 0,4,9) and this happens for the code 049--. So pulse rate of 049-- is taken as the required pulse rate. The same logic applies for ISD as well. Since nearest matching implies proximity of locations, this method will approximate pulse rate to the closest value. However if no matching occurs the pulse rate by default is taken to be 2 for STD and 1 for ISD.

8.FLAT RATE FOR LOCAL CALLS

Irrespective of facilities and concessions provided by P&T i.e., free calls concession of the first x calls, we have fix the cost for local calls a constant. Since this is the basis of call charging we have to fix this flat.

CHAPTER - 6

MAIN PROGRAM AND ISR

THE MAIN AND INTERRUPT SERVICE ROUTINES

INTRODUCTION

Any exchange software should have the following modules,

* A main program
* Interrupt Service Routines (ISR)
* Call states
* Subroutines

The function of each is described below.

THE MAIN PROGRAM

The main program does the system initialization for both hardware and software. This routine processes the various real time information collected and takes appropriate action according to each event. Each subscriber is marked with a distinct call state at different stages of a call. All subscribers are taken up one after the another sequentially and for each subscriber, depending on the call state, that particular call state routine is executed.

THE I.S.R.

The Interrupt Service Routines are called after a particular duration like 8ms,16ms,160ms. The function of the ISR is to collect information about the current status of a particular subscriber. The current status is stored in memory variables.

These variables are constantly upgraded with the latest information about the subscribers.

CALLSTATES

The function of the call state routines are to execute a particular block of functions depending on the status of each extension. The call state routines form the core of the system software.

SUBROUTINES

The subroutines perform the various operations which are often repeated in the software. They reduce the redundancy involved in the software. They are called by call states to perform a particular operation for that call state.

INTERRUPT SERVICE ROUTINES

There are basically 3 ISR s used,

* First after every 8ms
* Second after every 16ms
* Third after every 160ms

We shall discuss them in detail so that the basic concepts are made lucid.

8ms ISR

When the handset is lifted up from its hook the condition is known as OFF HOOK condition. When the handset is on the hook the condition is known as ON HOOK. The aim of the 8ms interrupt service routine is to collect the information from each extension whether it is in the ON HOOK or OFF HOOK condition. When the telephone goes OFF HOOK the HS (active low) line goes low. This sets the active bit(AB). The 8ms ISR checks this bit whenever this is executed. If the last two status of the bit is the same and it is in the set condition then OFF HOOK condition is stored. Otherwise ON HOOK condition is stored. If already in OFF HOOK condition it simply skips the execution.

16ms ISR

The functions of the 16ms ISR are,

* Digit collection
* Looking for time out & updating time counters
* Setting & verifying flags
* Execution according to flags

160ms ISR

The 160ms ISR is used for tone generations. The tone is saved in the memory as a tone table and it is saved as multiple of 160ms.

When this ISR is executed, it checks the tone table and finds the corresponding bit. If it is in logic 1 then sound of 400Hz is propagated through that line, otherwise no sound is propagated.

Another function of this ISR is to increase general count and check for its maximum value. This is very useful for detecting time outs.

CHAPTER - 7

CONCEPT OF CALL STATES

CALL STATES

A call state is a routine that is to be executed for a particular line. Any line should be in one of the call states. There are different call states which have different functions. When a call state is executed, it shall change the current call state of the line by examining some conditions. The main program consists of an infinite loop in which different call states for different lines are executed. Different call states are explained for clear understanding. In each call state current telephone is passed as the parameter.

IDLE CALL STATE

The idle call state contains the code for checking different flags and if any one of them is set, then it takes actions according to that. It looks out whether any digit is dialled, if so it changes the call state of the extension to the first digit analysis call state.

FIRST DIGIT ANALYSIS CALL STATE

This call state contains the code for checking the digit dialled and its validity. If it is not valid it gives an error message. If it is valid it checks whether it is a trunk. If it is not trunk it will establish connection with the called telephone provided it is free. If the other telephone is busy it will give a busy tone. But if trunk is dialled it will check for the extension eligibility. If eligible and the trunk line is free it will allocate the extension with the trunk and will change the call state to trunk access call state.

TERMINATION CALL STATE

This call state contains the code for checking the call termination condition. It will check for the HS line. If it goes high then it implies that the extension is into the termination call state.

RING BACK TONE CALL STATE

This call state will check whether the called extension goes OFF HOOK. Till then it will give a ring back tone. If the called extension goes OFF HOOK, then it will establish a connection and change both call state to intercom call state.

TRUNK ACCESS CALL STATE

The functions of this call state is same as that of ring back call state but it is applied to the telephone which is in the ringing state.

INTERCOM CALL STATE

This call state simply makes the connection and sets the necessary flags. If conversation is over it simply resets the flags.

CHAPTER - 8

STORING AND RETRIEVING OF CODES

STORING & RETRIEVING OF CODES

STORING OF CODES

To provide the user the option of entering the codes most frequently used, memory space is allocated for storage of 180 codes. The user can program in the memory 150 STD codes and 30 ISD codes varying according to requirements.

A separate block of memory locations are dedicated for this purpose. Seven locations are allotted for each code with each digit being stored in consecutive locations. The termination of the code is identified by storing FF after the last digit of the code. This procedure is repeated for all the codes.

The first 150 codes are the STD codes and the succeeding 30 codes are the ISD codes. Upto seven digits are stored in a particular code. Corresponding to every code there is a pulserate. Pulserates of the various codes are stored in another block of memory with pulserate for first code being stored in first location and succeeded by the next pulse rate for the next code and this is repeated for all the codes.

RETRIEVING OF CODES

When a particular code is dialled the digits are stored as continuos bytes in the memory locations. After ascertaining whether the code is ISD or STD and branching to the corresponding location, a search is made for finding maximum possible match between the code stored in the memory and the dialled code. When the maximum possible match is found ,corresponding pulserate is easily retrieved by using the serial number as a link.

CHAPTER - 9

DESIGN

CODE ENTRY DETAILS

STORING OF STANDARD CODES

9559

The STD / ISD codes and pulse rate for the selected cities will be loaded once 9559 is dialed . Customer will have to alter the pulse rate for the cities in concurrence with the pulse rate listed by local telephone exchanges. For obtaining the pulse rates the telephone directory of local exchange is referred. We can change the codes stored in the system to include some codes of our choice. 150 locations are provided for storing STD numbers and another 30 for storing ISD codes.

$\boxed{\text{A.2}}$ MODIFICATION OF STD / ISD CODE AND PULSE RATE

To alter STD /ISD codes or pulse rate of existing codes, we have to proceed as follows. Refer code identification algorithm given in page -- for optimization of code entry.

We have to ensure that programming is done from an extension which is declared as DTMF extension and dialling is done in DTMF mode.

The following are the options available.

A.2.1 To change STD / ISD code and corresponding pulse rate, we have to dial

91 + Serial no. (3 digits) + pulse rate (3 digits) + STD /ISD code followed by * key of the telephone.

The range of Serial no. is 001 to 150 for STD codes. For ISD codes serial no. is between 151 and 180. Similarly the range of value for pulse rate is 001 - 180.

After entering pulse rate, a beep tone can be heard. This is the acceptance tone for pulse rate entry. After entering STD code, dial * . Now dial tone will be heard which is the acceptance tone for STD code entry. Here * key acts as end of dialing code. This applies to all programming involving entry of pulse rate and STD code.

For example, we have initially stored serial no. 23 corresponding to the code 0278 (STD code of Bhavnagar) and pulse rate of 2. Now to store the STD code of 049650 (STD code of Chombala) and pulse rate of 4 in that position,we proceed as follows,

Dial, **91 023 004 049626***

A.2.2 To store a pulse rate for a number of STD codes stored in consecutive locations, we have to dial

92 + Starting Serial no. + Ending Serial no. + Pulse rate.

eg: To store a pulse rate of 4 for STD codes of serial no. 30 to 40, dial

92 030 040 004

Dial tone will be heard when dialing is completed.

Now pulse rate of 4 will be stored for all codes with serial no. 30 to 40.

A.2.3. If we have stored some pulse rate or STD / ISD code at a location using code 91, we have an option to store pulse rate and STD / ISD code in next location. For this, dial

93 + Pulse rate + STD / ISD code + *.

eg: Suppose we have entered a code and pulse rate at location with serial no. 23. Now we have to enter STD / ISD code (0491)and pulse rate (4)at next location ie.24, then dial

$\boxed{93\ 004\ 0491\ *}$

A.2.4. Using code 93 we can enter a pulse rate and STD / ISD code at consecutive locations. If we do not wish to enter pulse rate we have to enter STD / ISD codes only at consecutive locations. For this, dial

94 + STD / ISD code + *.

$\boxed{\text{A.3}}$ PRINT / VIEW STD CODES AND PULSE RATES

After we have entered the STD / ISD codes and their pulse rates, we can verify whether codes and pulse rates have been entered correctly. We can take a print out of the codes and pulse rates in printer or view it in serial port.

To take the print out in printer, the following options are available.

A.3.1 To print STD / ISD codes and pulse rates of all locations, dial 951

Dial tone will be heard and if printer is setup, then code will be printed out.

A.3.2 To print STD / ISD codes starting from a location, dial

952 + initial serial no.

Dial tone will be heard and codes starting from initial serial no. to the last serial no.(180) will be printed.

A.3.3 To print STD / ISD codes between two locations, dial

953 + initial serial no. + final serial no.

A.3.4 Alternately to view the codes and pulse rates in serial monitor or PC connected to serial port after setting up the serial port, dial

954

All codes and pulse rates will be displayed in the monitor.

A.4 UNIT PULSE COST

Now that codes and pulse rate are entered, Next parameter to be programmed is the unit pulse cost. This is the cost of one pulse. Default value for pulse cost is Rs 1.00. To program this, dial

972 + X + Y, where unit pulse cost will be X.Y.

eg: If we wish to program unit pulse cost of Rs 1.20, then dial

972 1 2

The maximum value that can be programmed is Rs 9.90.

A.5 PULSE RATE FOR LOCAL CALLS

Default value for pulse rate for local calls is 3 minutes i.e. call duration of 3 minute will be counted as one pulse. If we wish to alter this, dial

973 + X , where X is the pulse rate in minutes.

X can have a maximum value of 9.

A.6 CALCULATION OF CALL DURATION

There are two methods for identifying start of call which is needed for calculation of call duration. One of them is *polarity reversal* method. In this the call duration will be calculated from the time a polarity reversal is obtained in trunk (provided by P& T). If P&T does not provide polarity reversal in your trunk we can use the second method. This is the *time start* method. In this, call duration will be calculated after a pre-programmed time (called *no-reversal time*) from the time dialing is complete. After we have dialed out all the digits in trunk, system will wait for 6 seconds to confirm end of dialling and then for no-reversal time before marking start of call. Suppose we have programmed a no-reversal time of 9 seconds . Then call will be metered 15 seconds after we have completed dialling. This time can be varied by varying the no-reversal time.

A.7 CALL STORAGE AND DIRECTION

If we have enabled SMDR by dialling 9559, then call details will be stored in memory. We can also direct the call to printer and serial port of computer. If directed to printer, then as and when call is over, call details will be printed in the printer or displayed in monitor of computer, provided the setup is done properly. This is called online printing.

A.7.1 To enable online printing, dial

$\boxed{558\ 1}$

If online printing is enabled, the printer or computer connected to it becomes dedicated. Alternately we can disable online printing and take print out of call details in printer or store it in computer at a later time.

A.7.2 To disable online printing, dial

$\boxed{558\ 0}$

A.7.3 Another option for storage is whether all calls are to be stored or only STD calls need to be stored. This is needed when no. of calls made are large and you dont wish to take print out often. Memory in system can store details of 1100 calls.

If we wish to record all calls, dial

$\boxed{559\ 0}$

If we wish to record only STD calls, dial

559 1

A.7 CODE IDENTIFICATION ALGORITM & OPTIMIZATION

150 STD codes and 30 ISD codes will be stored in the system. This is for the convenience of the user who need not enter the codes while enabling the feature. We have to enter the pulse rates only. Naturally the question arises that what pulse rate will be assigned when the code you dial is not one among the codes we have stored. A method which looks for the maximum match between code dialed and the codes stored is used to deduce the pulse rate for the code. To illustrate see this example,

Suppose you have dialed to a place, say ALATHUR, with STD code 04922. System starts searching for this code among the codes which are stored. It checks match with the first code(serial no.1), the next and so on till the last STD code. The number of digits that are matching for each code is stored. Maximum digit matching is 3 (matching digits are 0,4 and 9) and this happens *first time* for the code 049-- . So pulse rate of code 049-- is taken as the required pulse rate. If the code 04922 was stored , then maximum match of 5 would have resulted and pulse rate corresponding to code 04922 would be assigned. The same logic applies to ISD codes as well. Since nearest matching implies proximity of locations, this method will approximate the pulse rate to closest value.

This method also gives scope to optimize the entry of codes. This is only needed if we wish to have more standard codes than the ones presently stored. It can be seen from the telephone directory that the codes to far-off places are almost same. That is, if we are at PALAKKAD in KERALA, the pulse rate to all places with STD code starting with 01 is 2. Hence we need to store only one code starting with 01 and a corresponding pulse rate of 2. Pulse rate to nearby places vary more significantly, Hence for accuracy they need to be stored explicitly.

A.8 TIME ZONES

Another aspect which determines the no. of pulses for a call is the time at which call is made. It can be seen that first time zone is between 6.00AM and 7.00 AM and the pulse rate multiplier is 1/3. This means that if we make a call at these times and speak for a duration corresponding to a time equivalent to 15 pulses, then system will calculate it as only 5 pulses (15*1/3).

If a call starts in one time zone and extends to next time zone then the time zone of the call will be the one in which conversation started.

Time zones are given by P& T. In normal cases user can follow the same. But if we wish to alter the time corresponding to the zones then provision has been provided to change the timings.

There are 6 time zones and hence 6 times in 24 hour format spread over 24 hours. The first time given is the time of start of zone with pulse multiplier 1/3. It is given as 06.00 (AM) This zone extends upto 07.00 (AM). If we alter any of this time then duration of first time zone will get extended or shortened.

If we wish to modify any of these timings, dial

A.8.1 To alter the time of start of zone with pulse multiplier 1/3 (given as 06.00 AM), dial

961.+ HH + MM

OPTIONS FOR PRINTING OF CALL DETAILS

The following are the options available for printing of call details. Printing can be directed to a printer or to serial port.

ALGORITHM

CS61- PROGRAMMING FOR CODE ENTRY

1. Check the hook status of an extension.
2. If no digit has been dialled within the period give error tone
3. If number of digits dialled is <2 ,return
4. If 2nd digit is >8 give error
5. If second digit is 2 ,dialled number is 92 else goto step 14
6. Get the initial serial number and convert to hex.
7. If Sno >= 151,ISD programming is initialised else STD

   programming is initialised
8. If Sno >180,give error.
9. If digit pointer >8 get final serialno
10. If final Sno < initial Sno , give error
11. Return the pulsate depending on the code
12. Store the pulse rate in location corresponding to initial serialno

    storing is continued till the final serialno location.
13. Give successful tone , change Cs to38 and reset ISD_SNO

14. If second digit is 3 check if 91 or 92 is dialled priorly else give error.
15. If Sno >180 give error.
16. If digptr <5 return else return hex pulserate
17. If second digit is 6 change to printing call state
    reset general count and return
18. If second digit is 5,repeat steps 18 and 19
19. If second digit is 4 check if 91 or 92 is dialled priorly
20. If not dialled give error
21. If Sno<181 change to cs62,reset and return
22. If 91 is not dialled ,check for 92
23. If digit ptr <5 return and give error
24. If not Initialise isd sno and check whether std or isd code
25. Store the pulse rate in location corresponding to Sno and
    give acceptance tone

CS62-PROGRAM STD CODE AND PULSERATE

1. Get the general count.
2. Feed 920ms of confirmation one

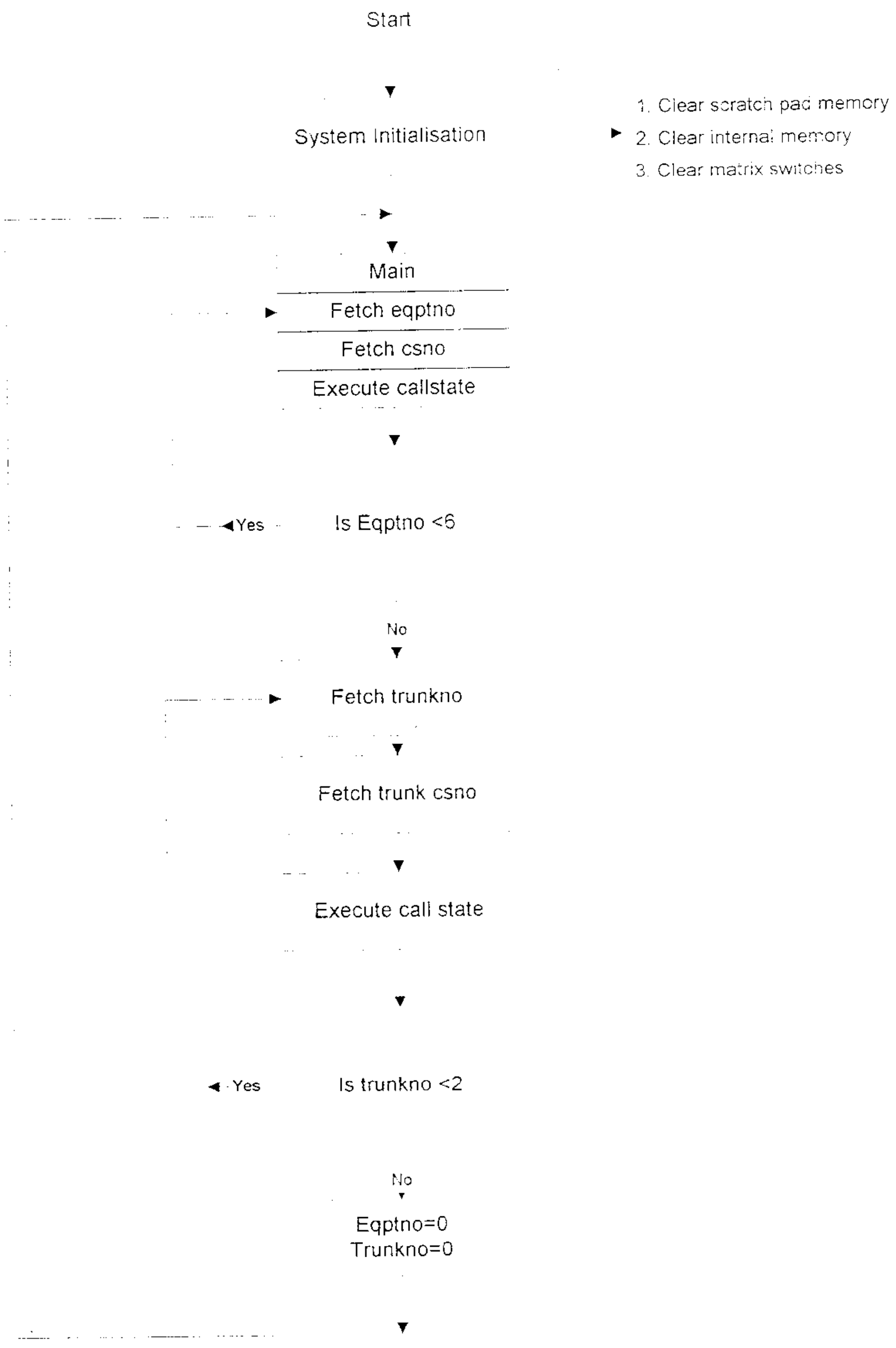
3. If not dialled within 15seconds, give an error tone and return
4. If dialled, check for OFF HOOK status
5. Get the digit pointer
6. If digit pointer >8, give error else get the Nth digit.
7. If * is not dialled, return else get the final entry
8. For ISD, Initialise ISD programming to 0.
9. For STD programming, remove the 0 dialled as the first digit.
10. Increment the serial number after entering each code.

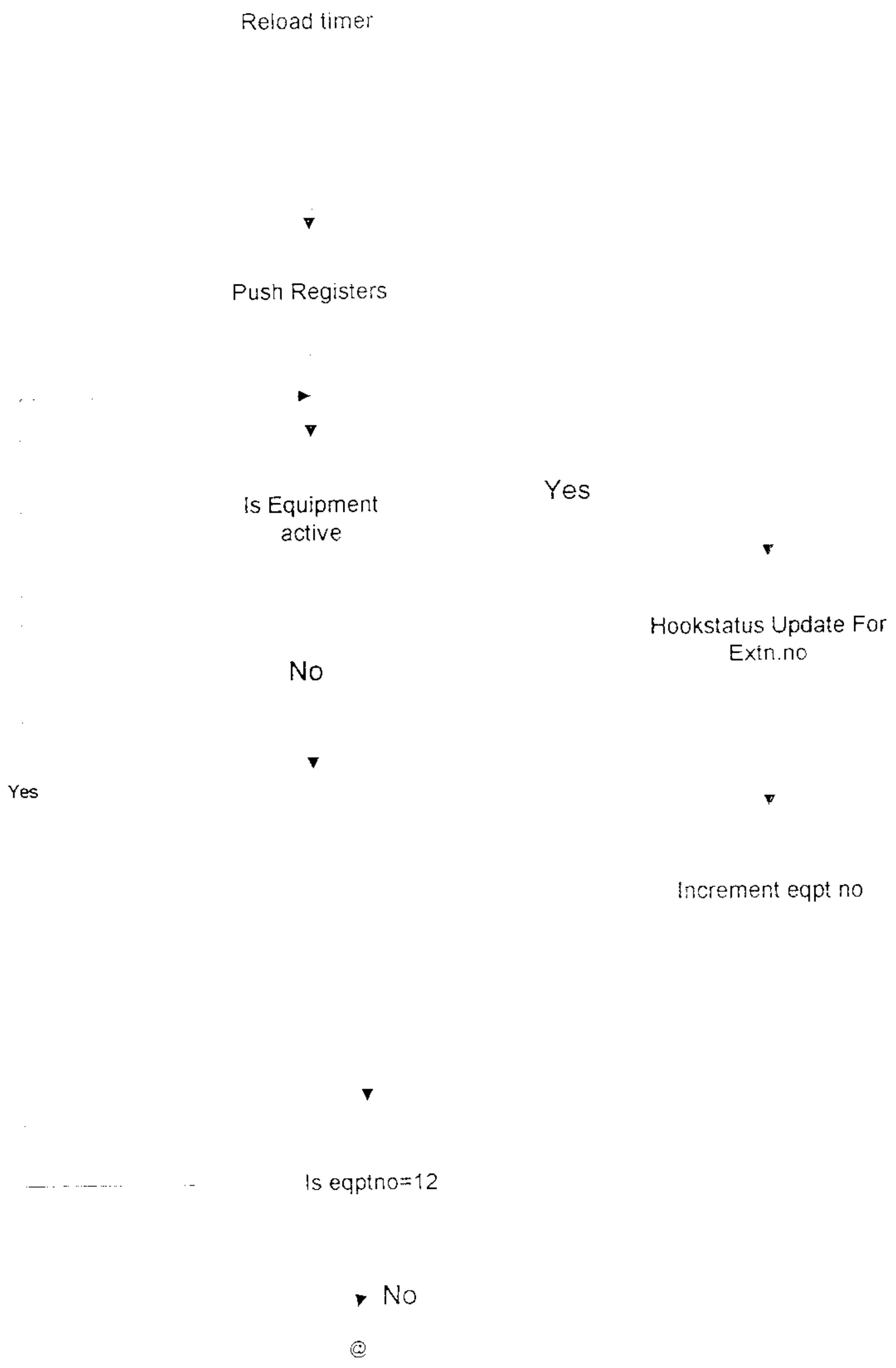
CS63 - TIME ZONE ,HOLIDAYS & CALL LIMITING

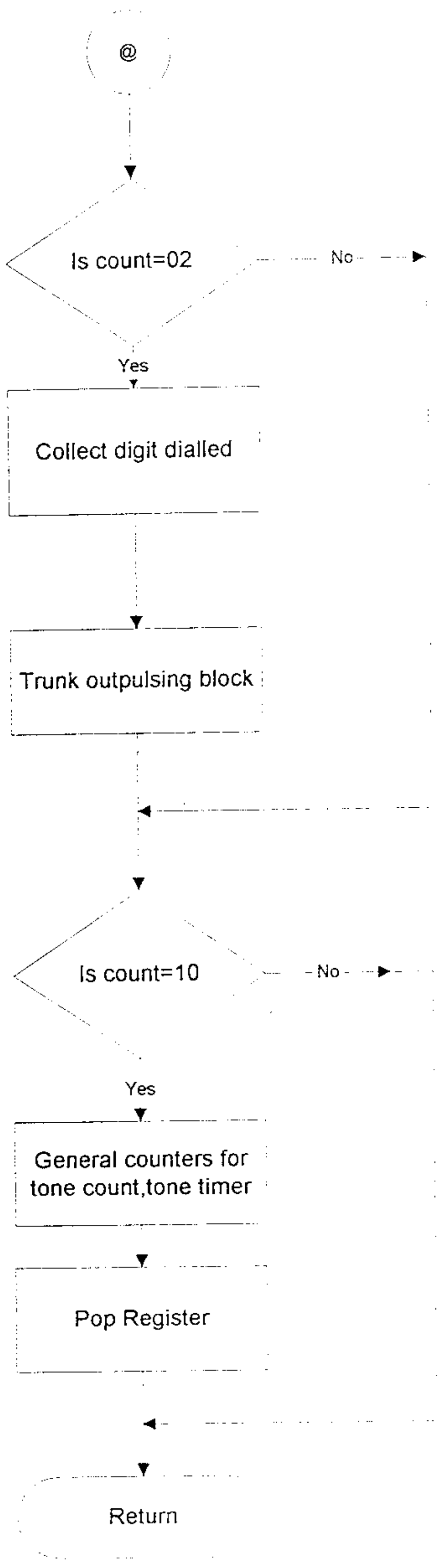
1. Get the hook block status for extension
2. If a digit has been dialled, get the digit pointer.
3. If second digit is 6,initialise time zones and get the third digit else go to step 7
4. If third digit >6 ,give error and return.
5. If digit pointer is <= 7,get the hour and minute and store it else give error and return
6. Validate hours and minutes and return
7. If 2nd digit <>7,give error and return.
8. If third digit is 1,store the holiday number, date and the month and return else go to step 9.
9. If third digit is 2 store the unit cost else go to step 11
10. Give dial tone and return.
11. If third digit is 3 and if dialled digit is not valid return else go to step 13

12. Store the local pulse rate and return
13. If third digit is 4 ,validate extension number else go to step 15
14. If extension is enabled ,collect the digits representing the limiting cost , store it and return.
15. If dialled digit is 97 and if third digit is 6,store digit for SMDR ENABLE else give error.
16. Give Dial tone and return.
17. If 95 is dialled ,check for third digit else give error and return.
18. If third digit is 7, enable or disable SMDR according to 4th digit else give error and return.
19. If fourth digit is 4 enable printer on and monitor else repeat process for 957..
20. Print STD codes
21. Give dial tone and return.
22. If third digit is 5 ,check for fourth digit else go to step 19.
23. If fourth digit is 9, enter standard STD codes.
24. If digit is dialled, check for 3rd digit
25. If third digit is 3 ,check for validity of initial and final serial sno.
26. If valid, set the printer ON, title and STD code .
27. Give dial tone and return.
28. If third digit is 5,set the default rate else return.
29. Disable call budgeting and initialise time zones.
30. Initialise pulse rate for local calls and enable SMDR.
31. Give tone and return.
32. If fourth digit is 3,get extension number else give error and return.
33. Get the extension details from HOOKBLOCK.
34. Calculate total extension cost if third digit <>6.

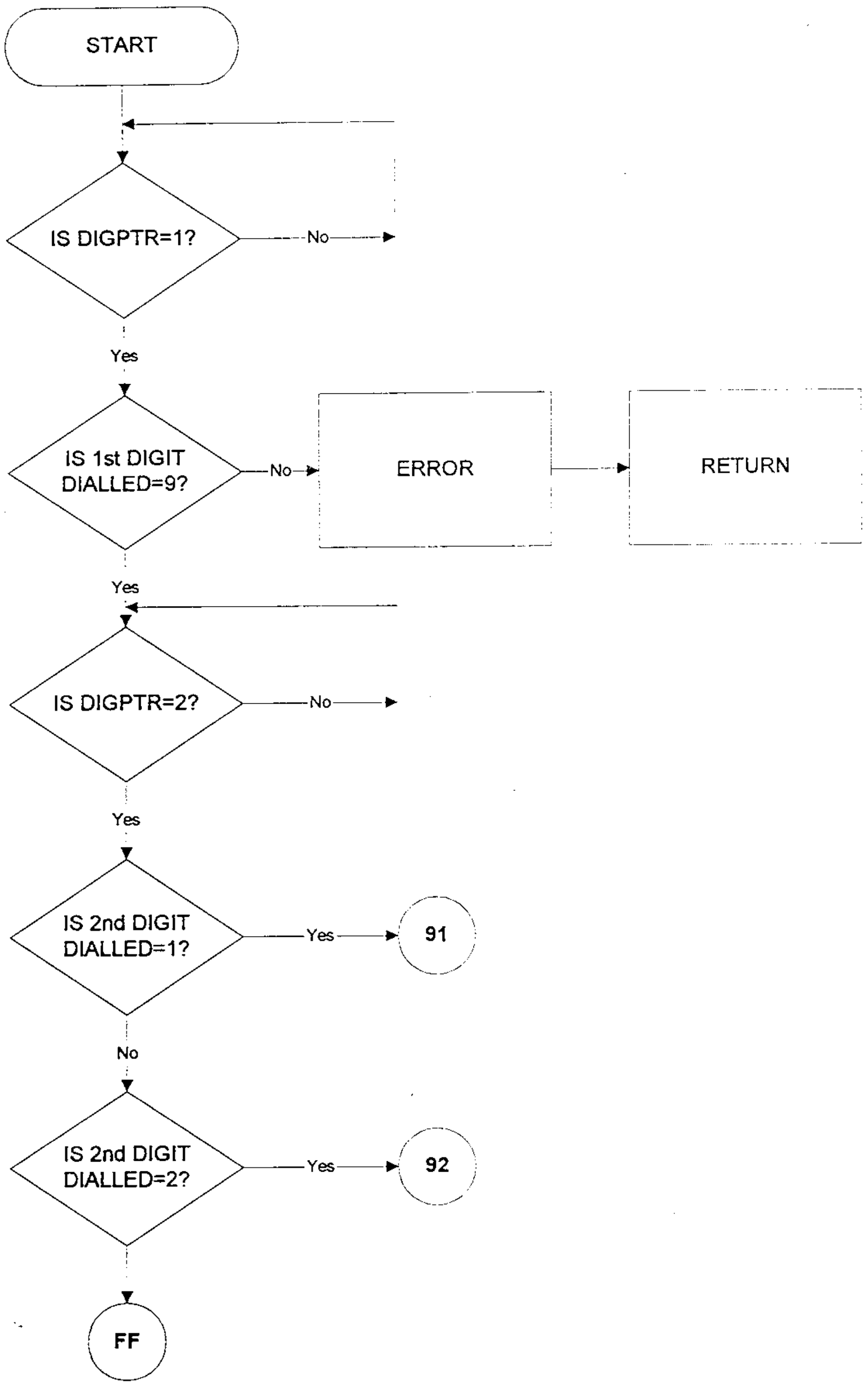
MAIN PROGRAM


Start
System Initialisation
1. Clear scratch pad memory
2. Clear internal memory
3. Clear matrix switches
Main
Fetch eqptno
Fetch csno
Execute callstate
Is Eqptno <6
Yes
No
Fetch trunkno
Fetch trunk csno
Execute call state
Is trunkno <2
Yes
No
Eqptno=0
Trunkno=0

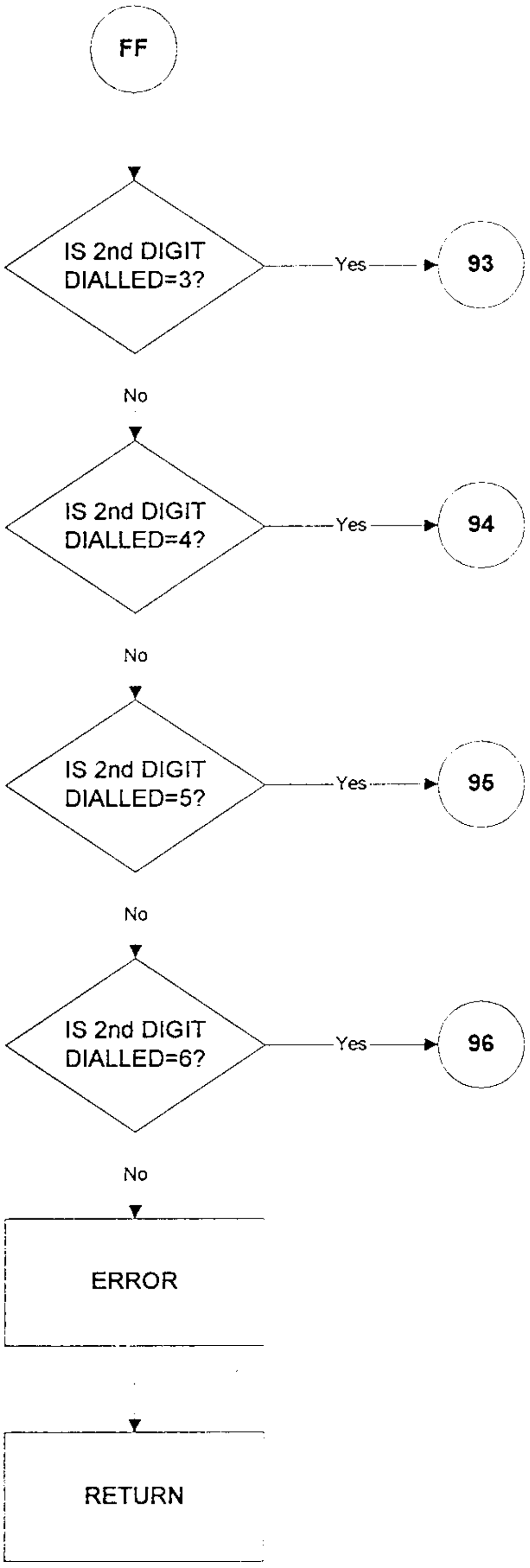

Reload timer
Push Registers
Is Equipment active
Yes
Hookstatus Update For Extn.no
No
Yes
Increment eqpt no
Is eqptno=12
No
@

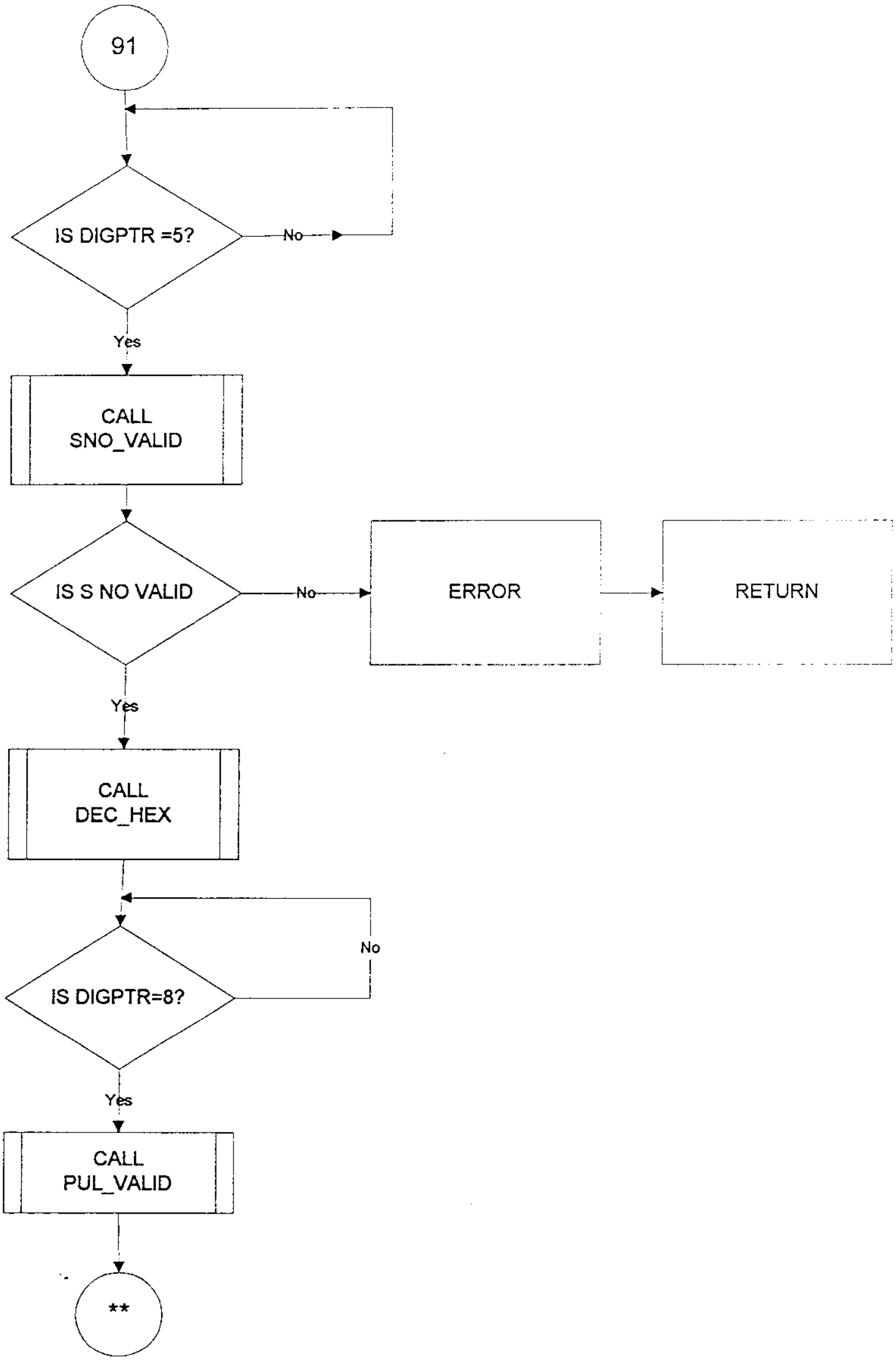

@
Is count=02
No
Yes
Collect digit dialled
Trunk outpulsing block
Is count=10
No
Yes
General counters for tone count,tone timer
Pop Register
Return

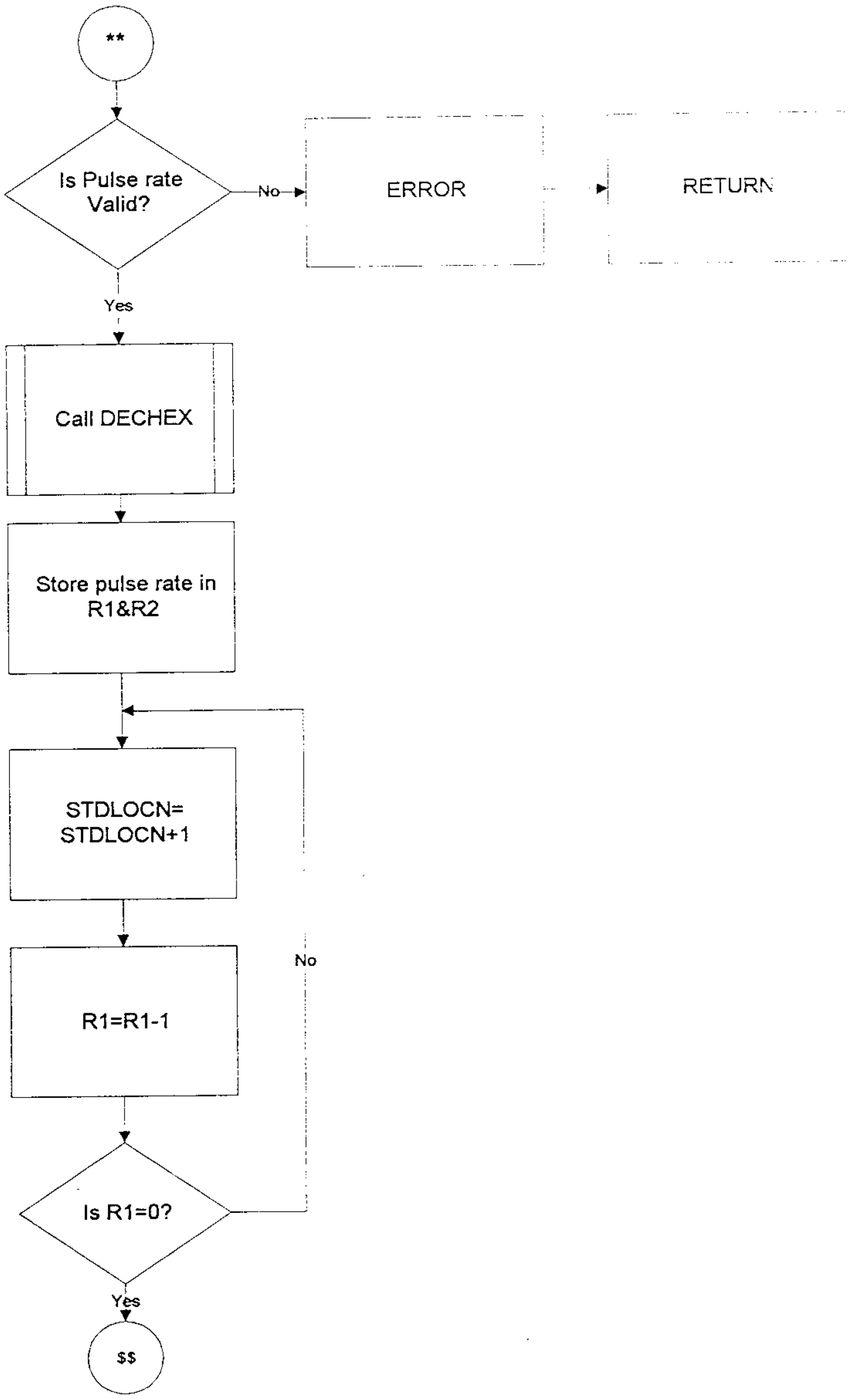
CODE ENTRY & STORING

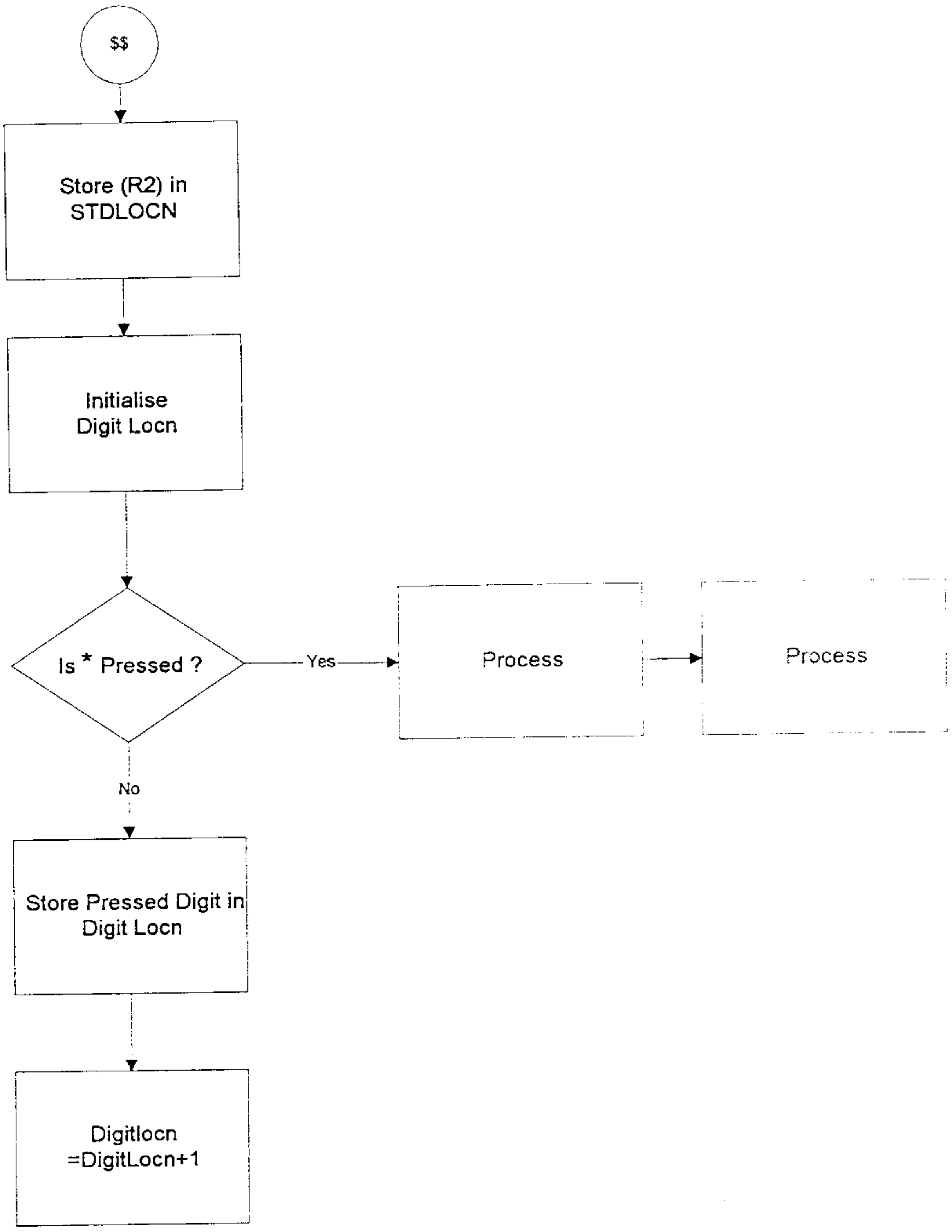


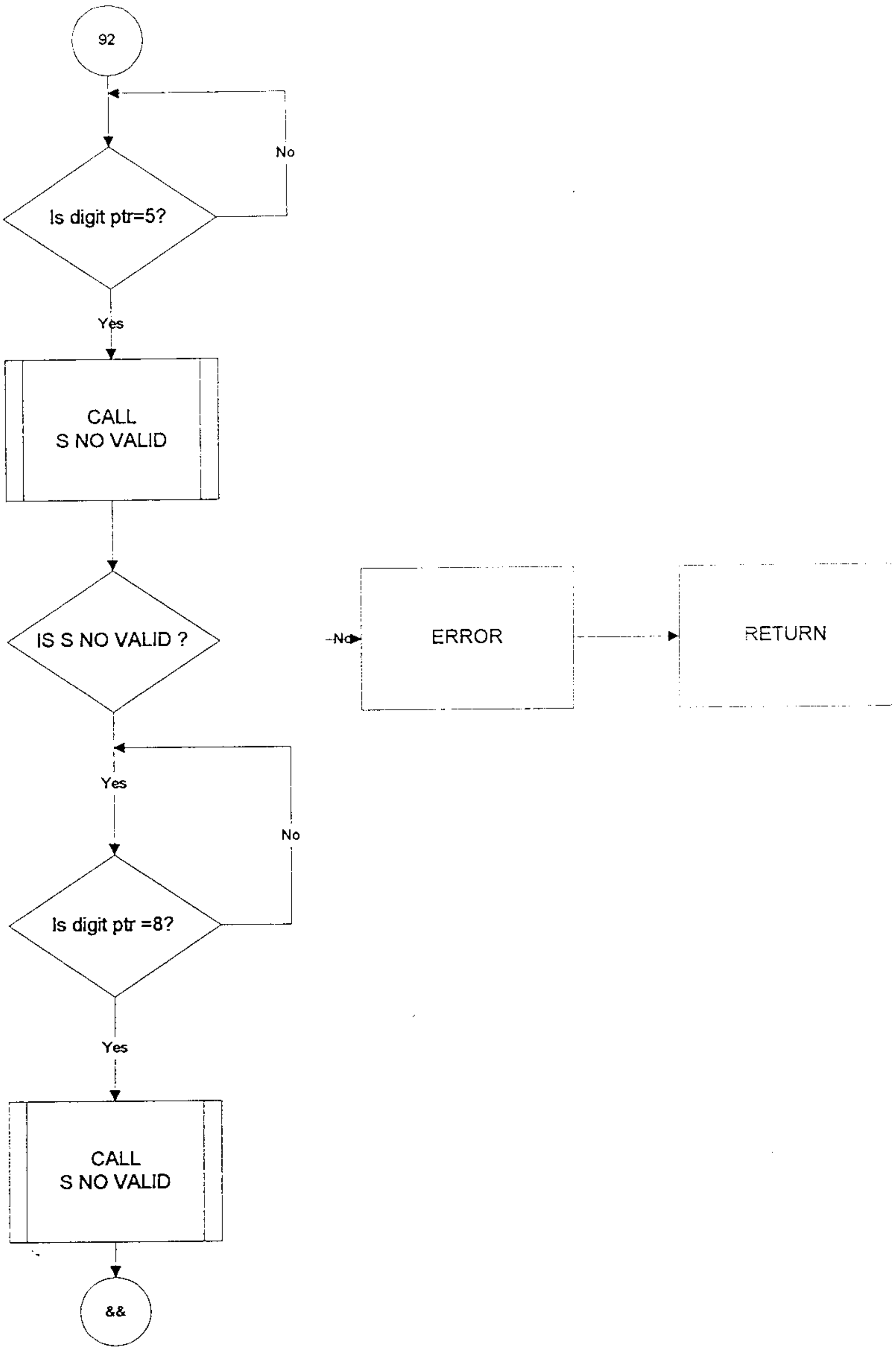
START
IS DIGPTR=1?
No
Yes
IS 1st DIGIT DIALLED=9?
No
ERROR
RETURN
Yes
IS DIGPTR=2?
No
Yes
IS 2nd DIGIT DIALLED=1?
Yes
91
No
IS 2nd DIGIT DIALLED=2?
Yes
92
FF

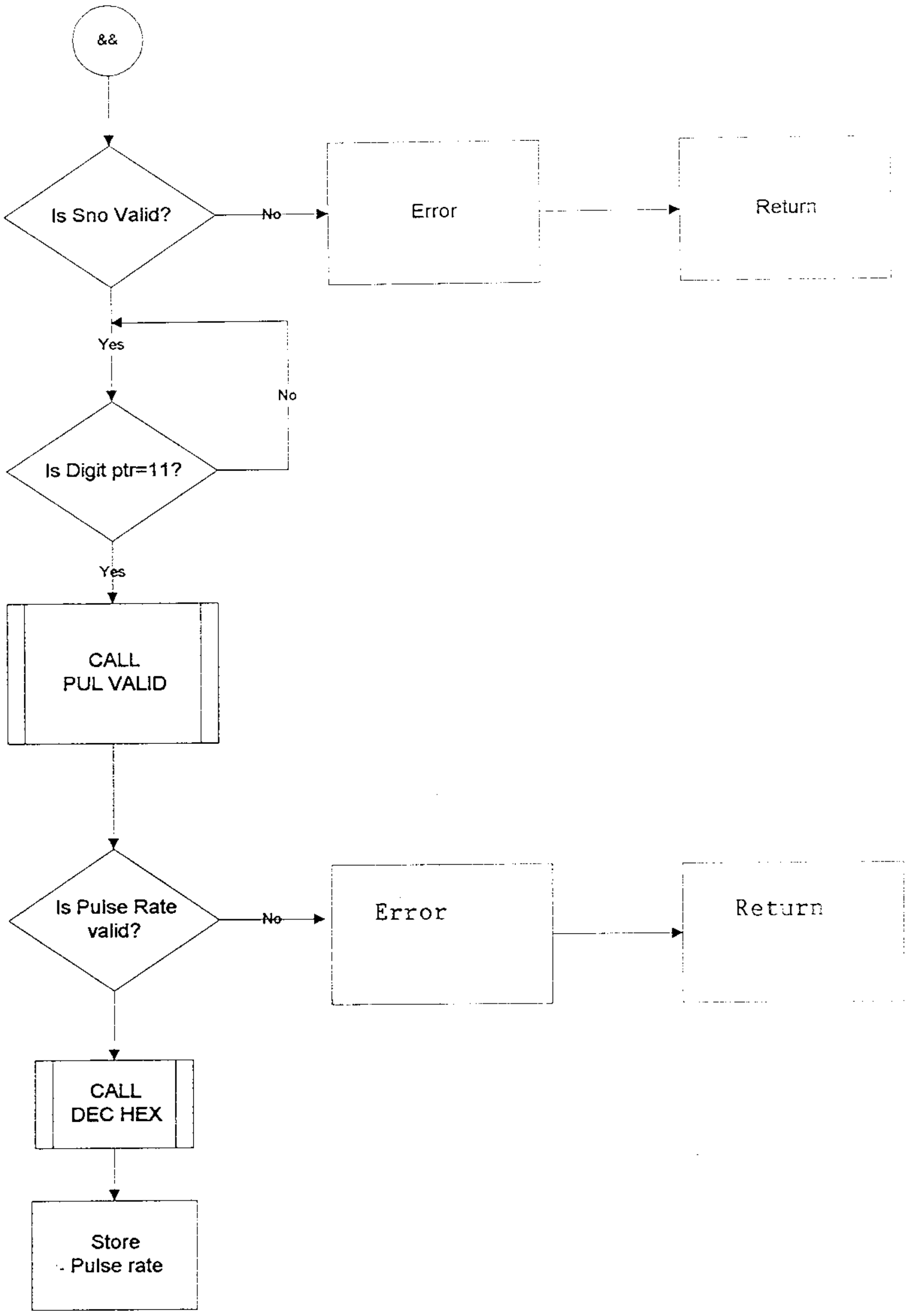

FF
IS 2nd DIGIT DIALLED=3?
Yes
93
No
IS 2nd DIGIT DIALLED=4?
Yes
94
No
IS 2nd DIGIT DIALLED=5?
Yes
95
No
IS 2nd DIGIT DIALLED=6?
Yes
96
No
ERROR
RETURN

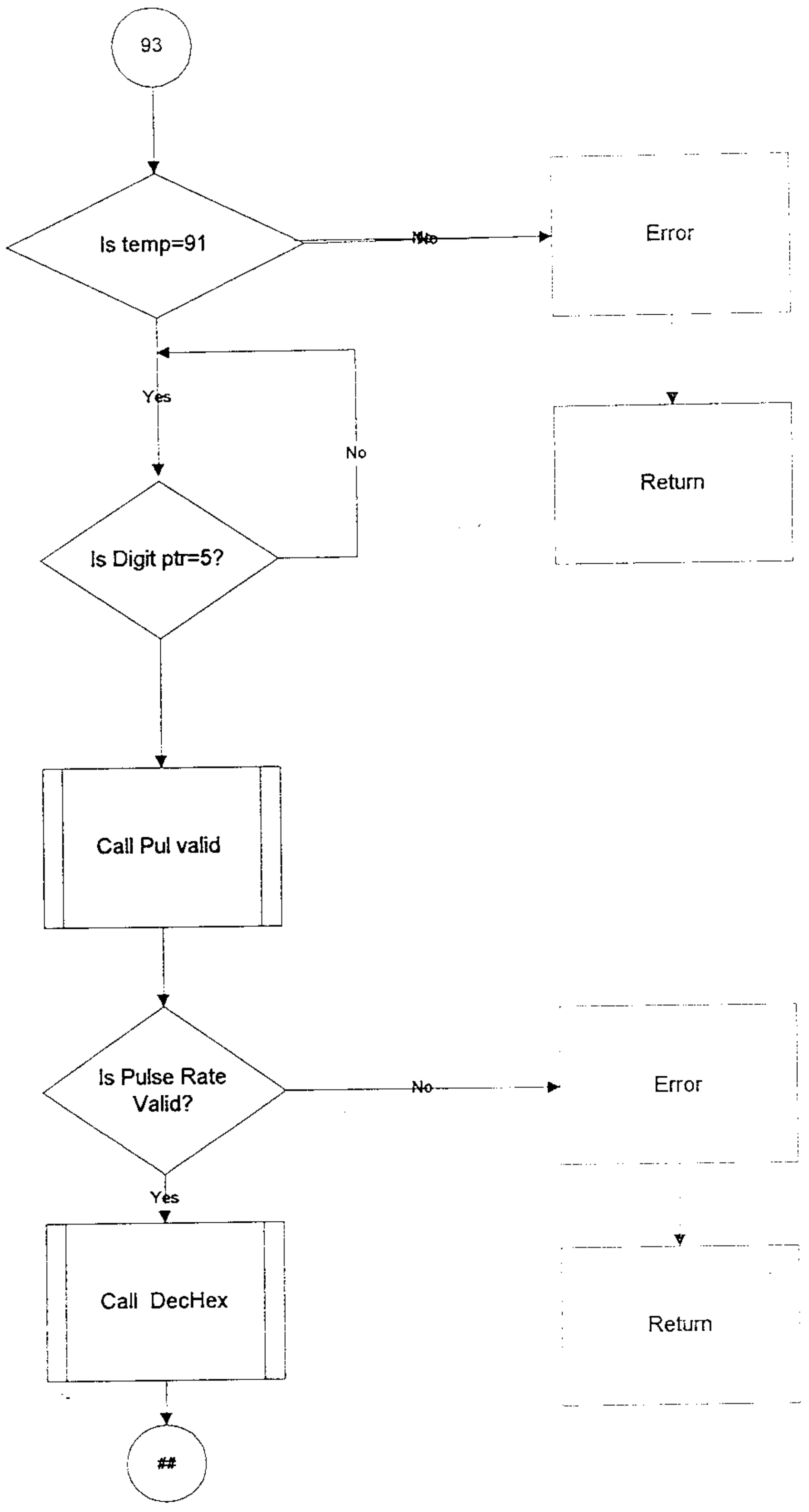

91
IS DIGPTR =5?
No
Yes
CALL SNO_VALID
IS S NO VALID
No
ERROR
RETURN
Yes
CALL DEC_HEX
IS DIGPTR=8?
No
Yes
CALL PUL_VALID
**

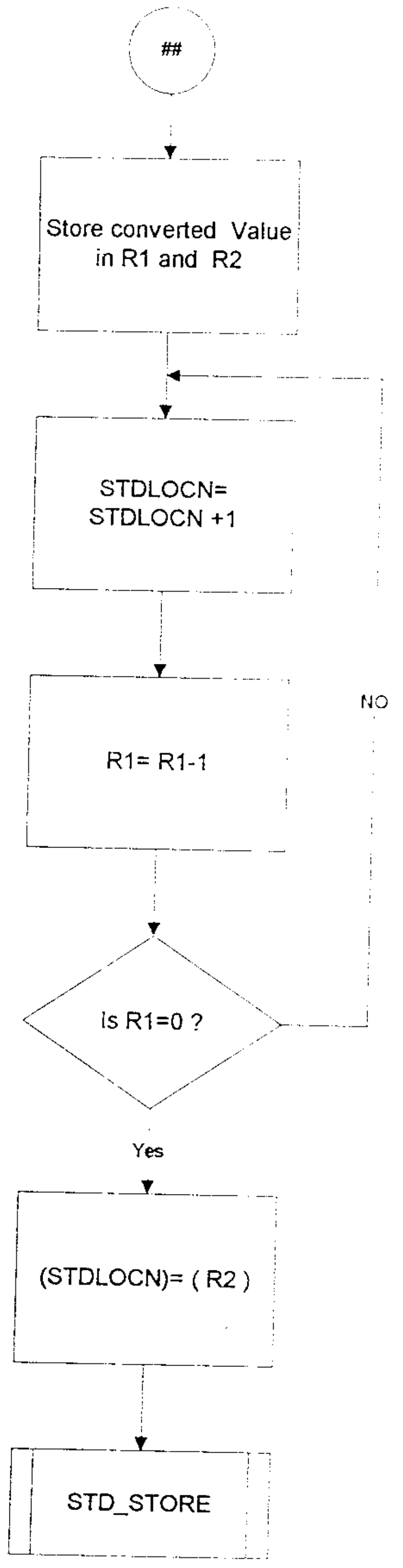

**
Is Pulse rate Valid?
No
ERROR
RETURN
Yes
Call DECHEX
Store pulse rate in R1&R2
STDLOCN= STDLOCN+1
R1=R1-1
No
Is R1=0?
Yes
$$

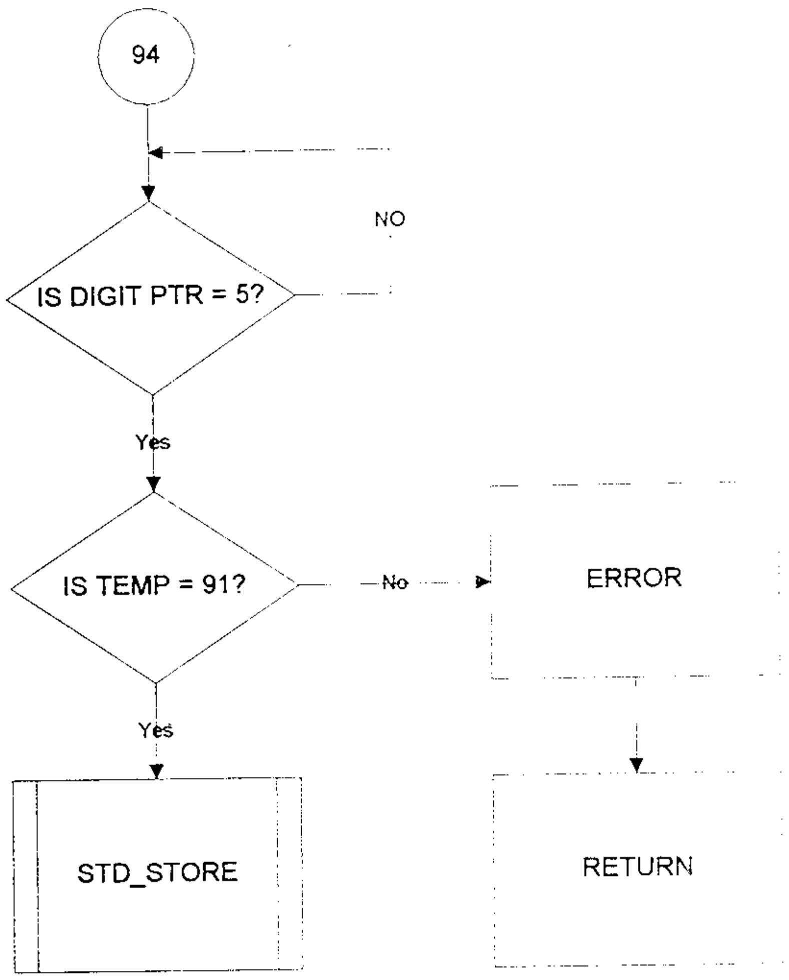

$$
Store (R2) in STDLOCN
Initialise Digit Locn
Is * Pressed ?
Yes
Process
Process
No
Store Pressed Digit in Digit Locn
Digitlocn =DigitLocn+1

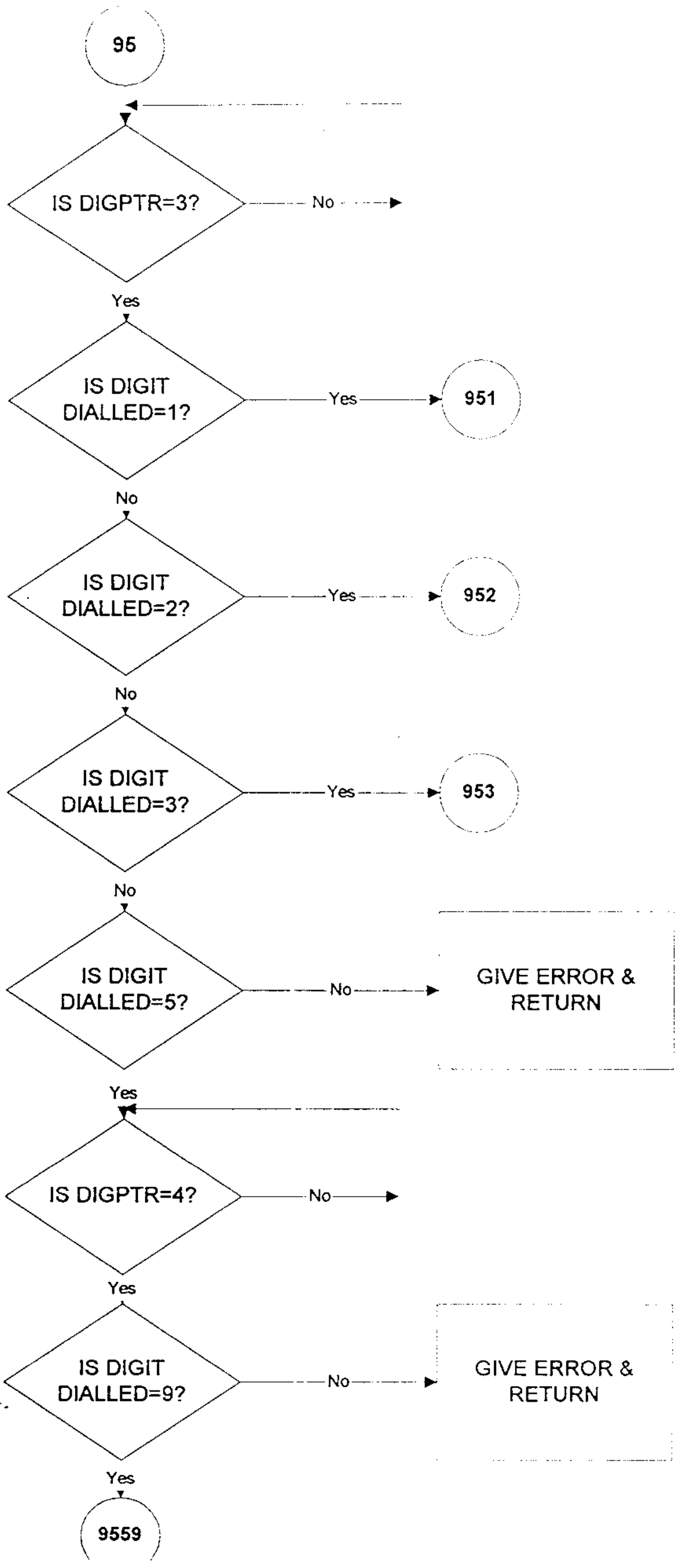

92
Is digit ptr=5?
No
Yes
CALL S NO VALID
IS S NO VALID ?
No
ERROR
RETURN
Yes
No
Is digit ptr =8?
Yes
CALL S NO VALID
&&

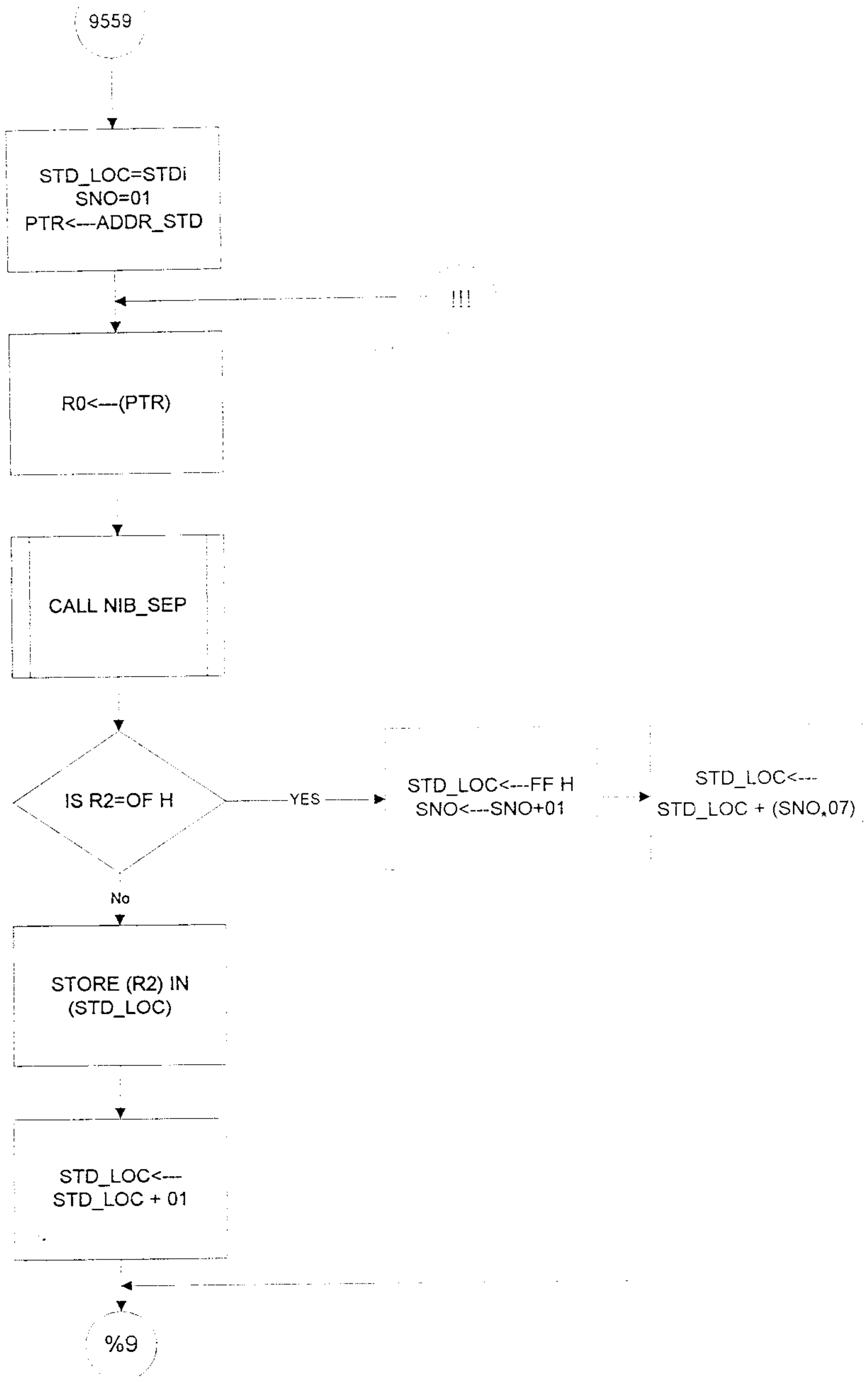

&&
Is Sno Valid?
No
Error
Return
Yes
No
Is Digit ptr=11?
Yes
CALL PUL VALID
Is Pulse Rate valid?
No
Error
Return
CALL DEC HEX
Store Pulse rate

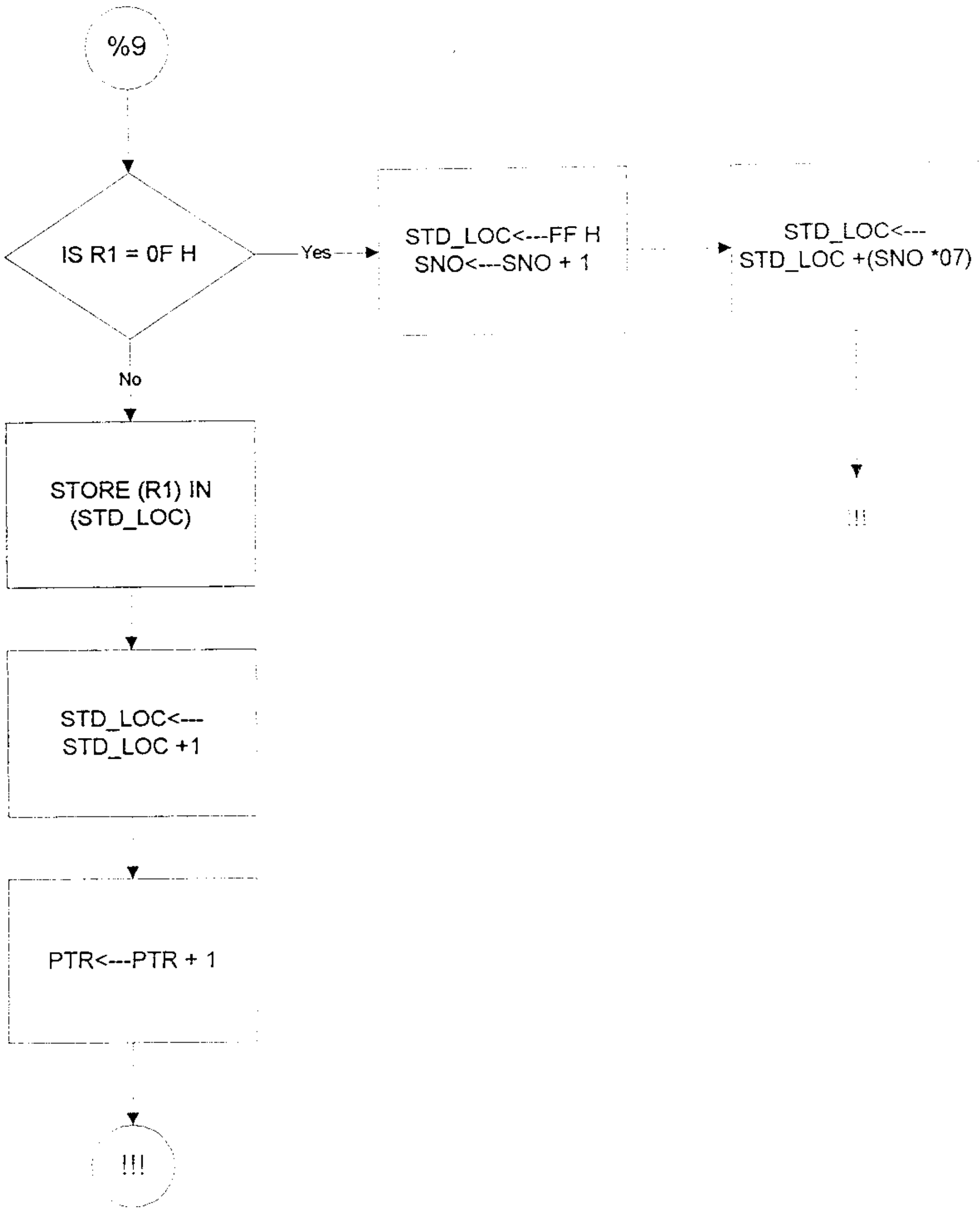

93
Is temp=91
No
Error
Return
Yes
No
Is Digit ptr=5?
Call Pul valid
Is Pulse Rate Valid?
No
Error
Yes
Call DecHex
Return
##


##
Store converted Value in R1 and R2
STDLOCN= STDLOCN +1
R1= R1-1
Is R1=0 ?
NO
Yes
(STDLOCN)= (R2)
STD_STORE

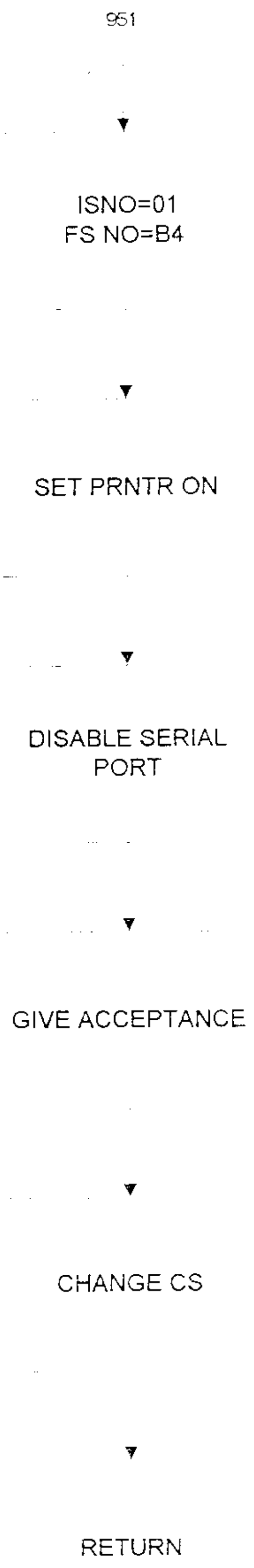

94
NO
IS DIGIT PTR = 5?
Yes
IS TEMP = 91?
No
ERROR
Yes
STD_STORE
RETURN

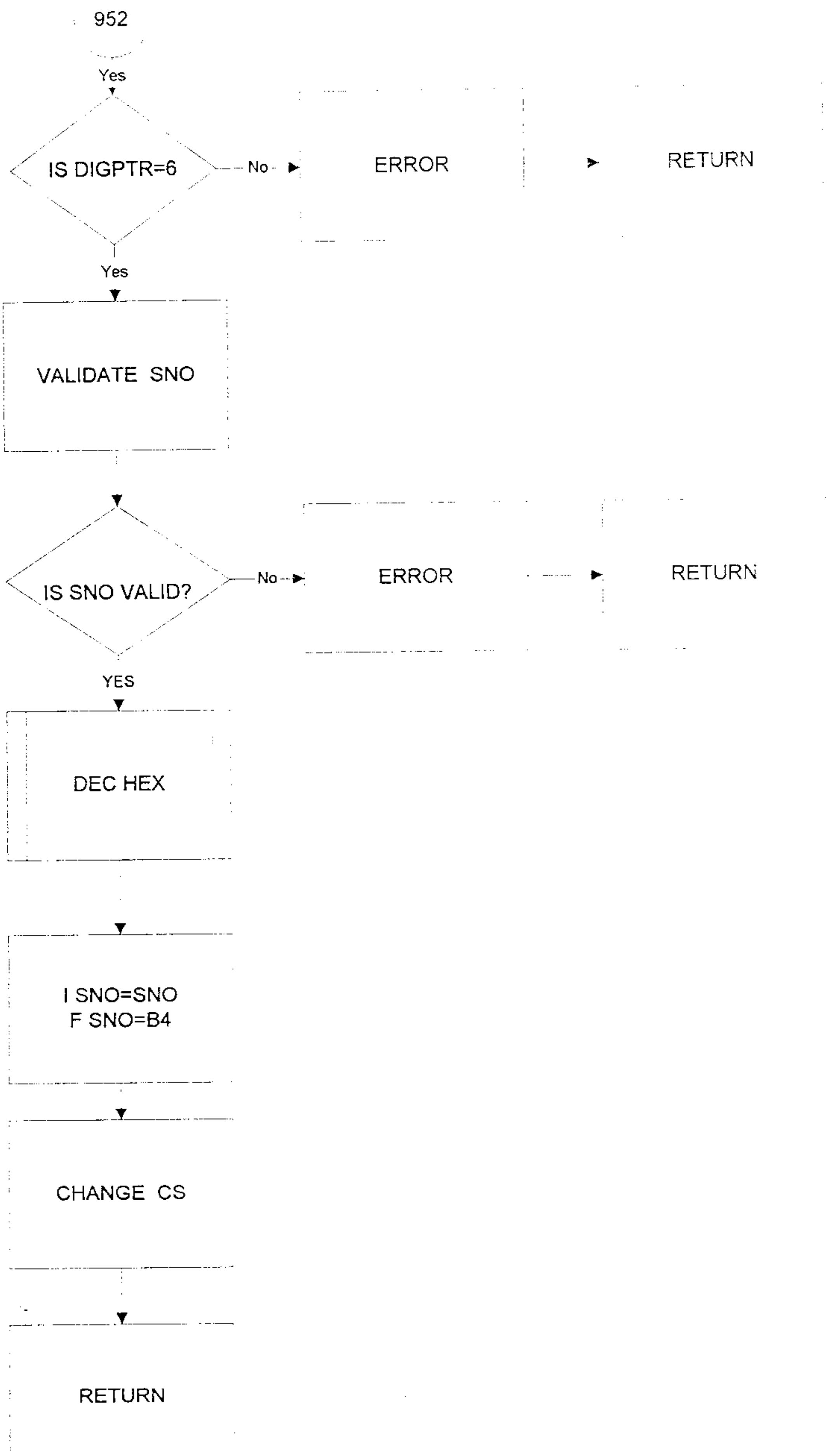

95
IS DIGPTR=3?
No
Yes
IS DIGIT DIALLED=1?
Yes
951
No
IS DIGIT DIALLED=2?
Yes
952
No
IS DIGIT DIALLED=3?
Yes
953
No
IS DIGIT DIALLED=5?
No
GIVE ERROR & RETURN
Yes
IS DIGPTR=4?
No
Yes
IS DIGIT DIALLED=9?
No
GIVE ERROR & RETURN
Yes
9559


9559
STD_LOC=STDi
SNO=01
PTR<---ADDR_STD
!!!
R0<---(PTR)
CALL NIB_SEP
IS R2=OF H
YES
STD_LOC<---FF H
SNO<---SNO+01
STD_LOC<---
STD_LOC + (SNO*07)
No
STORE (R2) IN
(STD_LOC)
STD_LOC<---
STD_LOC + 01
%9

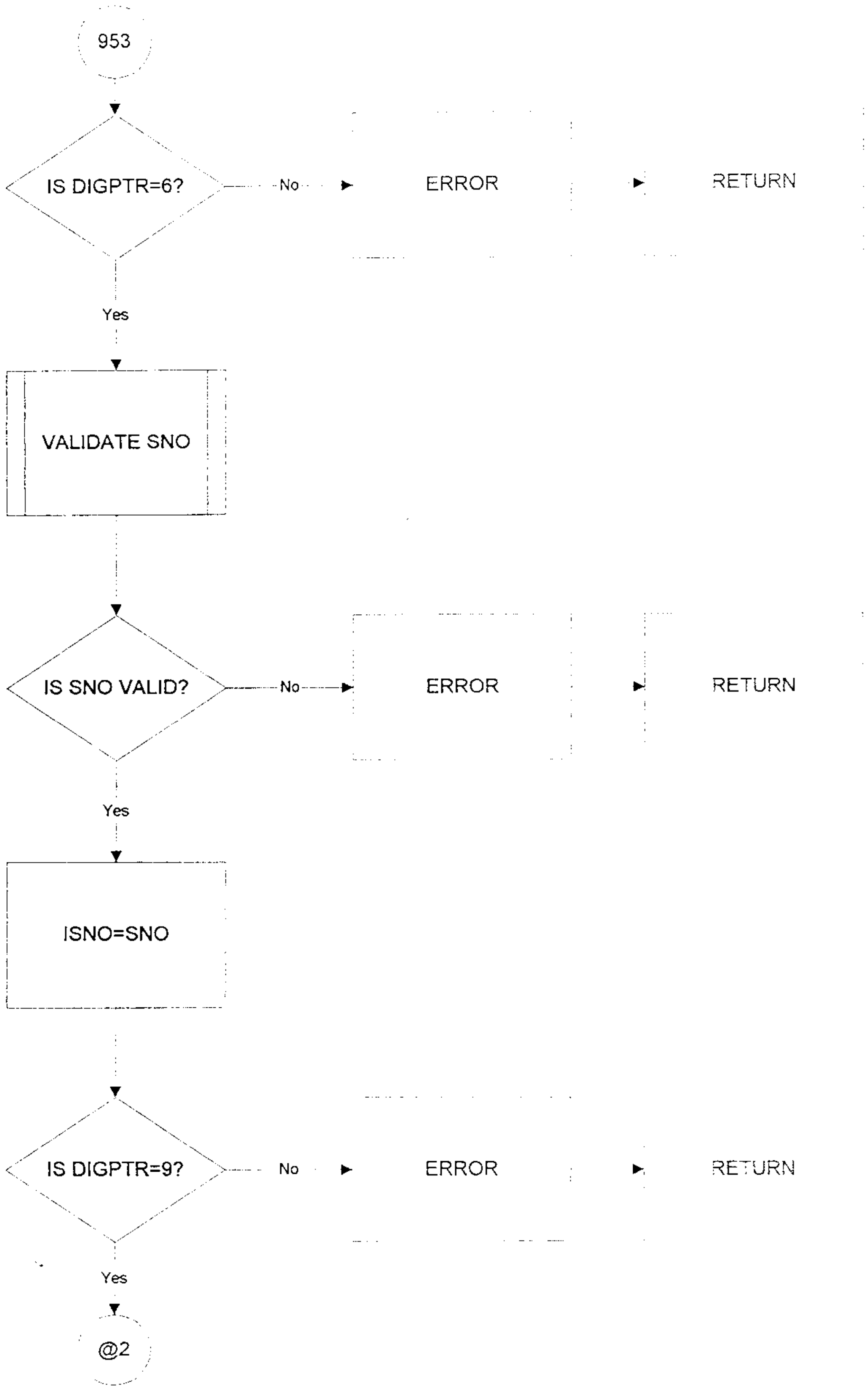

%9
IS R1 = 0F H
Yes
STD_LOC<---FF H
SNO<---SNO + 1
STD_LOC<---
STD_LOC +(SNO *07)
!!!
No
STORE (R1) IN
(STD_LOC)
STD_LOC<---
STD_LOC +1
PTR<---PTR + 1
!!!

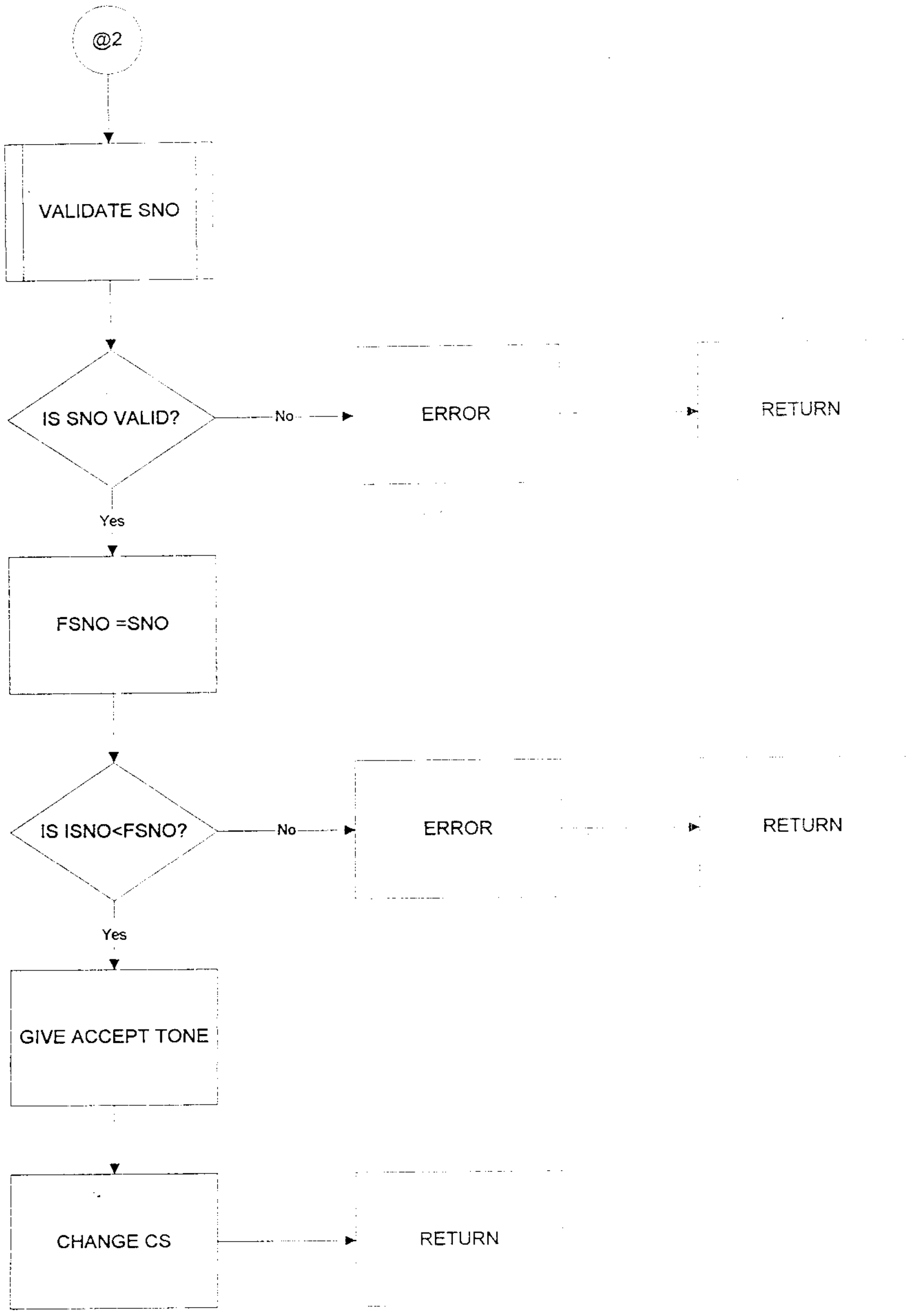
PRINTING

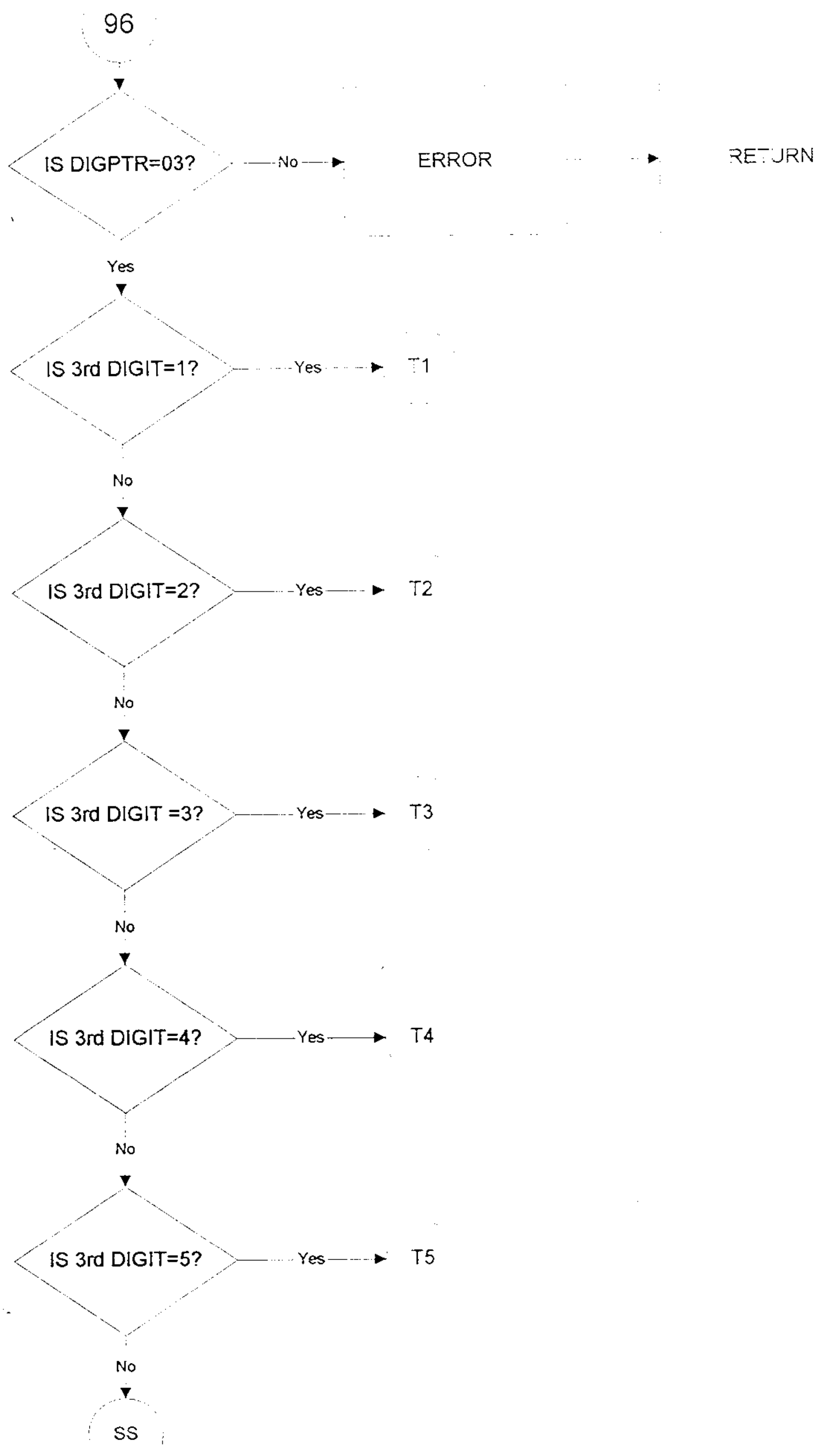

951
ISNO=01
FS NO=B4
SET PRNTR ON
DISABLE SERIAL PORT
GIVE ACCEPTANCE
CHANGE CS
RETURN

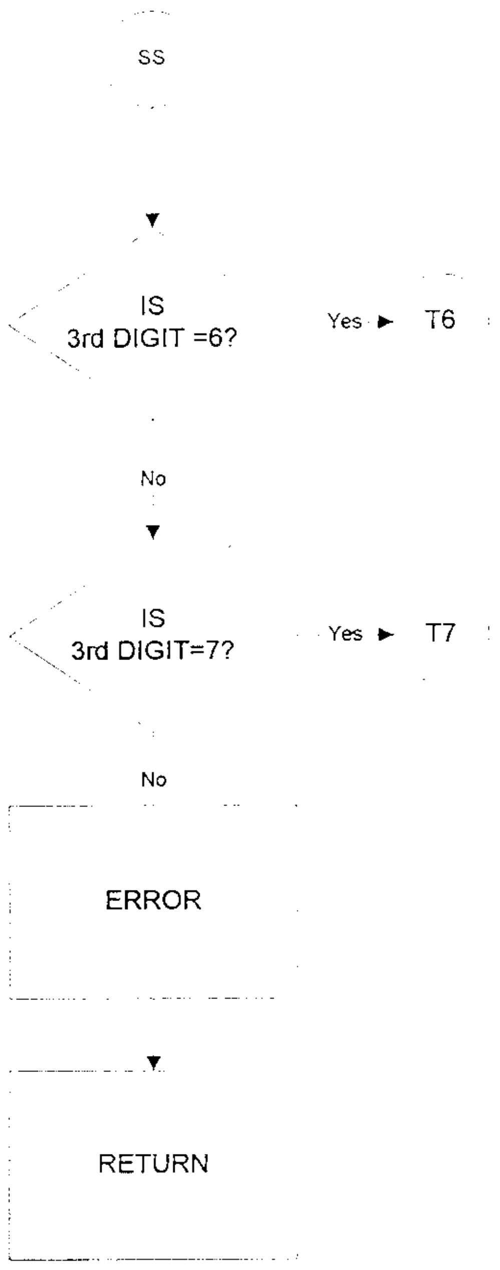


952
Yes
IS DIGPTR=6
No
ERROR
RETURN
Yes
VALIDATE SNO
IS SNO VALID?
No
ERROR
RETURN
YES
DEC HEX
I SNO=SNO
F SNO=B4
CHANGE CS
RETURN

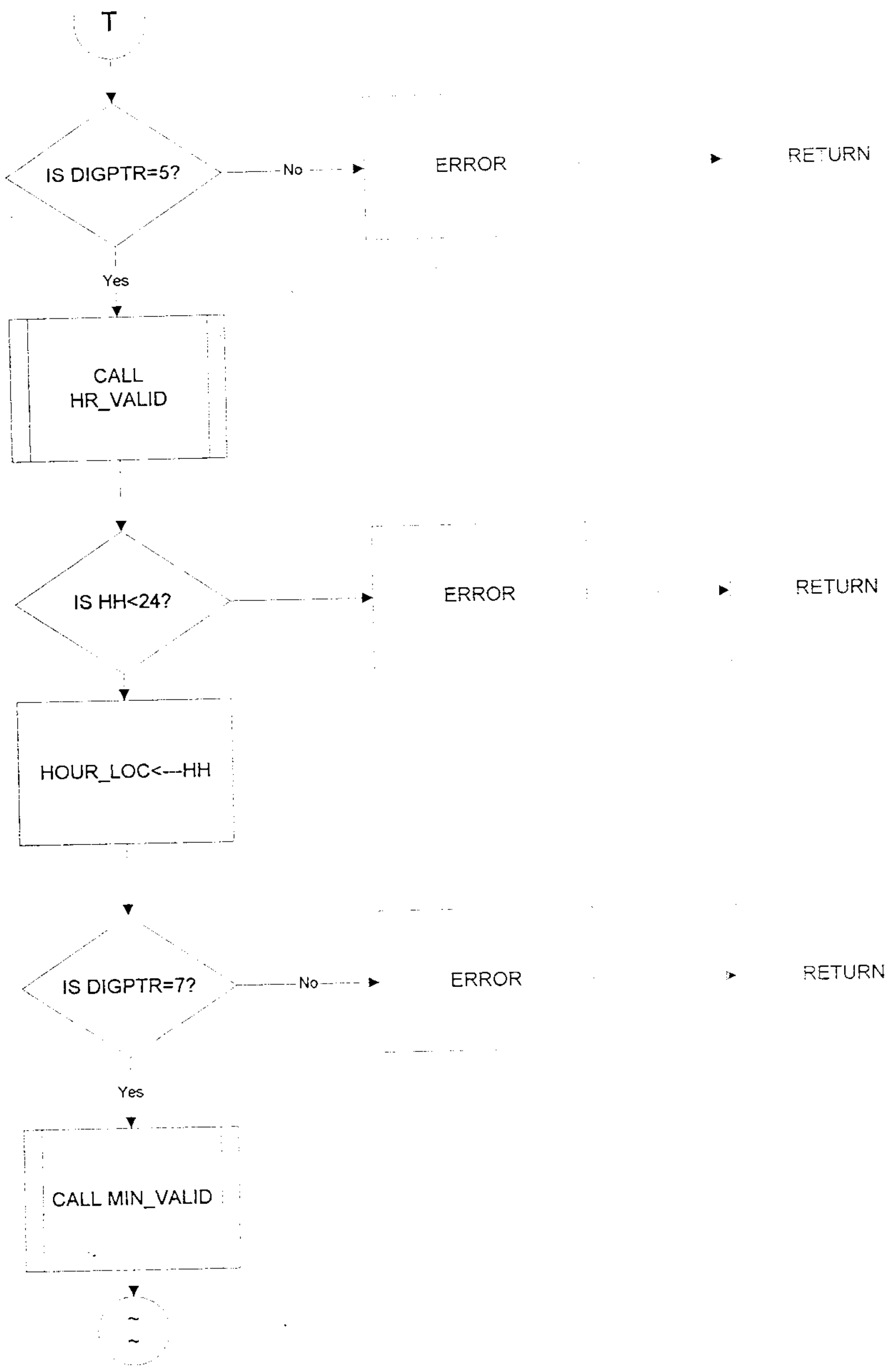

953
IS DIGPTR=6?
No
ERROR
RETURN
Yes
VALIDATE SNO
IS SNO VALID?
No
ERROR
RETURN
Yes
ISNO=SNO
IS DIGPTR=9?
No
ERROR
RETURN
Yes
@2

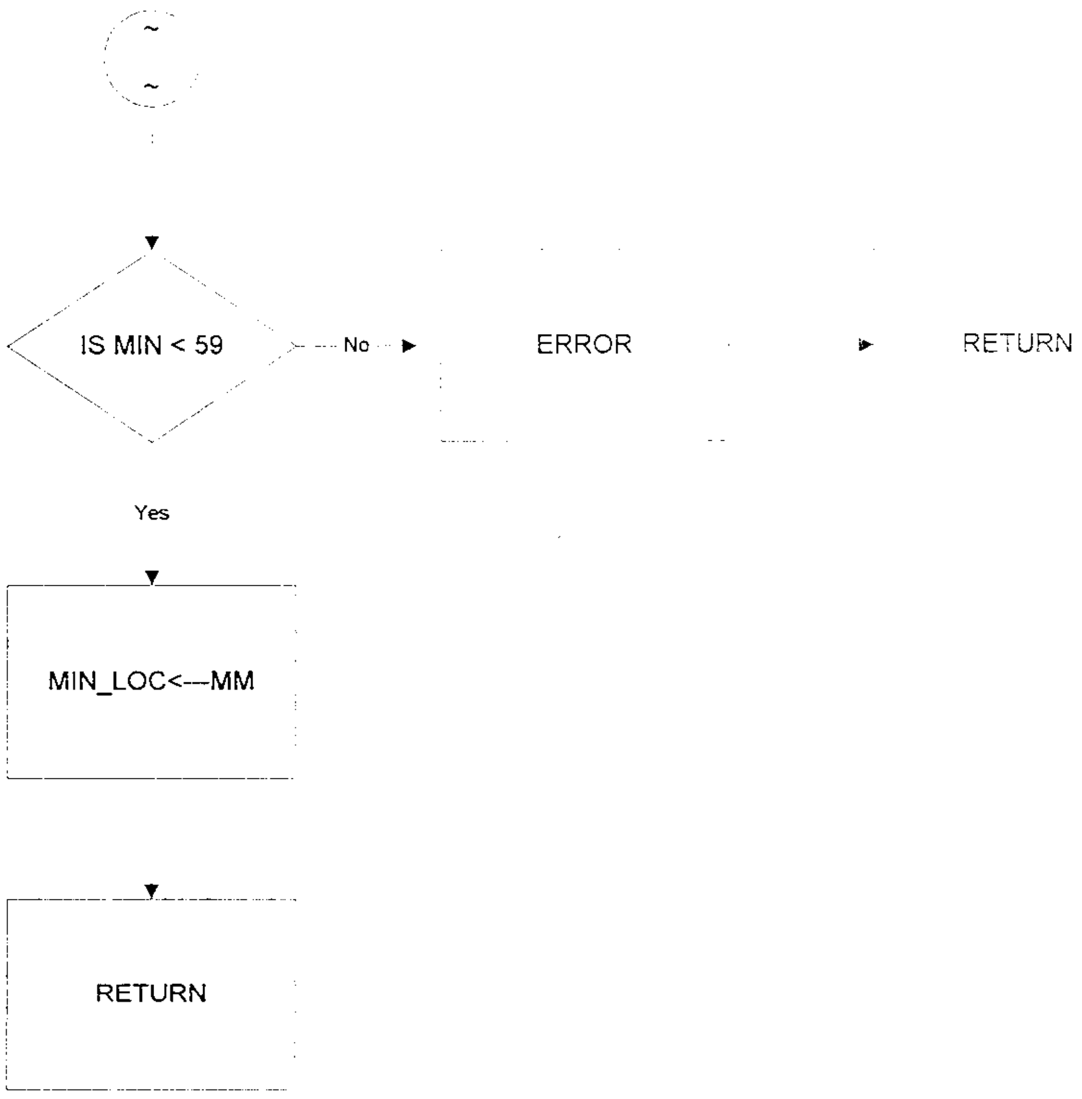

@2
VALIDATE SNO
IS SNO VALID?
No
ERROR
RETURN
Yes
FSNO =SNO
IS ISNO<FSNO?
No
ERROR
RETURN
Yes
GIVE ACCEPT TONE
CHANGE CS
RETURN


96
IS DIGPTR=03?
No
ERROR
RETURN
Yes
IS 3rd DIGIT=1?
Yes
T1
No
IS 3rd DIGIT=2?
Yes
T2
No
IS 3rd DIGIT =3?
Yes
T3
No
IS 3rd DIGIT=4?
Yes
T4
No
IS 3rd DIGIT=5?
Yes
T5
No
SS


SS
IS
3rd DIGIT =6?
Yes
T6
No
IS
3rd DIGIT=7?
Yes
T7
No
ERROR
RETURN

TIME ZONE ENTRY

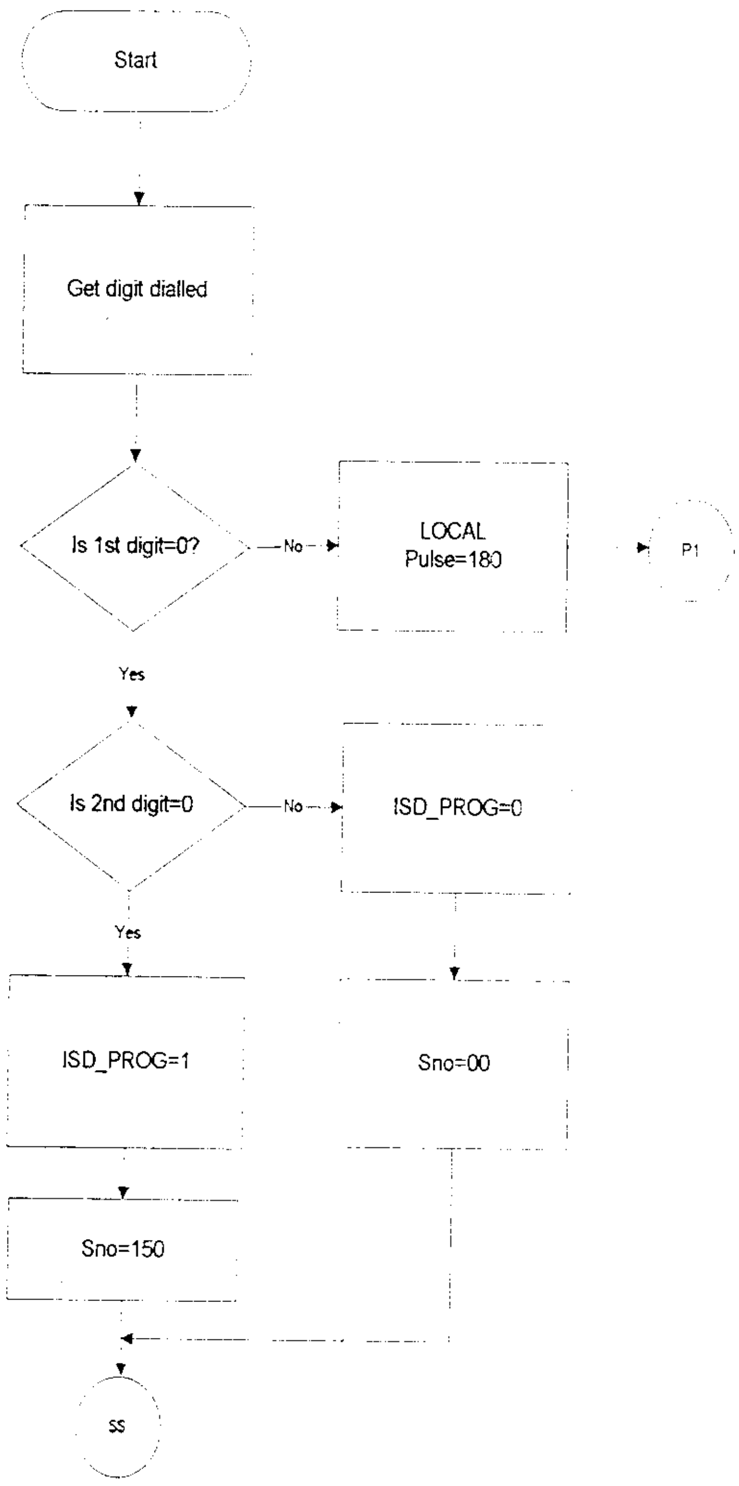

T
IS DIGPTR=5?
No
ERROR
RETURN
Yes
CALL HR_VALID
IS HH<24?
ERROR
RETURN
HOUR_LOC<---HH
IS DIGPTR=7?
No
ERROR
RETURN
Yes
CALL MIN_VALID

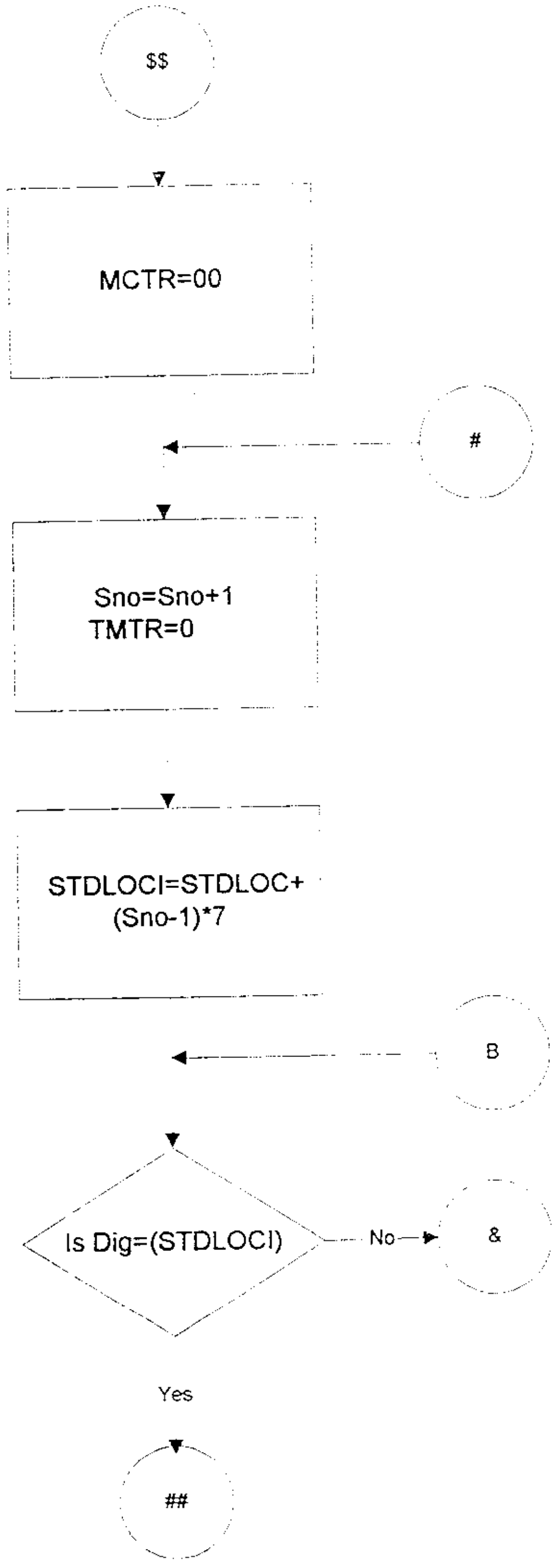


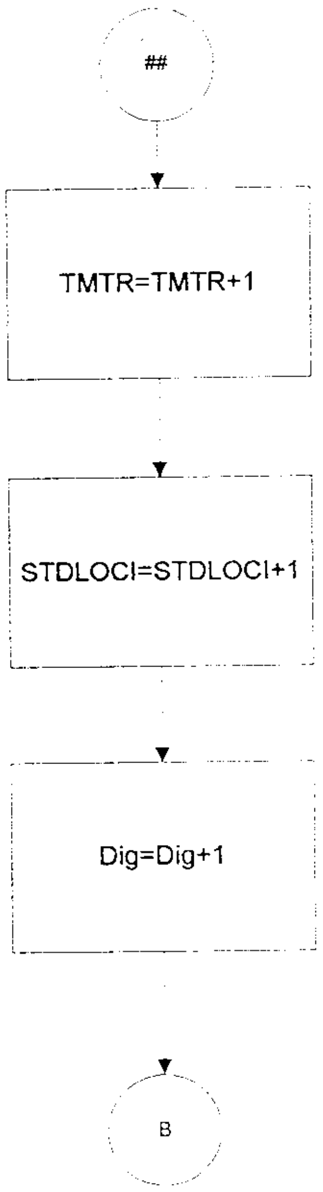
COMMENT: THIS OPERATION IS REPEATED FOR T1 to T6 WITH THE HOUR_LOC & MIN_LOC BEING DIFFERENT FOR EACH OF T (TI to T6)

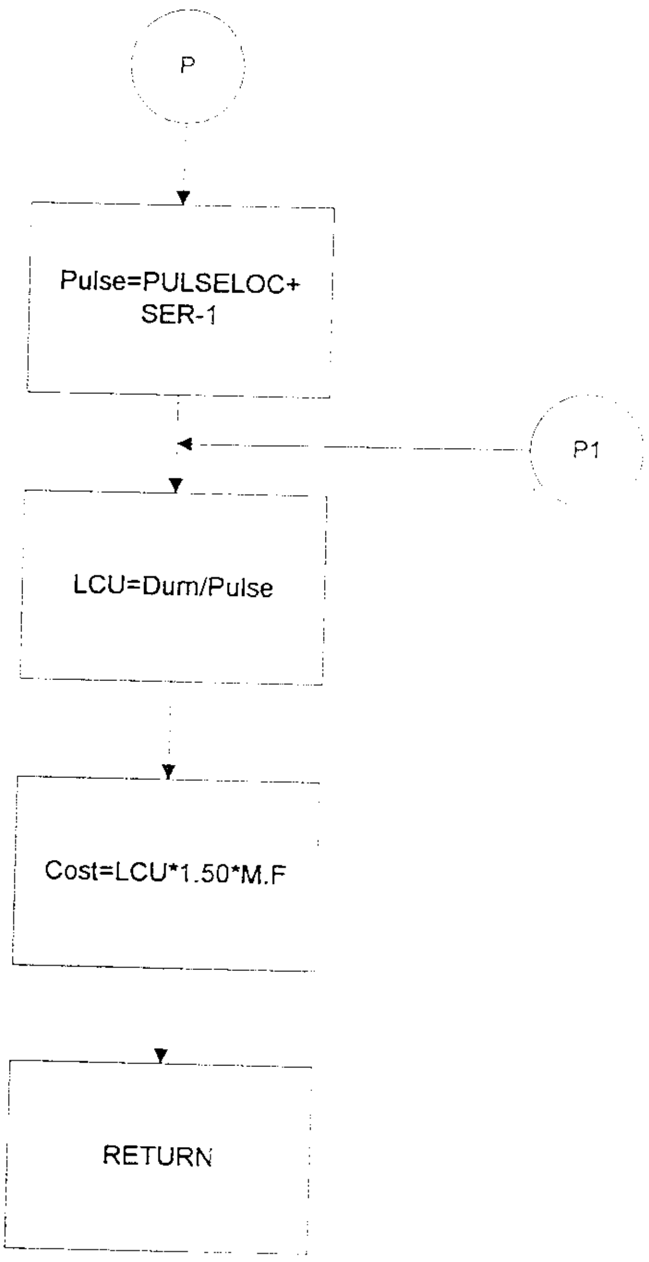
COST CALCULATION

COST CALCULATION




$$
#
MCTR=00
Sno=Sno+1
TMTR=0
STDLOCI=STDLOC+ (Sno-1)*7
B
Is Dig=(STDLOCI)
No
&
Yes
##


##
TMTR=TMTR+1
STDLOCI=STDLOCI+1
Dig=Dig+1
B


P
Pulse=PULSELOC+ SER-1
P1
LCU=Dum/Pulse
Cost=LCU*1.50*M.F
RETURN

CHAPTER - 10

TESTING AND VALIDATION

TESTING & VALIDATION

The ALP is written using 8051 micro-controller, in a DOS editor (NE). The program is saved as a ASM extension file in DOS. This .ASM file is converted into a .OBJ file using a software developed by 2500 A.D. The .OBJ file is converted into .LNK file and debugged using 8051 assembler. The debugged file is then converted to .HEX file and stored in the EPROM.

In order to test, consider that 91 has been dialled followed by the entry of serial number, pulse rate and STD codes. We have to check whether the software has stored these details. This can be done by inspecting the memory locations where these details are stored. The hard copy of codes and pulse rates can be obtained by enabling the SMDR and the printer.

Similarly we test all the different ways of entry of the STD codes, pulse rates. All the printing modes are tested by taking hard copies of the stored values using all the printing methods made available in the software.

The cost calculation program is also tested in the same fashion. We give values at storage locations and test for the accuracy of cost. Real time testing is also done by dialling different locations. This value of the cost is compared to the PC calculated cost. The difference between the costs are analysed and we check for inaccuracies. If the differences are very large, we rectify them by increasing the precision.

CHAPTER - 11

CONCLUSION AND FUTURE DEVELOPMENTS

CHAPTER - 12

BIBLIOGRAPHY