Consiglio Nazionale delle Ricerche

STELLA
SATELLITE TRANSMISSION EXPERIMENT LINKING LABORATORIES

DOCUMENT NUMBER WP/1/78
(WORKING PAPER)

Implementation of STELLA on a minicomputer - the PDP 11

Edited by: L. Lenzini

143

CNUCE

Edited by: Luciano Lenzini

STELLA

SATELLITE TRANSMISSION EXPERIMENT LINKING LABORATORIES

DOCUMENT NUMBER A/1/78

ARCHITECTURE AND IMPLEMENTATION OF STELLA

F. Caneschi, N. Celandroni, L. Lenzini,
E. Perotto, E. Zucchelli
CNR-Istituto CNUCE

April 1978

INTRODUCTION

High energy physics (t)is, at the same time, a highly centralized and a widely spread out activity. Although experiments are conduced and large quantities of data are collected at the very few big centres which have accelerators, the experimenters, some 2,000 in Europe, come from about 150 different Universities and Institutes throughout western Europe. Experiments are carried out as a result of collaborations, usually between several different Universities, and much of the planning and data analysis is performed away from the accelerator centres. This situation makes University and home life often very difficult for the physicists, and is costly in time and money. The heavy computing effort needed both for the planning and the analysis of experiments is made unnecessarily difficult by the absence of good and economical data transmission services between Universities and Laboratories in different countries. This leads to much travelling, to duplication of programming, to problems due to computers of the right kind being in the wrong place, and to time lags in communicating results of data analysis.

Important gains in the efficiency of conducting high energy physics experiments might be achieved if a specialized high speed (approx. 1Mbit/sec.) service were available, to directly interconnect the accelerator laboratories with the other large centres which have powerful computers where large quantities of data could be stored. The speed of 1Mbit/sec. allows the transfer of a tape full of data in just a few minutes, thus making it possible operations in collaboration between different centres on significant samples of data on the same time scale as would be possible within a single centre. This would effectively provide visiting groups with access to their home computers during their experiments, and could also help in load sharing between different national centres.

In order to obtain technical experience in this type of service, and also to acquire some of the information necessary for a serious analysis of its costs and benefits, CERN and four major European national high energy physics centres, INFN in Italy, Saclay in France, Rutherford Laboratory in the U.K. and Desy in Federal Republic of Germany, reacted with considerable interest to the European Space Agency's (ESA) suggestion to utilize the Orbital Test Satellite to provide an experimental service of the type described. The minimum service necessary for this experiment would be one that allows high speed transmission of bulk data and data samples, both on magnetic tape, from a small

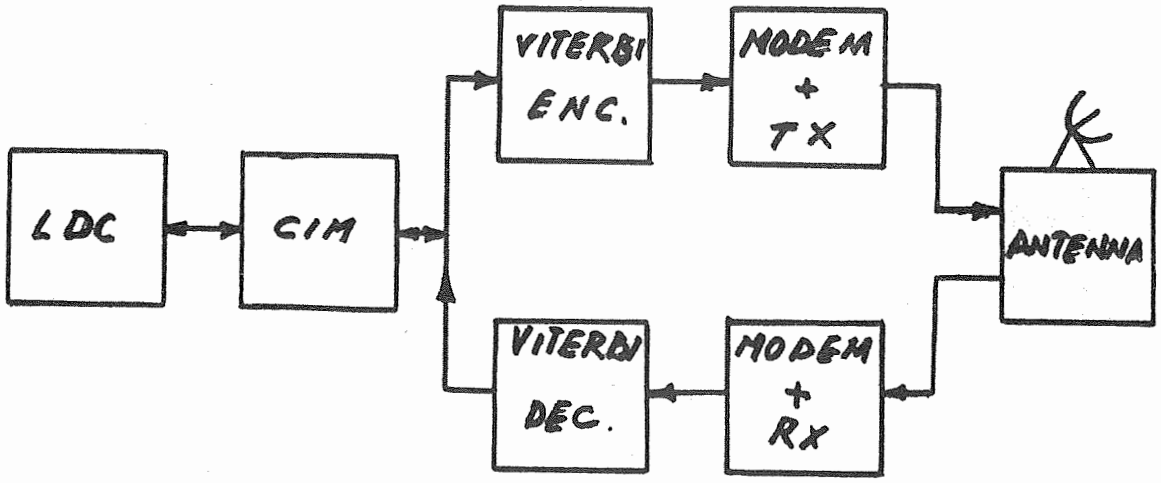

LDC
CIM
VITERBI
ENC.
MODEM
+
TX
ANTENNA
VITERBI
DEC.
MODEM
+
RX

Fig 1

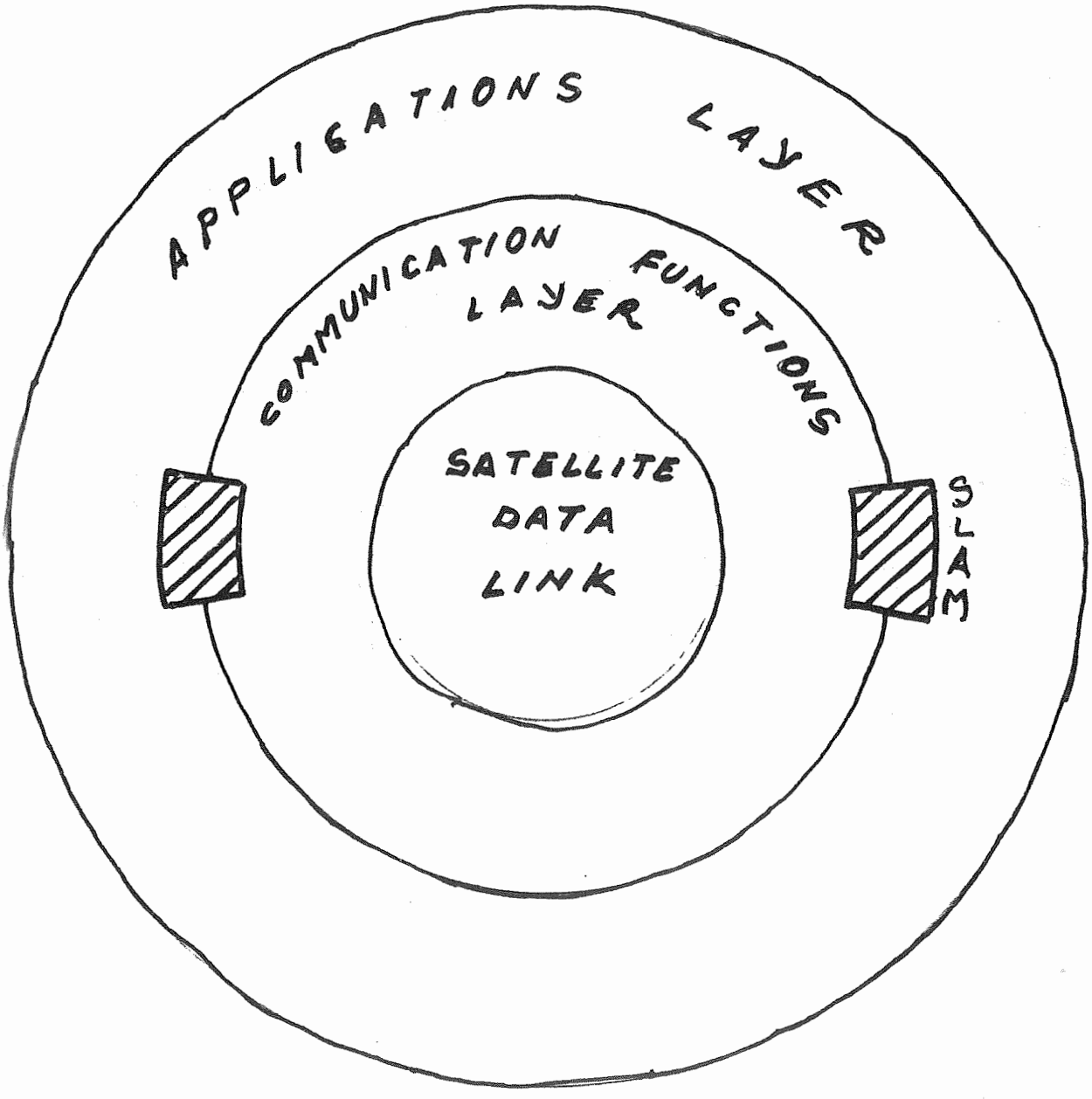

APPLICATIONS LAYER
COMMUNICATION FUNCTIONS LAYER
SATELLITE DATA LINK
SLAM

Fig 2

stand-alone computer at CERN, the major source of data, to the other four centres. An experimental service of this type would be extremely valuable in allowing ESA, the high energy phisics Laboratories and national PTTs to check the design and performance of the satellite earth stations, to consider problems of interfacing the earth stations with computers, to check the error rates obtained, to study different access and control procedures, etc. While both ESA and the PTTs would concentrate on extracting technically and industrially useful information, and the Laboratories would study the implications for users in general, and would analyze the possible costs and benefits for high energy physics in particular.

This project is known as STELLA project and the Italian part is being carried out by a collaboration among the following institutions:

CNUCE-Institute of the National Research Council (CNR)
INFN-Italian National Institute of Nuclear Physics
TELESPAZIO

The activity of this Italian collaboration can be divided into three fundamental areas:

a) Study and implementation of a generalized Satellite Communication System (SCS). This System is a result of the union of software subsystems and hardware components distributed through all the National Laboratories. These subsystems and components interact and cooperate through the Satellite Data Link.

b) Definition and implementation of an access method to the Satellite Communication System, SLAM (Satellite Link Access Method), with the objective of broadening the spectrum of Applications accessing the Satellite Communication System.

c) Implementation of the tape transfer service with the objective of both providing a user friendly access to the Satellite Communication System services for inexperienced and to gather real life informations on the performance characteristics of STELLA, thus validating its functional design.

1. SYSTEM CONFIGURATION

The system hardware configuration for all the Laboratories involved in the STELLA project is shown in fig. 1.

1.1 Link Driving Computer (LDC).

There are two LDCs. Their basic function is that of providing the communication between Application programs, using a very simple end-to-end protocol. In the initial part of the STELLA experiment, each LDC runs only a single Application program. There is no routing problem (towards the Application) in this case, even if Application identification is required because different Applications can run at different time.
Link Driving Computers at each installation are connected to the earth station via the Computer Interface Modul (see below).

1.2 Computer Interface Module (CIM)

The CIM (Computer Interface Module) hardware is a full-duplex data communications interface between each LDC (Link Driving Computer) and its associated earth station. It provides eight standard CCITT circuits: TX (103), RX (104), RTS (105), CTS (106), DSR (107), TSET (114), and RSET (115) for modem control by the LDC. It also offers essentially automatic HDLC framing/deframing, bit insertion/deletion and CRC checking of all data blocks transmitted/received by the LDC, with associated status words to allow the LDC to monitor operations and intervene for error-recovery.

CIM contains a microprocessor controlling a special-purpose "COM" chip for the HDLC function. (Actually two COM chips are used to simplify the design, one for transmitting and one for receiving). The microprocessor handles initialization of the chips, transmission of enough HDLC frame flags before and after each block to satisfy modem (and Viterbi) hardware requirements, and provides end-of-block interrupt and status to the LDC. Within each HDLC frame, data flows directly between the LDC memory and the modem via the COM chips.

The COM chip (COM 5025) is a LSI MOS circuit manufactured by Standard Microsystem Corp., New York. It basically handles automatic bit insertion/deletion, CRC generation/checking, framing/deframing and serial/parallel - parallel/serial conversion. It also

provides a variety of status and control functions.

1.3 VITERBI encoder/decoder

Encodes/Decodes bit stream in an error correcting code.

1.4 MODEM

Performs modulation/demodulation. Supplies clock signal.

1.5 TX

Transmitter

1.6 RX

Receiver

1.7 Antenna.

The hardware interface betwen CIM (M6800) and LDC is reported in Annex 1. This interface is built on the assumption that the LDC is a PDP 11 minicomputer (Italian example). Annex 2 describes specifications of the software interface for accessing the CIM functions from an Application program running under the PDP 11 operating system (RSX 11M) used in the Italian environment.

2. EXPERIMENTAL SOFTWARE

The STELLA system is conceptually divided into two layers. The inner layer, the Satellite Communication System, provides facilities for transmitting and receiving packets between two Application programs. The outer layer consists of Application programs that use the Satellite Communication System to communicate with each other. These two layers each have their own protocols for communication, built upon the protocols provided by the inner layer. In addition, the functions of a layer may involve several protocols.

The term Application is used to indicate the generic end user. More precisely, an Application is defined as any process or set of coordinated processes that access the Satellite Communication System in order to obtain services. Towards the external boundary, Applications can be directly attached to one or more of the end user mechanisms (physical devices) which represent the ultimate sources or destinations of information. These devices, if present, are also included in the conceptual frame of the term Application. For example a user terminal and the code supporting I/O operations on the processor to which the terminal is connected are considered to be a single process of an Application. By this definition any information exchange across the Satellite Communication System must take place between Applications.

3. SATELLITE COMMUNICATION SYSTEM

The Satellite Communication System provides the packet transmission capability by using a physical communication medium together with a well defined packet format.

As shown in fig. 2 the Satellite Communication System is structured into a Satellite Data Link (SDL) and a Communication Functions Layer.
The Satellite Data Link provides the physical communication medium capable of carrying data from one LDC to another one. The SDL is made up of a full duplex circuit called Connection. The SDL does not guarantee the integrity of packets.

The Communication Function Layer has the internal structure shown in figs. 3 and 4 depending on the role of the LDC. In these figures the modules which compose each of the processes described in the paper STELLA software are also shown.
The functions provided by each of the modules within anyone process can be deduced by looking at the flowchart given in Annex 3.

4. SLAM

SLAM (Satellite Link Access Method) is a set of macros which allows Applications to use the facilities provided by the Satellite Communication System. Before starting any transmission activity, an Application asks the Satellite Communication System for a Satellite Link Control Block (SLCB) using the SLAM macro instruction OPENSLK.
When an Application wishes to connect to another, it issues a CRQST (CALL REQUEST) operation.
An Application wishing to be connected by a CRQST request, must issue an INVITE operation. If the Application requested by the CRQST exists and has issued an INVITE, the two Applications become addressable without the intervention of the respective Satellite Services Managers (SSMs).
The SLH component carries out the text information exchange task by executing the SEND and RECEIVE macros issued by the Applications. SEND is used to send a packet, and the packet will be buffered at the receiving side (given sufficient storage) until a matching RECEIVE is issued by the Application at this side. A SEND is concluded as soon as the packet leaves the antenna. An asynchronous branch to the sending Application is made upon detection of packet losts (NACK received).
A MAIL is used for sending a Broadcast Message (restricted in length) from anyone Application to another.
The Broadcast Message is received in an asynchronous fashion, and no special operation by the receiver is necessary.
WSYNCH is used to synchronize the sender and receiver activities. The CTERM and CLOSLK macros are used respectively to release the "session" between the Application and the Satellite Data Link. More detailed information about SLAM can be found in Annex 4.

5. STATE DIAGRAMS

In designing STELLA, the layered architecture implied several layers of protocols. The various functions implemented in each level lead to the implementation of additional protocols which were kept relatively independent of each other. The various protocols naturally lead to a modularized approach. Early in the program design stage, it was observed that the various protocols of the Satellite Communication System could be described with state diagrams. It was observed that state of the SLCB could also be represented with a state diagram. The events that affected the state of th SLCB were messages from the Satellite Data Link, macro calls from the Application program, and actions from the operator. In general, each action was passed through a "filter" of validity checks before it was allowed to affect the state. In other words, the Application calls were checked for internal logical consistency and for consistency with the current state of the SLCB. Only after passing these checks were Application requests passed to the module that carried out the request.
Fig. 3 shows that the CALL/RETURN interface is one filter, and the ROUTER is another. Operator actions requested by the operator station are also checked for errors.
Fig.5 shows the major states of the Application layer, relative to an SLCB. Each portion is discussed separately below.
A state may have many substates, but these are normally relevant only to the module that handles actions within that state.
The states of Fig.5 are grouped as follows. The lower one third are states where the SLCB is in session. The middle third are used when going in and out of session.
Several of the states and transitions represent convenient but not essential features which have not been described in this paper. The remaining upper third is relative to the closing and opening of an SLCB under various conditions, including a Satellite Link going down indication.
The state diagram was an effective tool for the design of nearly every module in the PDP11, and was also useful for describing the more complex protocols of the network. In some cases, the state diagram was just an informal design aid; in other cases such as is shown in Fig.5, considerable information about the state of the SLCB was represented by the state diagram. Naturally, sometimes considerable computations are necessary for an event that affects the state of the SLCB, and that cannot be

represented on the state diagram. The diagram of Fig.5 is useful also as an aid in explaining the relationships of the SLAM operations to a writer of an Application. For example, it is easy to see that a particular SLCB cannot be used both for a MAIL and for an OPESLK at the same time. Of course, an Application may have only one LSCB in use at a time as desired. In the SLCB the state is represented by a small number. Naturally, there is much other information contained in the SLCB, and that is the information which is not represented directly in the state diagram. However, the maintenance of this extra information was sometimes facilitated by study of the state diagram, and nothing that transitions into and out of certain groups of states required certain operations to be done. For example, transition into the session states requires that the SSM call the SLH components telling them to initialize their work areas in the SLCB. When transitions leave the in-session state group, a similar closeout call is needed and it is also necessary for SSM to disconnect the ground station address from the SLCB. If the states are chosen carefully, and the various significant actions are associated with transitions in and out of (groups of) states, then the state diagram becomes an aid in verifying informally various logical behavior of the module the diagram describes.

Further observations can be made about state diagrams and their use as design tools. If a particular module is viewed in the state diagram, then the states themselves represent periods when the task is not in execution. This means that events arrive at a task one at a time and cause it to start execution in the same state as it was left in when it last stopped execution. In the PDP11 the interrupt level scheme of the hardware used is such that no event can arrive at the module for handling while the module is in execution. Events are queued, for all practical purposes, in the task scheduling queue. Thus, when a task stops executing, it becomes free to execute again based when an event significant to it occurs. The state information in the SLCB represents the information that is to be retained during periods when the module is not in execution. This viewpoint of the state diagram allows a useful informal design methodology. Again we will take the SLCB states as an example. A period when the task is not executing is pictured as a period when anything can happen. When designing a task and its states, must be kept this in mind, and a decision must be taken wheter any possible event should affect that module. If so, then this fould lead to additional events that affect the module in question. For example, in SSM, the observation that there could be a signal that the

satellite link is going down (from the operator interface, perhaps), caused the addition of a state to the design.

The design of the module at the flowchart level (informal high level language notation has also been useful) can proceed easily from the state diagram, if this has been developed in detail. One simply begins with an event and state combination and then decides on the computations necessary to proceed to the next event. If there are event/state combinations that are illegal, the combination may be an indication of a need for an error signal. For example, a CRQST operation can only be carried out starting from the open SLCB state. In all other states, the operation is rejected and returned to the Application with the appropriate return code. Often it is possible to discover different events that cause the same computations. As an example, there are several events which cause a session to be closed. These share several routines in common, although each has some unique features.

CONCLUSIONS

It is easy to observe that in the (PDP 11) implementation described here the various layers and modules reflect the various protocols of the system. It is quite natural to divide a system into modules in order to make the interfaces relatively simple, and so that the complexity is within the modules, rather than at interfaces between modules. What is not so obvious is that the converse is also necessary for a good system design. The design of the protocols and the major layers of the system was aimed at defining simple interfaces, whereby separate modules and possibly separate hardware could implement the various layers of the system.
The state diagram methodology can be followed at varying levels of formality, analogously to program description using flowcharts and similar design and documentation aids. The state diagram methodology has been used at levels of programming complexity ranging from the HSR, HST of fig.3 or fig.4 to various protocols in intermediate stages up to the combined use of the SLCB by SSM and SLH. The implementation of STELLA on the PDP 11 shown here illustrates the usefulness of this methodology for describing interactions of asynchronous events with the programs that must handle these events. It does this with more clarity than the more widely used documentation techniques such as flowcharting.

In fig.5 States are indicated by circles suitably named. Each transition between states is represented by an arrow having the following properties:

(1)-The tail of the arrow starts from the current state.

(2)-The head of the arrow ends at a circle which refers to the next state.

(3)-The input associated with the transition appears along the arrow.

(4)-If the next state is coincident with the current state, the transition arrow is represented as a loop.

REFERENCE

(1)-C. Adams, S. Dinwiddy and S. Steppel - Multiple Access to the Satellite Channel, and the computer to earth station interface - Document DD/MGNH/arh - 20 August 1976.

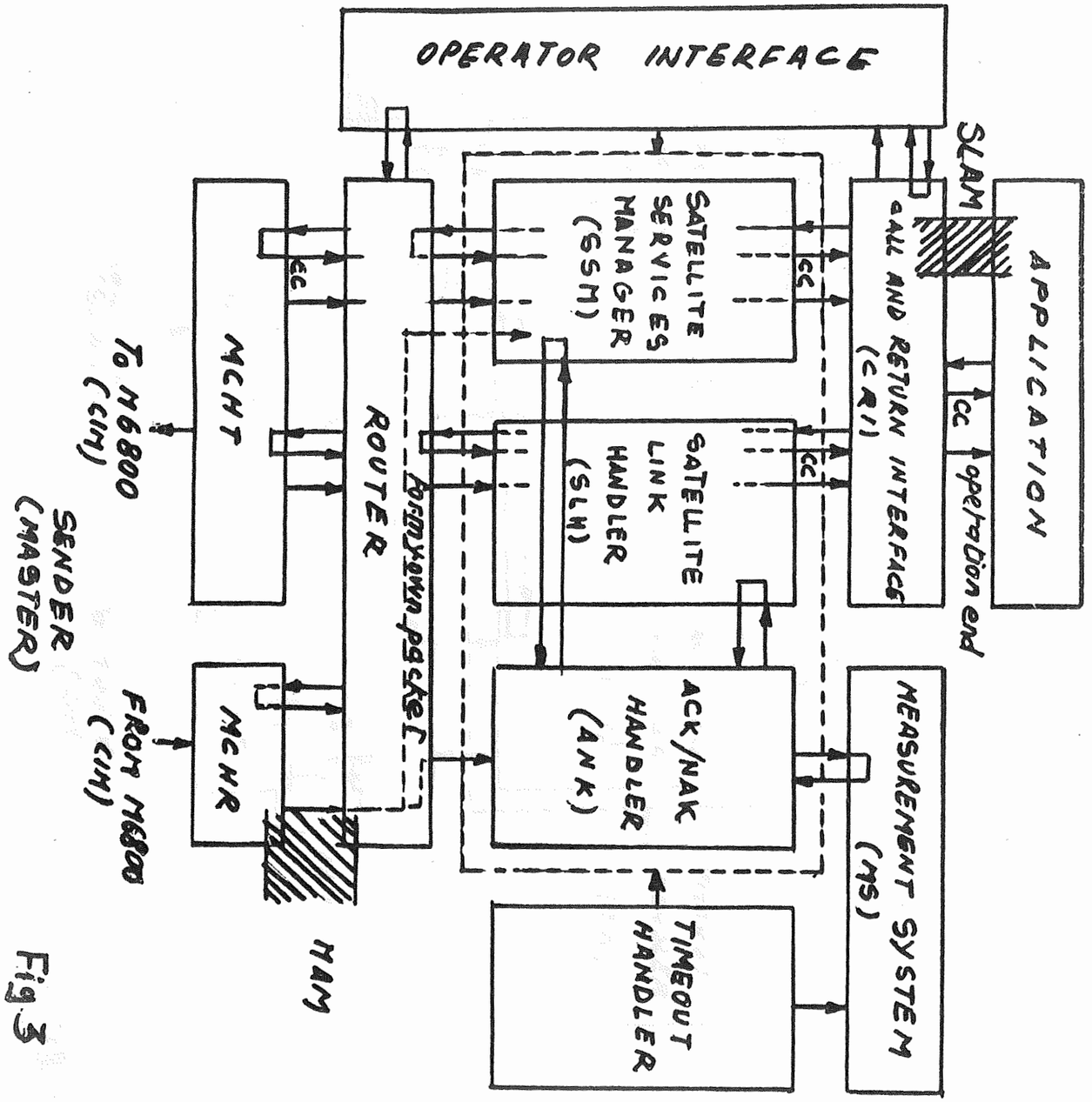


Fig 3

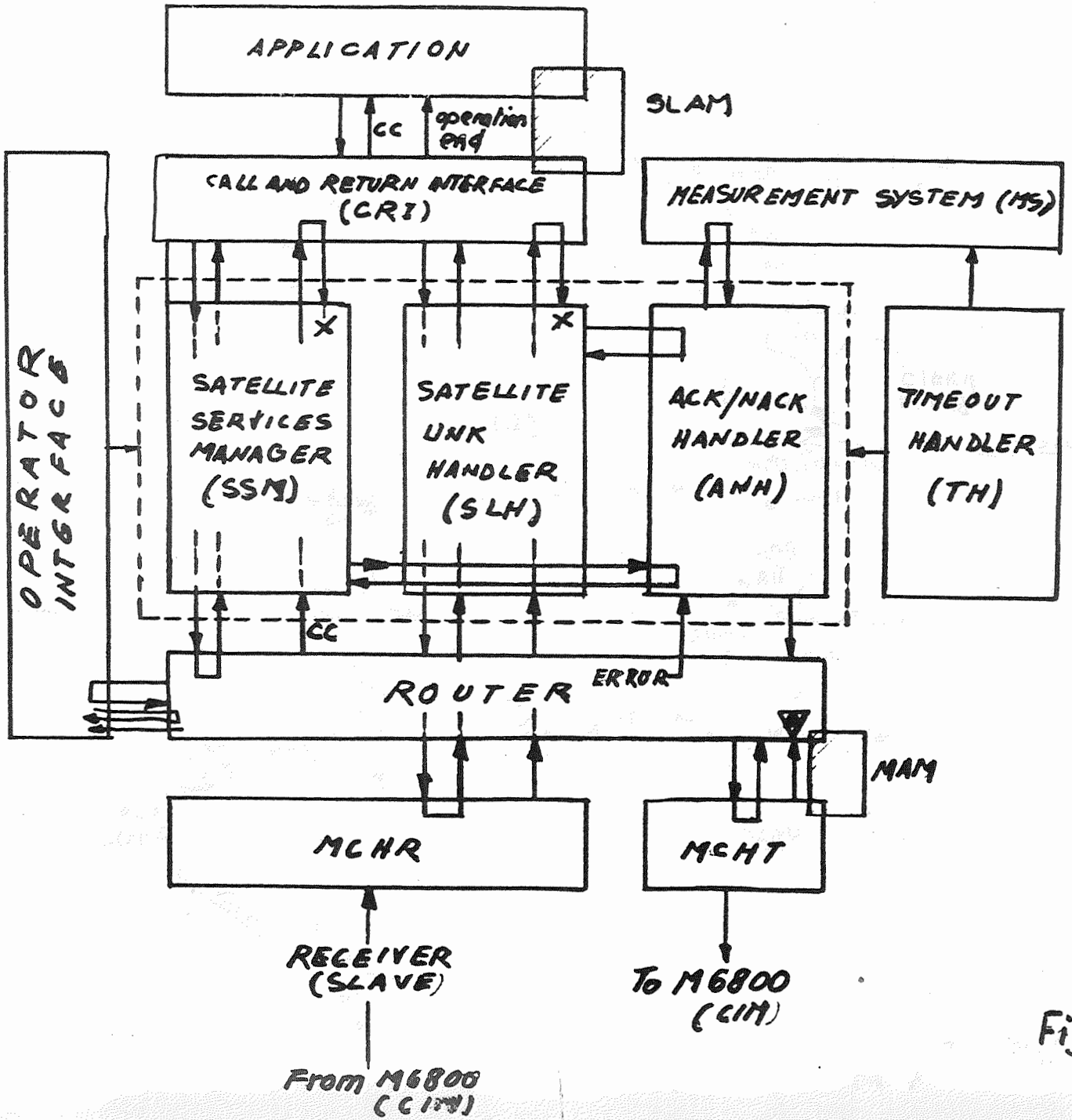
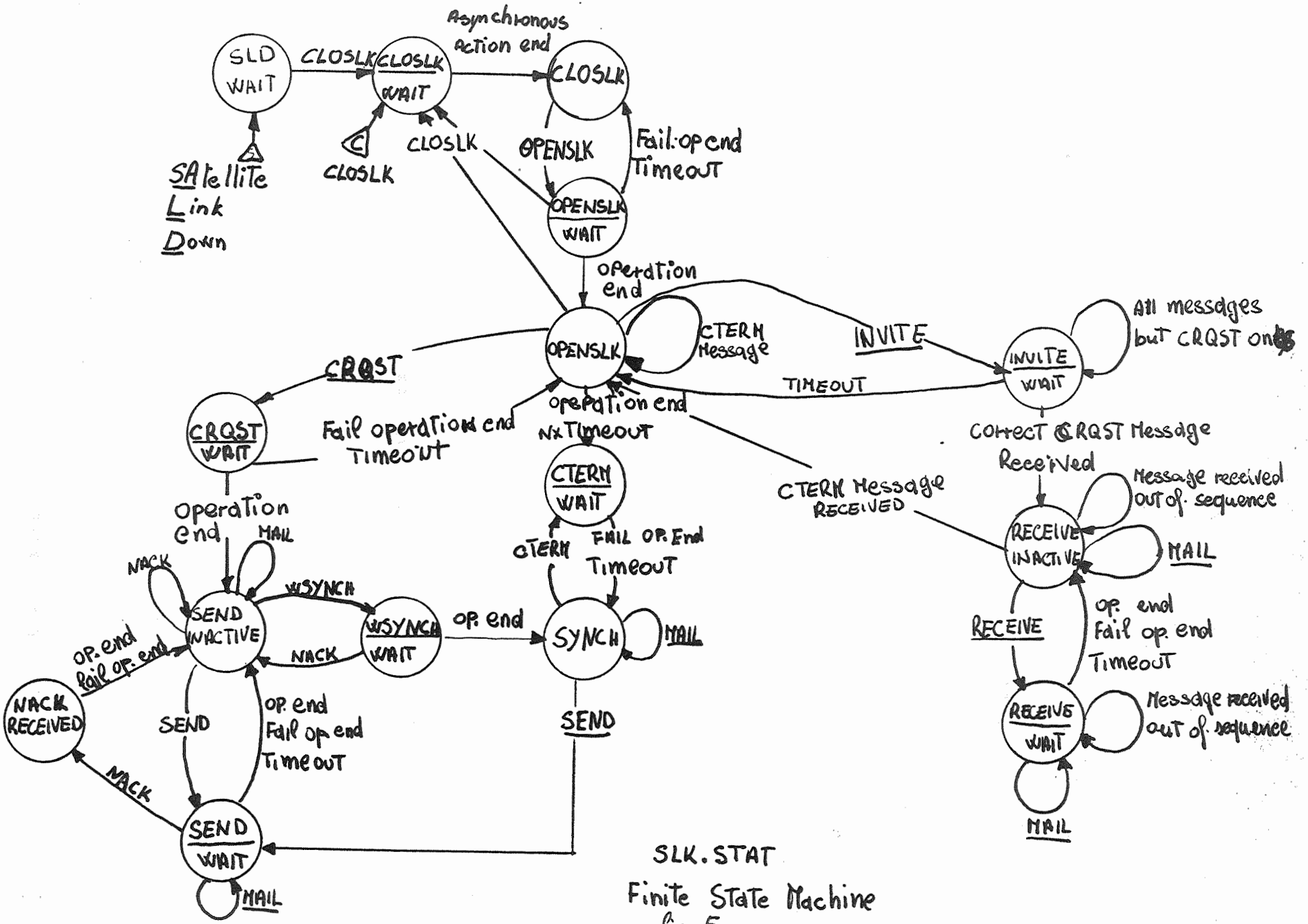

APPLICATION
SLAM
CC
operation end
CALL AND RETURN INTERFACE (CRI)
MEASUREMENT SYSTEM (MS)
OPERATOR INTERFACE
SATELLITE SERVICES MANAGER (SSM)
SATELLITE UNK HANDLER (SLH)
ACK/NACK HANDLER (ANH)
TIMEOUT HANDLER (TH)
CC
ROUTER
ERROR
MAM
MCHR
MCHT
RECEIVER (SLAVE)
To M6800 (CIM)
From M6800 (CIM)

Fig 4



SLK.STAT
Finite State Machine
fig. 5

STELLA

SATELLITE TRANSMISSION EXPERIMENT LINKING LABORATORIES

DOCUMENT NUMBER A/2/78

A GENERAL DESCRIPTION OF THE INTERFACE BETWEEN THE MOTOROLA M6800 AND THE PDP 11

P. La Vopa, A. Ranieri
INFN - BARI

April 1978

A GENERAL DESCRIPTION OF THE INTERFACE BETWEEN THE MOTOROLA M6800 AND THE PDP-11.

ABSTRACT

The following is a general description of how we think to implement the interface between the M6800 and the PDP-11.

The general registers are described as well as the relative flow of data and control signals.

INTERFACE

In FIG.1 is shown a block diagram of the special purpose interface. It is composed of the following registers:

CONTROL AND STATUS REGISTERS (CSR), it will contain the control and status of the system at any time;

MEMORY ADDRESS REGISTERS (MAR), there are two of them, one for reception and one for trasmission; they are used in DMA transfers and will contain the physical memory address where the data will be stored;

WORD COUNTER REGISTERS (WCR), as above, there are two of them, and they will contain the number of data transferred.

In order to calculate and adjust the difference in phase that will be needed between the Reception and Trasmission signals, there is another service register, the CLOCK COUNTER.

In the scheme, also, is shown the control circuitery to generate interrupts and NPR requests, to address the registers, to gate and transfer the signals.

THE INTERFACE OPERATION

Obviously, the main aim of the interface, here described, is to transfer data during a Reception or Trasmission operation.

It is organized to handle this at the same time; this means that two indipendent DMA operations will procede in parallel.

After an Initialization, during wich all registers are cleared, and the Start interrupt is enabled, the system goes in a Wait state.

At some point will arrive the Start interrupt generated by the COM logic. It will set a bit in the CSR, this will cause an interrupt request. Upon this request is granted by the PDP-11 processor, the control program will load the Reception DMA registers with the appropriate values, then it will set the DMA-enable bit in the CSR.

At this point the interface is ready to start with DMA transfers that will take place when the relative bit is set by the FIFO logic into the CSR.

When this happens a NPR request is begun, and the data is transferred into the memory.

This procedure will go on up to the end of the data burst.

Then a Data End interrupt will be generated by the COM logic.

At this point the control program, if no error occured, will write the data on the tape and will set everything to start the transmission of the ACK; i.e. will load appropriately the relative memory buffer and will send a Start Transmission Request to the M6800. When this request is granted, the interface will enable the DMA transfer, that will start as soon as the FIFO-Tx will be ready.

This operation will go on until the WCR-Tx will reach the zero value, then an End Transmission interrupt is generated either to the PDP-11 as to the M6800.

If there was an error during the reception, the interface will transmit a NACK, but the procedure will be the same as before.

So a complete cycle of the interface was described; obviously the sequence should be different in the real operation, because the reception and transmission are handled in parallel.

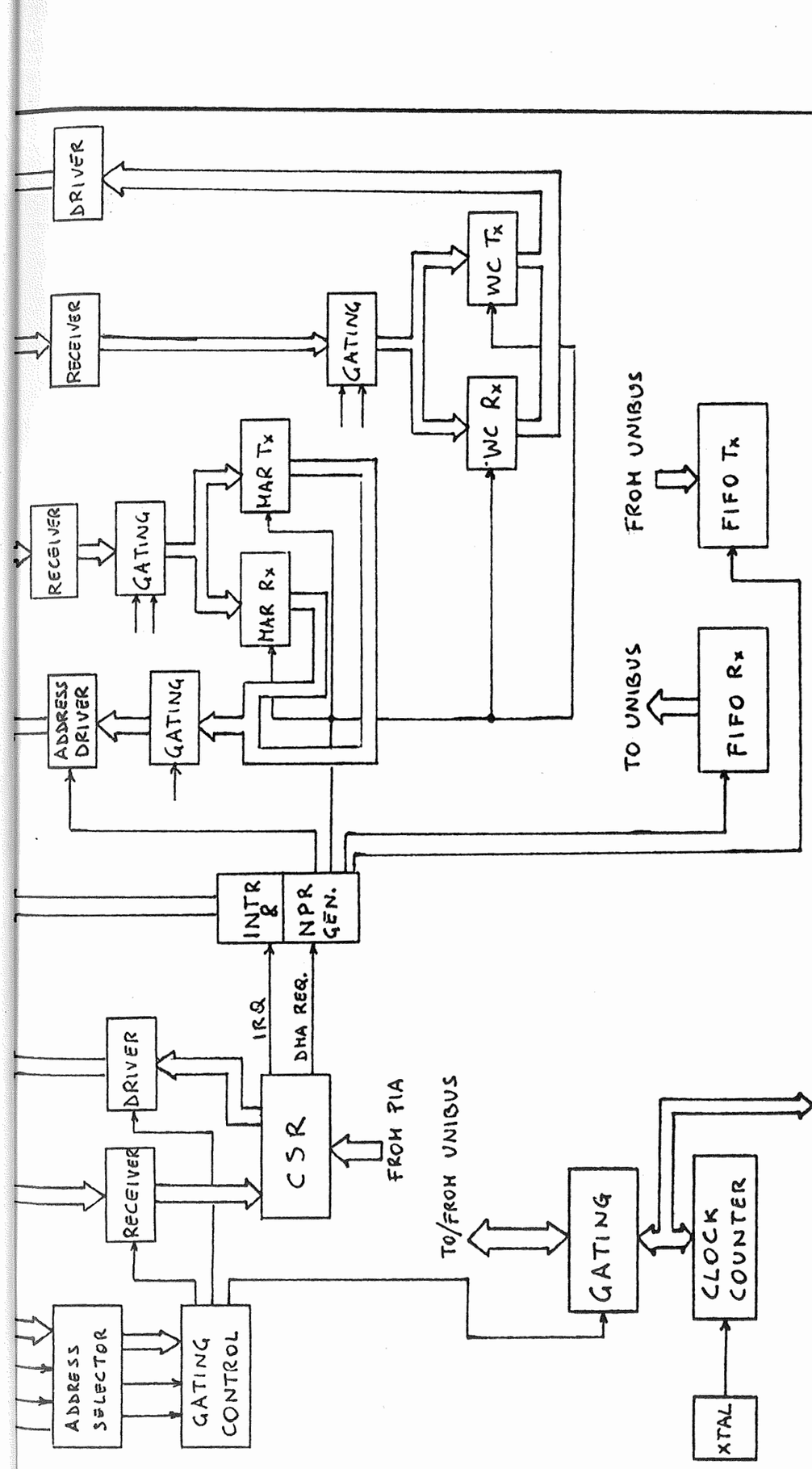


FIG. 1

ANNEX 2

STELLA

SATELLITE TRANSMISSION EXPERIMENT LINKING LABORATORIES

DOCUMENT NUMBER A/4/78

MAM
MICROCOMPUTER ACCES METHOD

F. Caneschi, M. Celandroni, L. Lenzini,
E. Perotto, E. Zucchelli
CNR-Istituto CNUCE

April 1978

RSX 11M General Device Software Interface
April 1978

RSX 11M programs involved with data transfer have some common requirements. There currently exist many device handlers, or "access methods", which contend with the scheduling and peculiarities of specific I/O devices, but that in turn present their own peculiarities to programs which make use of them. In order to ease the later addition of functions, it seems reasonable that a generalized software interface should be defined which is as device independent as possible. Such an interface should allow development of network programs to proceed on a structured basis. For example, if a program wished to deliver a message of a given length to a specific terminal, it should be possible to make the request, and allow other layers of programming to output the message on a local 2741 or package it with other messages for transmission to other systems. Likewise, large volumes of input or output data should be able to be transmitted to or from a disk, TP line, or host CPU without the driving program concerning itself with the particular source or destination.

The following is a preliminary definition of such an interface, and it is hoped that this definition will evolve as requirements are recognized and design decisions are made. Presently, communication between a Using Program and a Device Handler should provide for at least the following functions:

(1) Request for allocation of a device by the Using Program.

(2) Response by the Device Handler about the request for allocation.

(3) Delivery of asynchronous device activity interupts by the Device Handler to the Using Program.

(4) Request by the Using Program for data transfer, including message data, character count, direction of transfer (to or from PDP 11 storage), etc.

(5) Delivery of synchronous responses to data transfer requests at their completion by the Device Handler to the Using Program.

(6) Request by the Using Program to terminate a data transfer in progress.

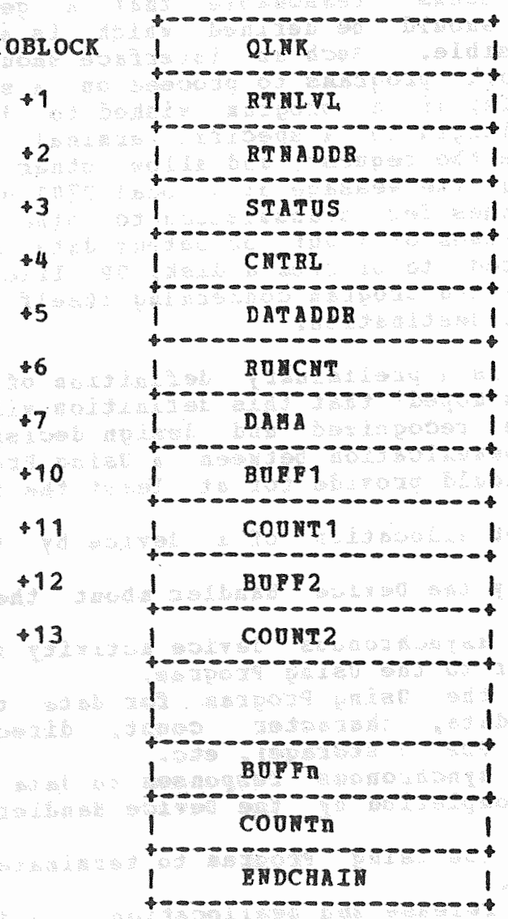
(7) Request for release and deallocation of a device by the Using Program.

Total device independence may be impossible or impractical; however, it is hoped that any dependencies will be isolated to well-defined areas. Implementation of such an interface for a specific device may be as simple as providing a small routine which receives general interface requests and translates them to appropriate requests to the RSX 11M support; Using Programs written to use the interface can make use of Device Handlers written at a later date with minimal, if any, changes.

It is proposed that each device handler provide three entry points of the form:

ALxxxxxx - Allocation Request
OPxxxxxx - Operation Request
CLxxxxxx - Clear Request

In calling each of these entry points, the Using Program should provide a return address in index register 7, and a pointer in index register 5 to a request list of the following form:



ALLOCATION REQUEST

In order for a Using Program to establish its use of a specific device it must first call the ALxxxxxx entry of the Device Handler with a 6-word IOLT (Input/Output LisT). This IOLT will remain the property of the Device Handler until the control is not given back to the Using Program. The informations contained in this IOLT will be used by the Device Handler to signal asynchronous conditions to the Using Program. The functions performed by ALxxxxxx is that of getting and initializing, for the device under way, a table called Task Control Block (TCB from now on) which will be used by the Device Handler during the various operations. The use of words within the IOLT is as follows:

QLNK
: This word is used by the system EMT mechanism; it is initially zero.

RTNLVL
: This contains the level on which the Using Program is to be notified, via the EMT mechanism, of asynchronous events.

RTNADDR
: This has the address within the Using Program to which control is given, via EMT, upon occurrence of an asynchronous event. When the Using Program gains control, its index register 5 will point to the QLNK field of the IOLT.

STATUS
: This word must be set initially zero by the Using Program. This word is set by the Device Handler to specify the type of asynchronous event which has occurred. (Status of X'0001' is reserved for signalling "normal interupt". Other status codes are currently device dependent.) The Using Program must clear the STATUS word before another asynchronous event may be passed to the Using Program by this means.

ALLTCBA
: If this word contains zero the TCB area has to be got dinamically within ALxxxxxx. If this word contains a value different from zero the TCB area is provided by the Using Program and the content of ALLTCBA is the TCB area address.

DAMA
: This word is set by the Using Program on call to ALxxxxxx to specify the requested device. This field will not be used by the Device Handler after return from ALxxxxxx, and may be used by the Using Program for any purpose.

Upon return from calling the ALxxxxxx entry point, the condition code will be set as follows:

CC=0
: The device has been allocated.

CC=1
: The allocation could not be carried out because there was no work area space available.

CC=2

The device is already allocated.

CC=3

The device requested is undefined or illegal, or the IOLT is incorrect.

OPERATION REQUEST

The Using Program can make requests for data transfers to or from an allocated device by calling the OPxxxxxx entry of the Device handler with register 5 pointing to a list called IOLT. A request may be executed, queued, or rejected at the discretion of the Device Handler. Upon completion of the request (if has not been rejected with a non-zero codition code) the IOLT is returned to the user via the EMT mechanism, with the STATUS word set to indicate the disposition of the operation. The use of the words in the IOLT is as follows:

QLNK

This is used by the system EMT mechanism. It must be initially zero.

RTNLVL

This word contains the level on which the Using Program is to be notified, via the EMT mechanism, of completion of processing of the operation, if the IOLT was not rejected initially.

RTNADDR

This word contains the address within the Using Program at which it is given control via the EMT mechanism, after processing of the operation is completed, if the IOLT was not rejected initially.

STATUS

This word is set by the Device Handler to specify the ending conditions of a request. Bits 0-7 and bits 8-15 contain two different logical pices of information whose meanings depends upon the the currently device or operation code.

CNTRL

This value is set by the Using Program on call to OPxxxxxx to specify the type of operation requested. There are several "standard" operations that have agreed upon, including read and write.

DATADDR

It will be used in the future.

RESBCNT

This word must be initialized to the number of bytes to be written by the Device Handler on write operation or the maximum number of bytes that the Device Handler is allowed to read from the device. When the operation is ended this word contains the residual byte count.

DAMA

This field contains the TCB address which refers to the device under way. This address is provided by the allocation step in the word ALLDAMA when the area is got by the User Program.

BUFF1, BUFF2, ..., BUFFn

These words contain the addresses of the buffers supplied by the using program for data transfer. Data characters are packed two per words. Buffer data containing an odd number of characters will have unpredictable data in the low order 8 bits of the last buffer word used.

COUNT1, COUNT2, ..., COUNTn

On data transfer from the buffer to the device, these words will contain the character count to be transferred. Before any I/O operation, on data transfer to the buffer from the device, a generic COUNT is assumed to designate the lenght of the buffer. Bit 0 of each COUNT specifies whether or not the buffer can be released by the Storage Management Supervisor Routines. More precisely if bit 0 is off the buffer can be released, if bit 0 is on the buffer cannot be released.

ENDCHAIN

This word follows the last count in the chain and must be always set to zero. Its function is to indicate the end of the chain.

The Using Program calls OPxxxxxx with register 5 containing the address of the IOLT, and register 7 set to the return address. All registers, except possibly register 6, are preserved by OPxxxxxx until an immediate return via register 7. Upon return via register 7, the condition code will be set as follows:

CC=0

The request has been accepted. The IOLT will be returned to the Using Program via EMT at the address and level given in the IOLT.

CC=1

The request has not been accepted; STATUS has been set to reflect the reason for rejection.

For all operations, the following values of status are valid upon immediate return.

STATUS = 2

The request has not been accepted because of the state of the device. The Device Handler may not be prepared to queue requests for a busy device, or initialization of the device may not be yet complete.

STATUS = 3

The request has not been accepted. The device requested is not valid, or other items in the IOLT are not valid.

When an IOLT is accepted, it is returned to the user via a EMT operation by the Device Handler. At that time, the Using Program only has the address of the IOLT in register 5, and he is given control according to the information in the RTNLVL and RTNADDR fields. For all operations, when they completed normally the status word is set to one; on the other hand the returned status contains a value which is device dependent.

Operations of reading, writing, and deallocation are described below, sometimes with additional meanings for the returned STATUS.

READ OPERATION

Entry Point: OPxxxxxx
CNTRL: X'0002'

Data is to be transmitted to the buffer from the device. BUFF1 and COUNT1 fields tell the Device Handler the buffer address in which to store the characters and, how much space, in characters, there is available for storing the data that is read in. After the read is done, COUNT1 field specifies how many characters were actually read into BUFFER1. If the space is not available a data chaining will be provided by the Device Handler. The same procedure as for BUFF1 and COUNT1 will then be followed for a generic BUFFn and COUNTn.

WRITE OPERATION

Entry Point: OPxxxxxx
CNTRL: X'0001'

Data is transmitted from the buffer to the device. BUFF1 and COUNT1 fields tell the Device Handler the buffer address from which to get the characters and, how much space, in characters, has to be written by the Device Handler. If there is more than one buffer a data chaining is provided by the Device Handler. The same procedure as for BUFF1 and COUNT1 will then be followed for a generic BUFFn and COUNTn.

INTERUPT GENERATION REQUEST

Entry Point: OPxxxxxx
CNTRL: X'8000'

This is a request for the device handler to generate an interupt type of action to alter the state of the device. Some examples are the generation of an interupt on the byte multiplexor interface for a 2702 Transmission Control Unit emulator, to simulate the pressing of an attention key, and the case where a 2741 is in transmit mode, and the user wishes to send the equivalent of a reverse break, to put it into receive control mode.

This operation is not implemented for all devices, and is rather device dependent. The STATUS field values are not defined yet.

DE-ALLOCATE OPERATION

Entry Point: OPxxxxxx
CNTRL: X'FFFF'

The device is to be deallocated. Upon return of the IOLT to the user by EMT, no more operations will be accepted by the Device Handler for that device until a Using Program allocates that device. At the time the deallocation request is processed, if there are any outstanding requests that have not been completed by the Device Handler, they are returned to the Using Program via EMT, with their STATUS code set to -4. Upon completion of the deallocation, the deallocation IOLT is returned to the user via EMT, as usual, with its STATUS field set to 1.

Note that it is up to the designer of the device handler whether this operation will be queued after the currently queued operations, and therefore will take affect only when they are done, or whether this operation will cause the previously specified operations to be terminated, with an appropriate STATUS code.

HIO REQUEST

Entry Point: HIOxxxxx

A Using Program may request that a previously specified operation be terminated early. The previously specified operation can only be specified by calling OPxxxxxx and it cannot be the ALxxxxxx operation. The Using Program notifies the Device Handler by calling it at HIOxxxxx, with the address in register 5 of the TCB which refers to the device to be halted. Register 7 should point to the return point in the Using Program. It may or may not find the IOLT in its tables. If it does not, or if there are other reasons, it may reject the HIO operation. In any case, the contents of the IOLT is never changed by the HIO operation.

The Device Handler has only the possibility of stopping the operation under way if there is one. In any case, the condition code is set by the HIOxxxxx operation upon return from the call by the Device Handler. Upon return from the call, the condition code is set as follows:

CC=0

Clearing of the requested operation is complete.

CC=1

The operation could not be found in the Device Handler tables. It possibly has already completed.

CC=2

The device specified in the DEVDEFN field of the IOLT is undefined. The clear operation is otherwise ignored.

STELLA

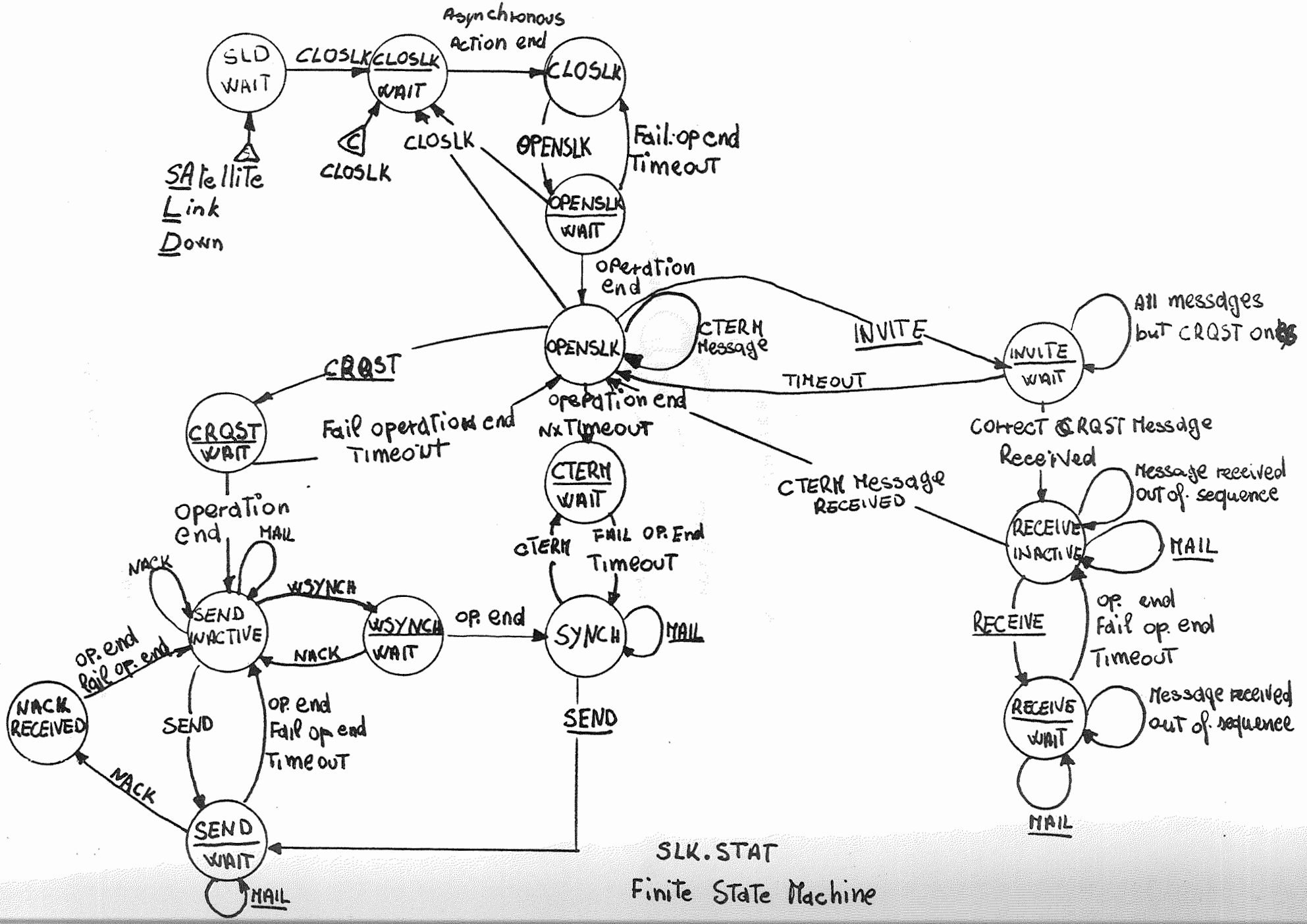
SATELLITE TRANSMISSION EXPERIMENT LINKING LABORATORIES

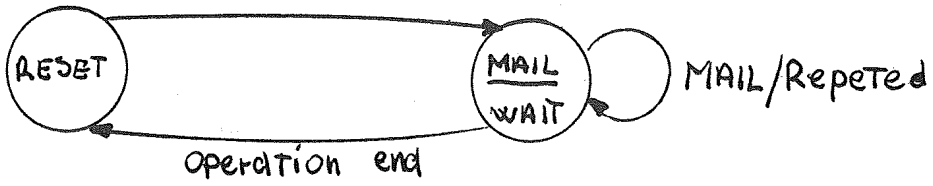
DOCUMENT NUMBER I/1/78

IMPLEMENTATION OF STELLA ON A MINICOMPUTER - THE PDP 11

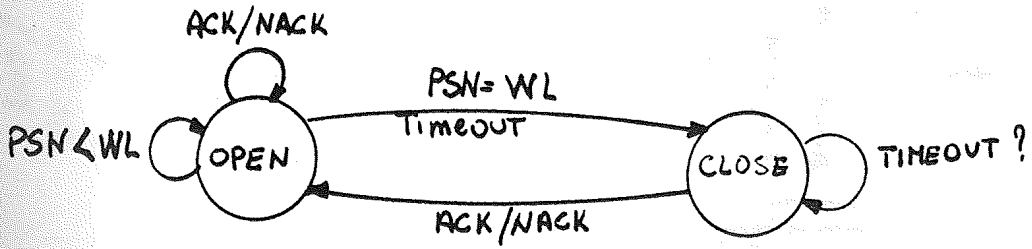
F. Caneschi, M. Celandroni, L. Lenzini,
E. Perotto, E. Zucchelli
CNR-Istituto CNUCE

April 1978

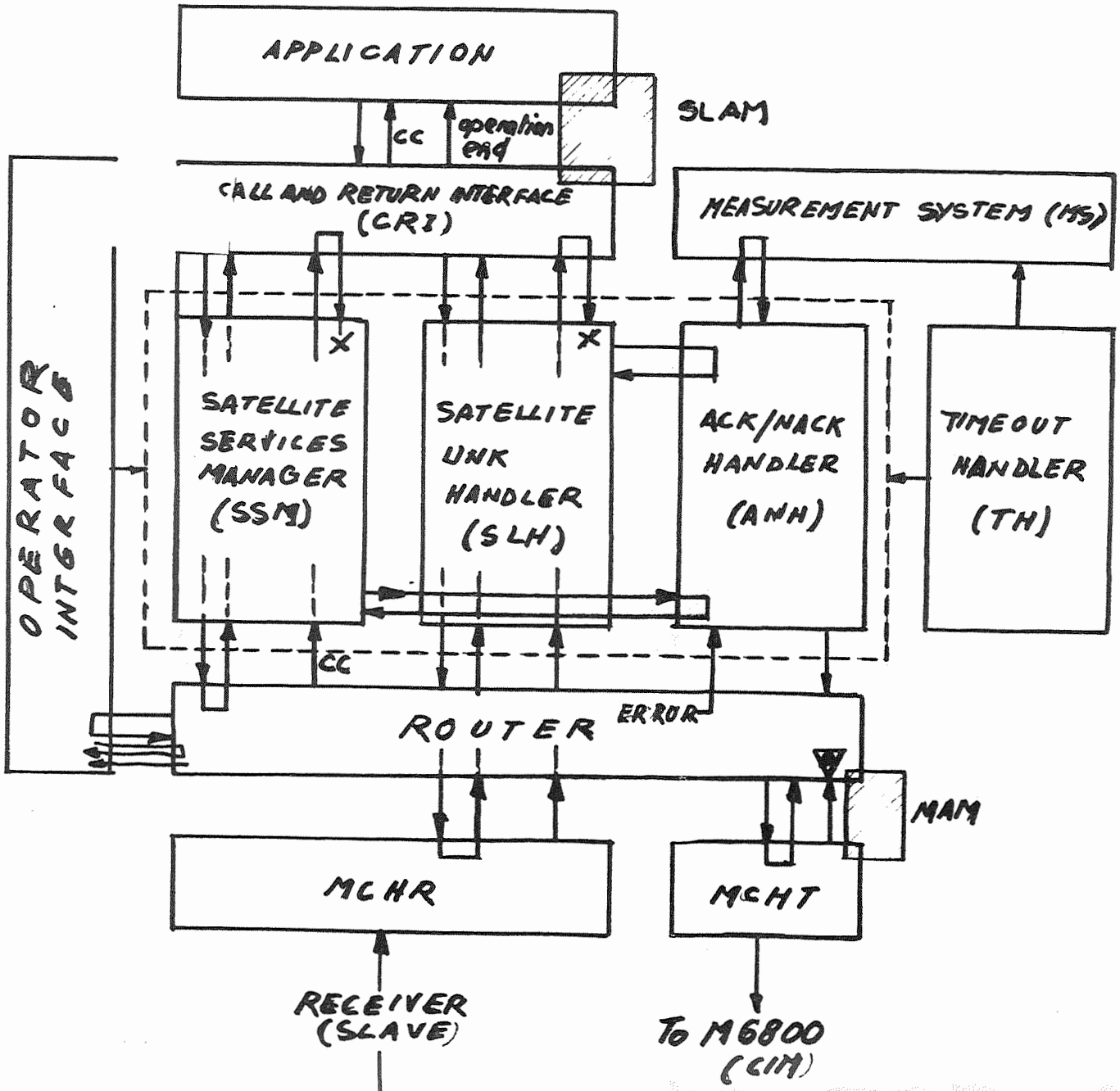

SLD WAIT
CLOSLK
CLOSLK WAIT
Asynchronous Action end
CLOSLK
SATellite Link Down
CLOSLK
CLOSLK
OPENSLK
Fail: op end Timeout
OPENSLK WAIT
operation end
OPENSLK
CTERM Message
INVITE
All messages but CRQST ones
INVITE WAIT
TIMEOUT
CRQST
CRQST WAIT
Fail operation end Timeout
operation end
NxTimeout
CTERM WAIT
Correct CRQST Message Received
CTERM Message RECEIVED
Message received out of sequence
RECEIVE INACTIVE
MAIL
operation end
MAIL
NACK
SEND INACTIVE
WSYNCH
WSYNCH WAIT
op. end
SYNCH
MAIL
CTERM
FAIL OP. End Timeout
op. end Fail op. end
NACK
NACK RECEIVED
SEND
op. end Fail op. end Timeout
RECEIVE
op. end Fail op. end Timeout
RECEIVE WAIT
Message received out of sequence
NACK
SEND
SEND WAIT
MAIL
MAIL
SLK.STAT
Finite State Machine



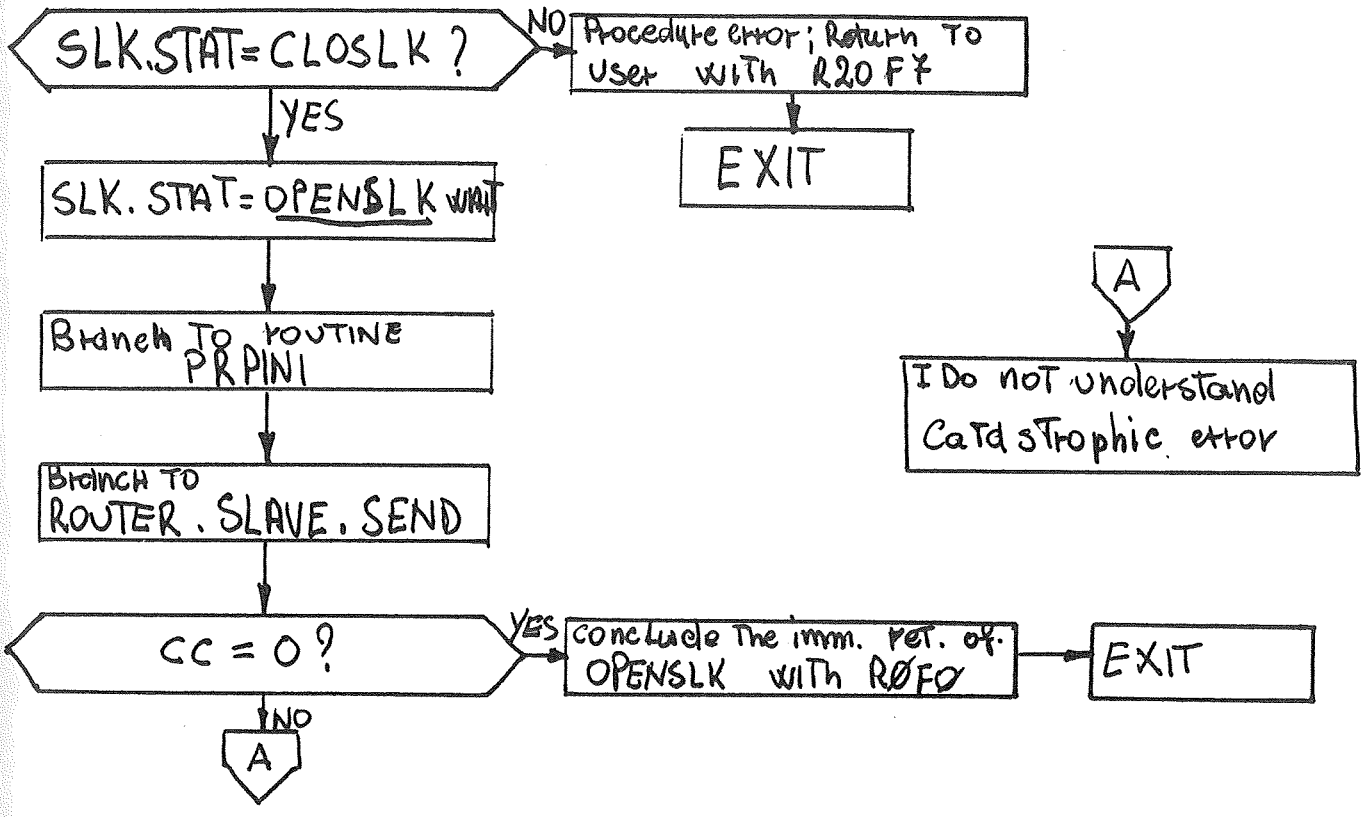
MAIL . SLAVE



WINDOW . MASTER


APPLICATION
SLAM
CC
operation end
CALL AND RETURN INTERFACE (CRI)
MEASUREMENT SYSTEM (MS)
OPERATOR INTERFACE
SATELLITE SERVICES MANAGER (SSM)
SATELLITE LINK HANDLER (SLH)
ACK/NACK HANDLER (ANH)
TIMEOUT HANDLER (TH)
CC
ROUTER
ERROR
MAM
MCHR
MCHT
RECEIVER (SLAVE)
To M6800 (CIM)
From M6800

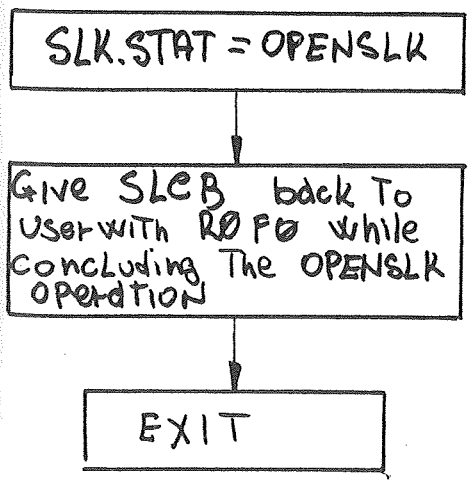
SSM. SLAVE (OPENSLK)
ENTRY SLOPN



RPINI (prepare message for inizialization) has The following functions:

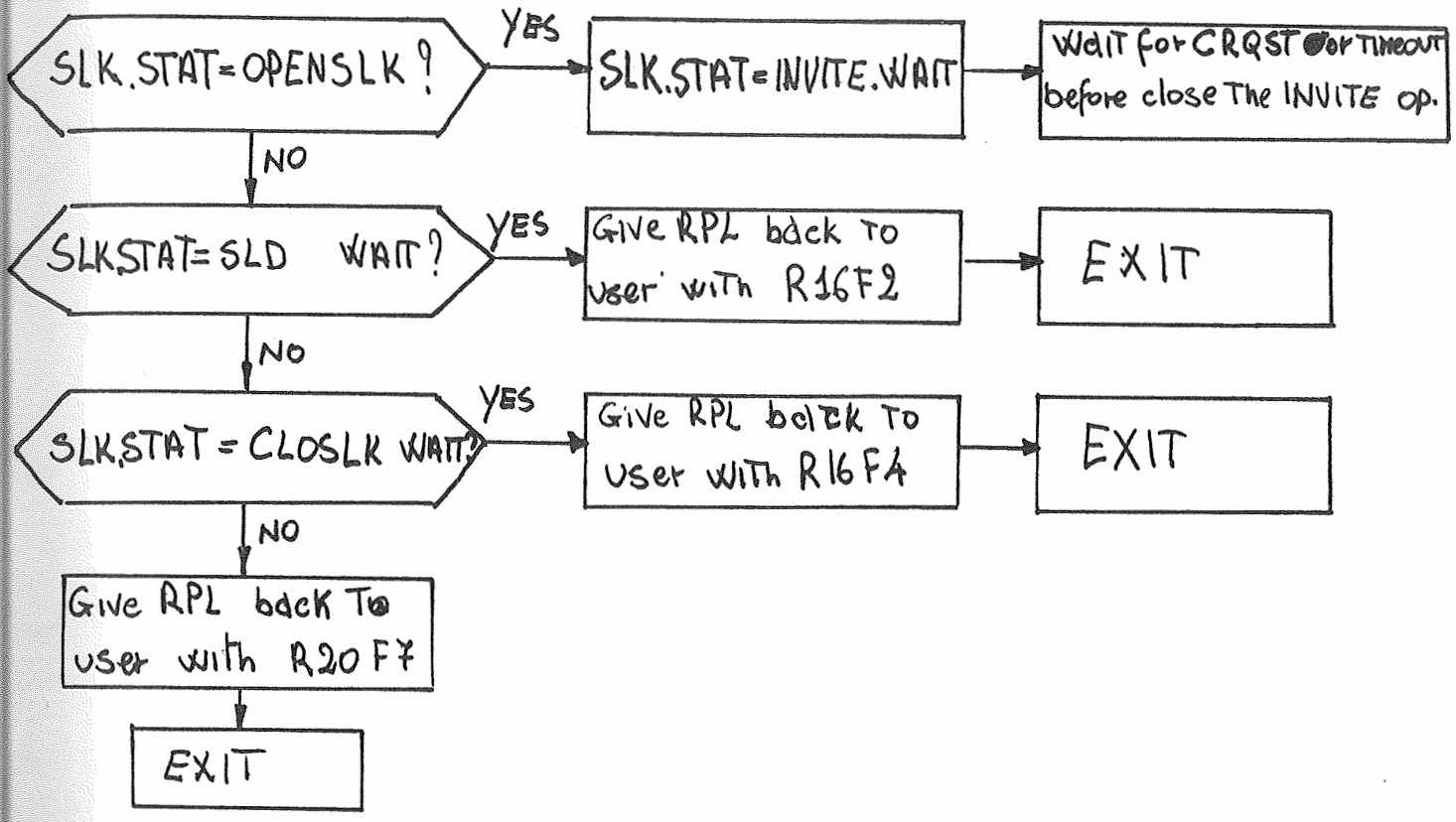
- GeT an IOBLOCK & Buffer
- Prepare The appropriate message for The Inizialization in The Buffer

ENTRY SSHOPN
(From ROUTER. SLAVE. RECEIVE)

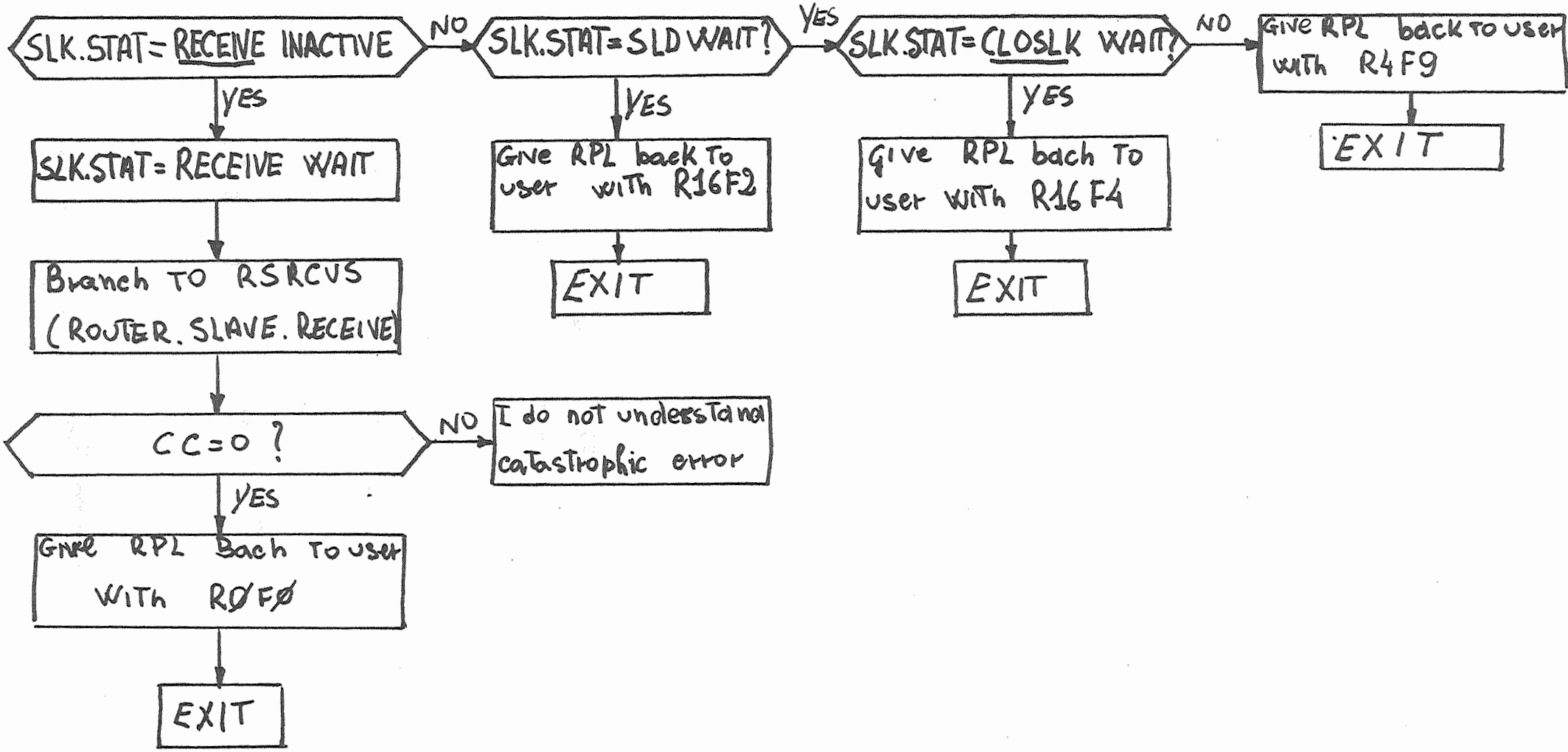


SSM.SLAVE (INVITE)

ENTRY SLINV

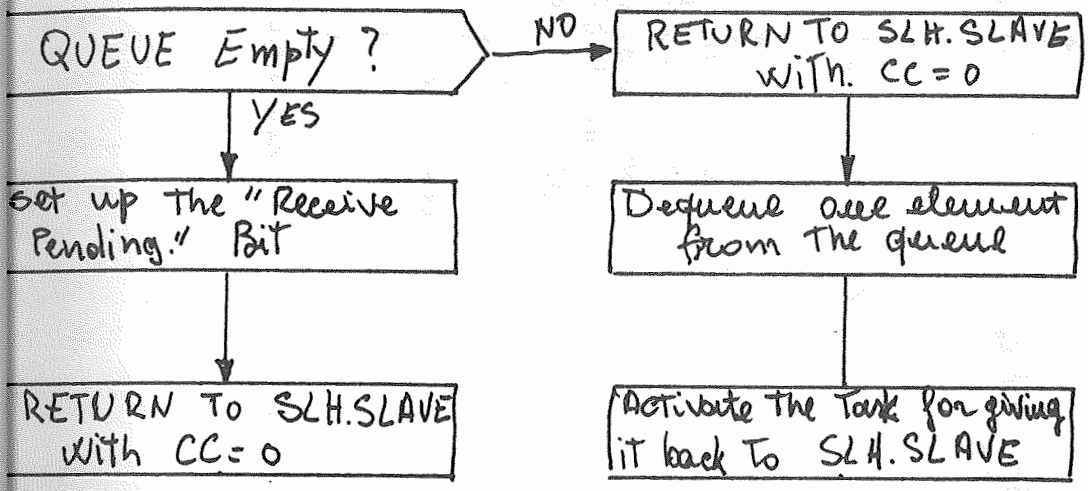


SLK.SLAVE (RECEIVE)
ENTRY SLRCV



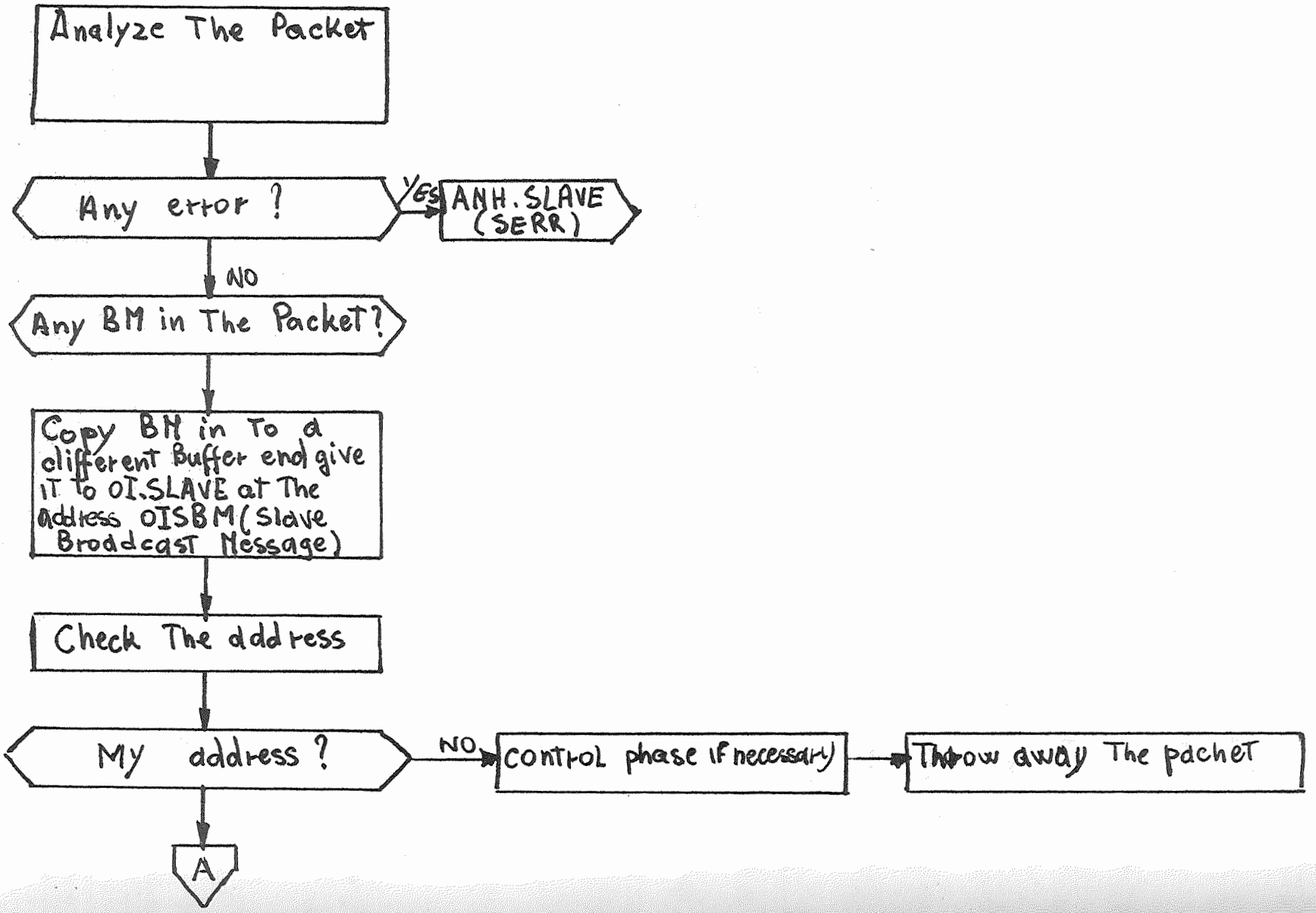
ROUTER.SLAVE.RECEIVE

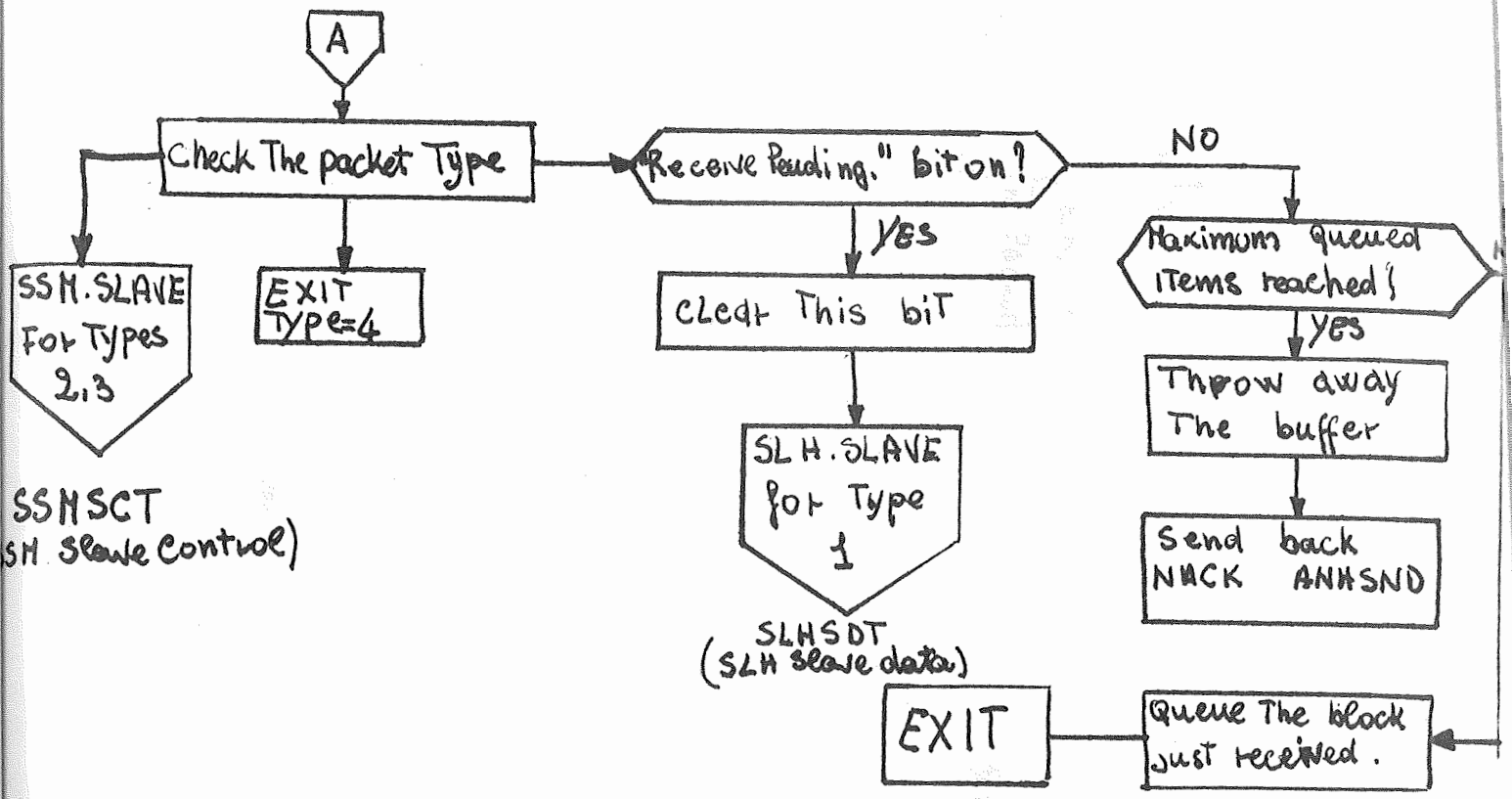
Entry RSRCVS (Entered from SLH.SLAVE



ROUTER.SLAVE.RECEIVE (Entered from MCHR.SLAVE)

ENTRY RSRCVM





Notes:

1- The receive buffer end count will be always given by us. IT will not contain The control information stored in the head of the received block.

2- The receive macro will contain a return code Telling The user To free a previous given buffer

3- IT is assumed That packets having Type 4 in The STH (satellite Trasmission header) They will not carry The PSN (Packet Sequence Number)

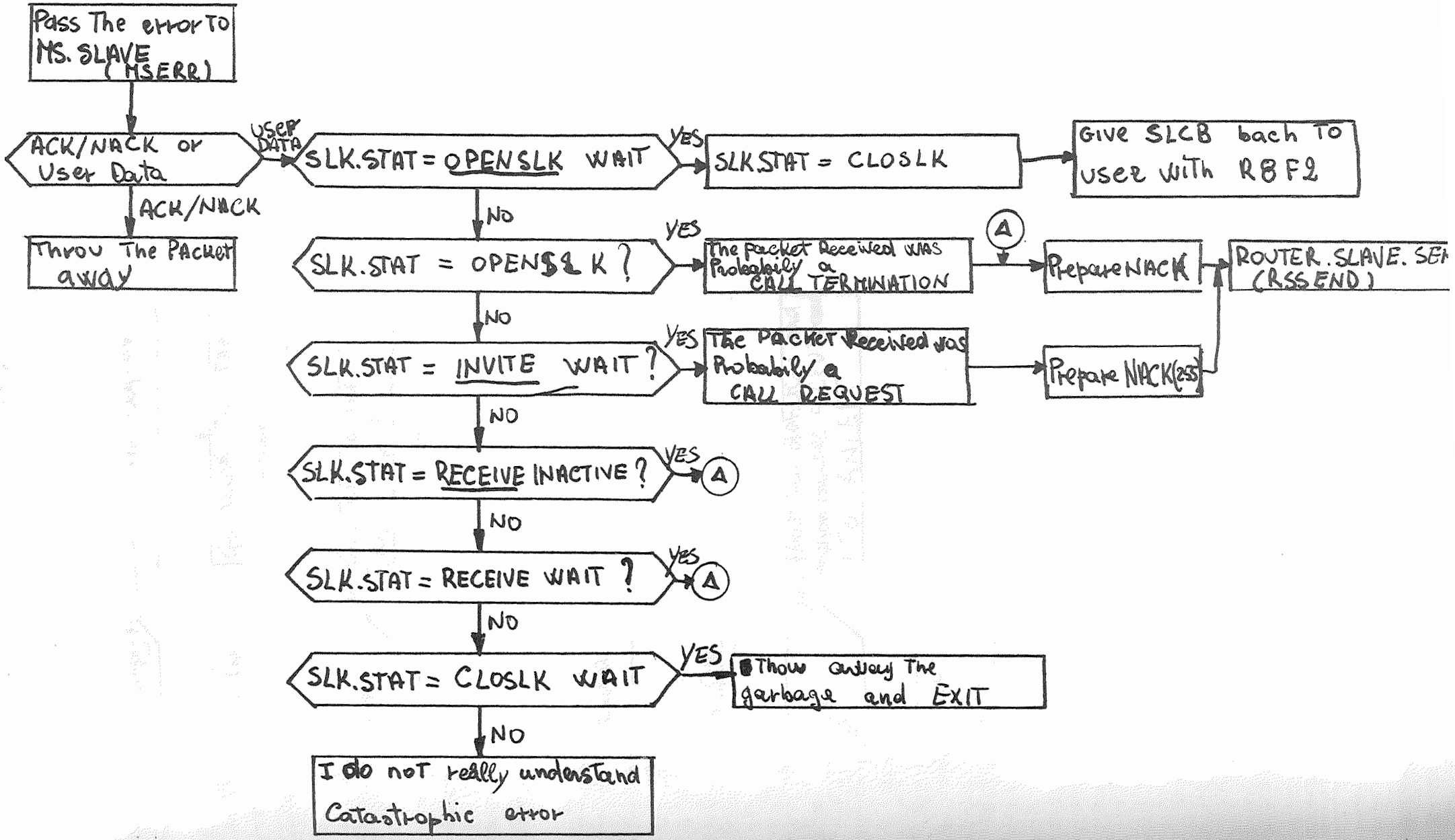
4- In The above flowchart Reference Bursts have not been Taken in To account. IS a matter of fact They have To be defined!

5- anume That a packet is received in error (SERR). IT is possible to understand whether or not This packet refers to a user data packet o To an ACK/NACK packet?

- If a packet is received in error it is assumed that it refers To The earth station under consideration

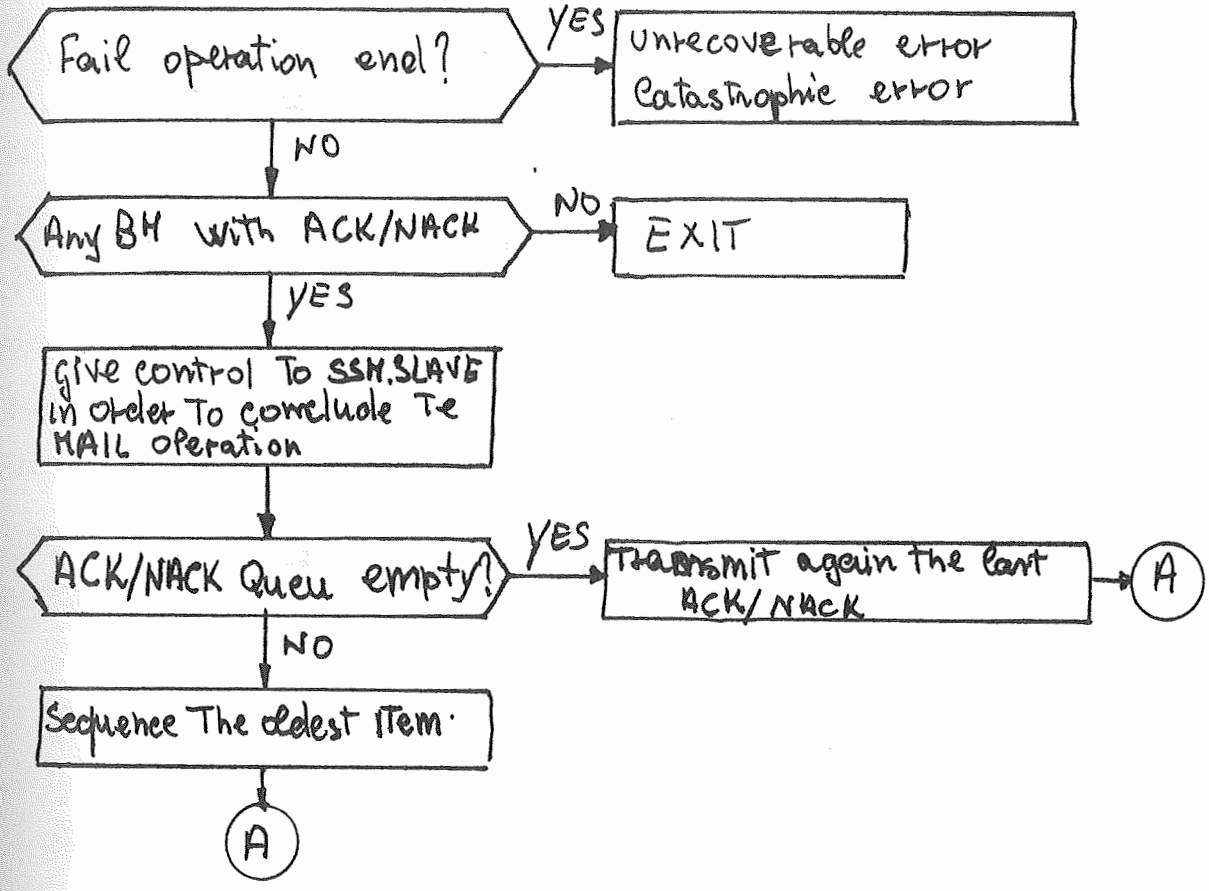
ANH.SLAVE

ENTRY SERR (from ROUTER.SLAVE.RECEIVE)



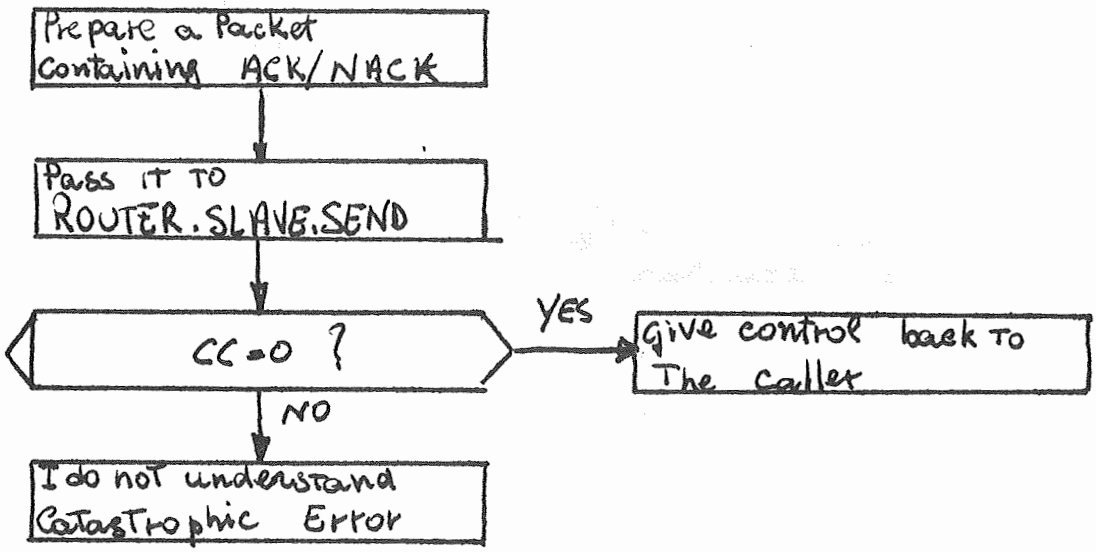
branch here when ACK or NACK left The antenna

Entry ANLEFT (from MCHT.SLAVE)

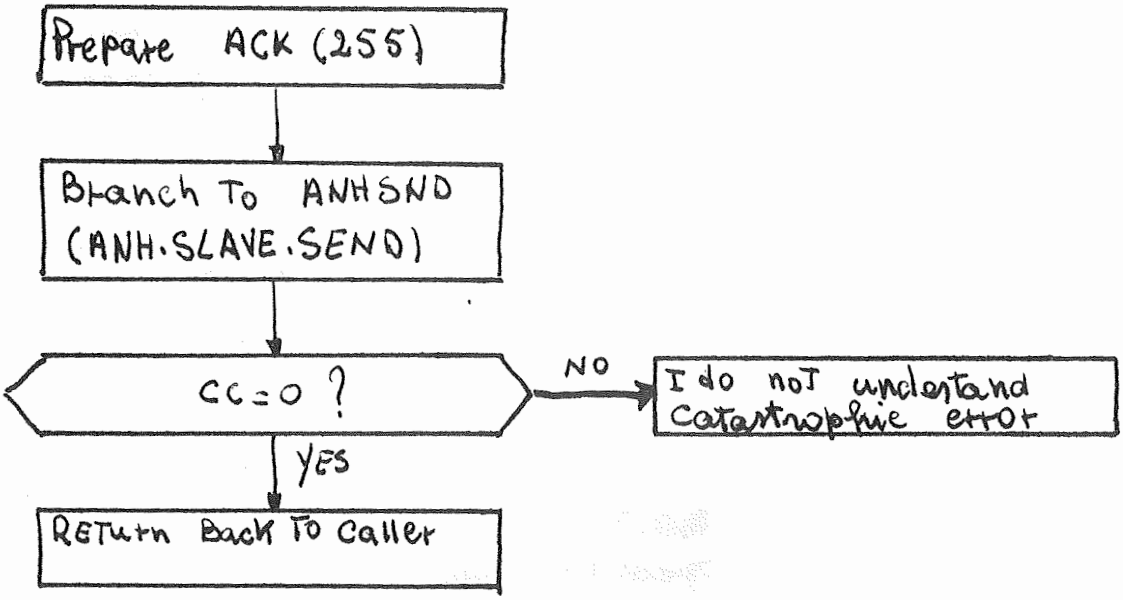


ANH.SLAVE.SEND

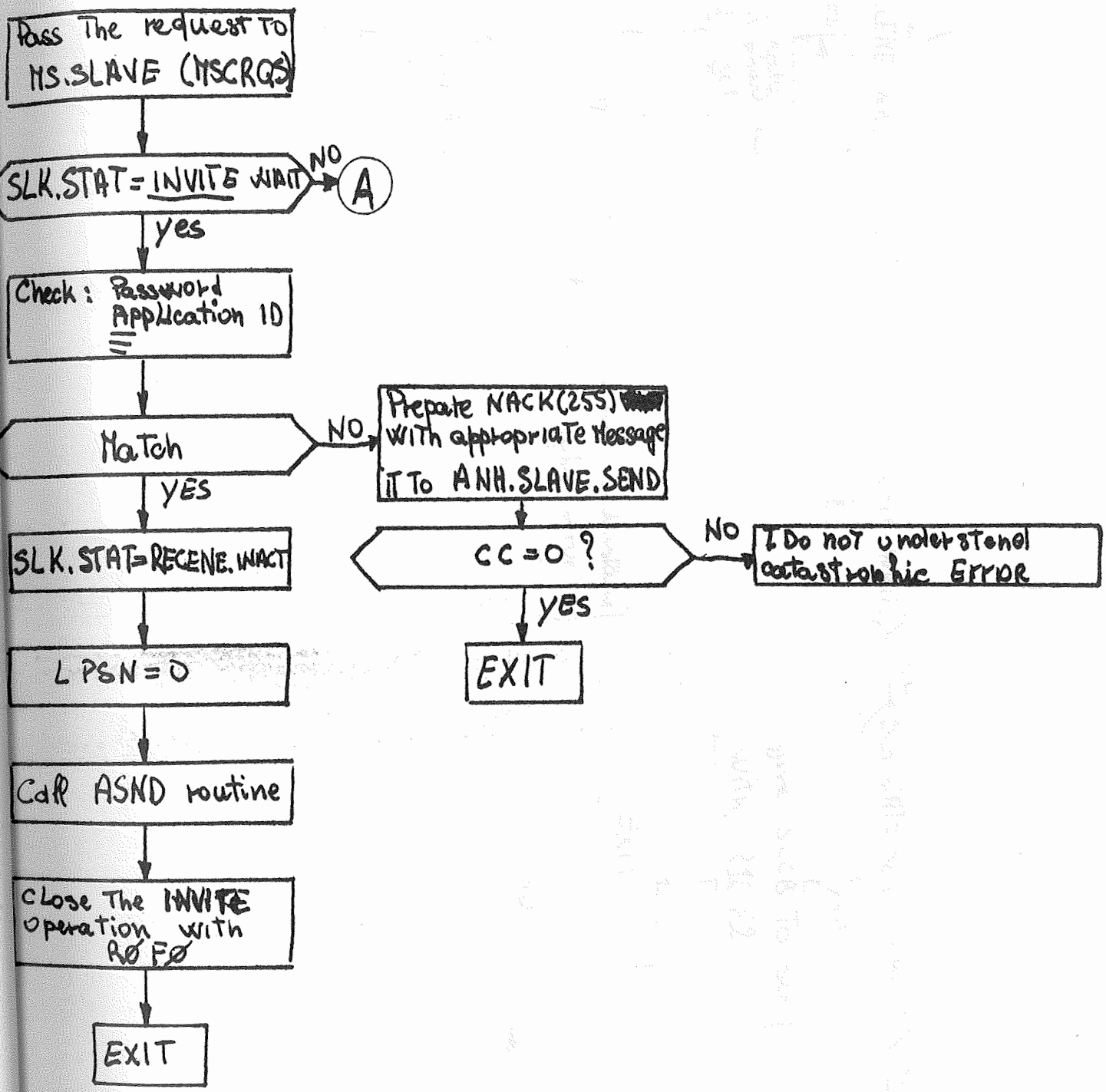
Entry : ANHSND



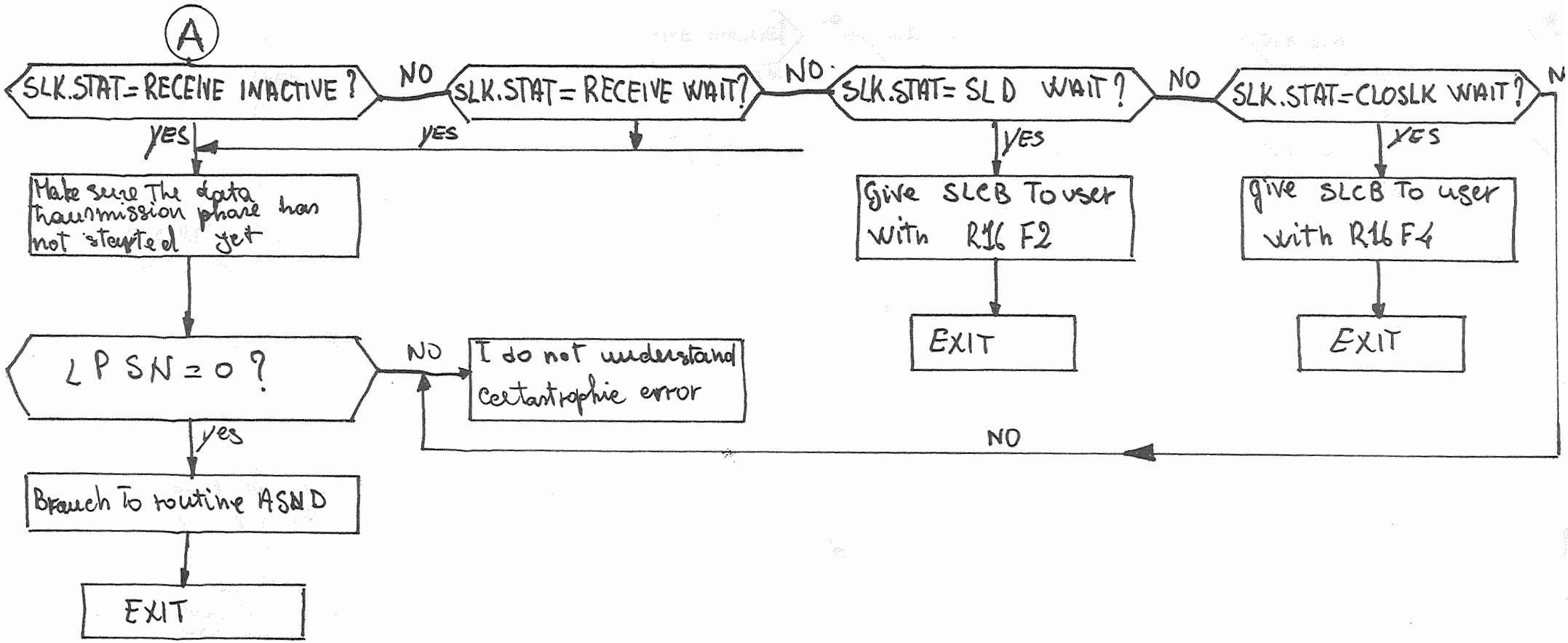
ASND (Routine)



SSM.SLAVE.RECEIVE (from ROUTER.SLAVE.RECEIVE)
ENTRY SSMSRQ Tipy = 2 CALL REQUEST

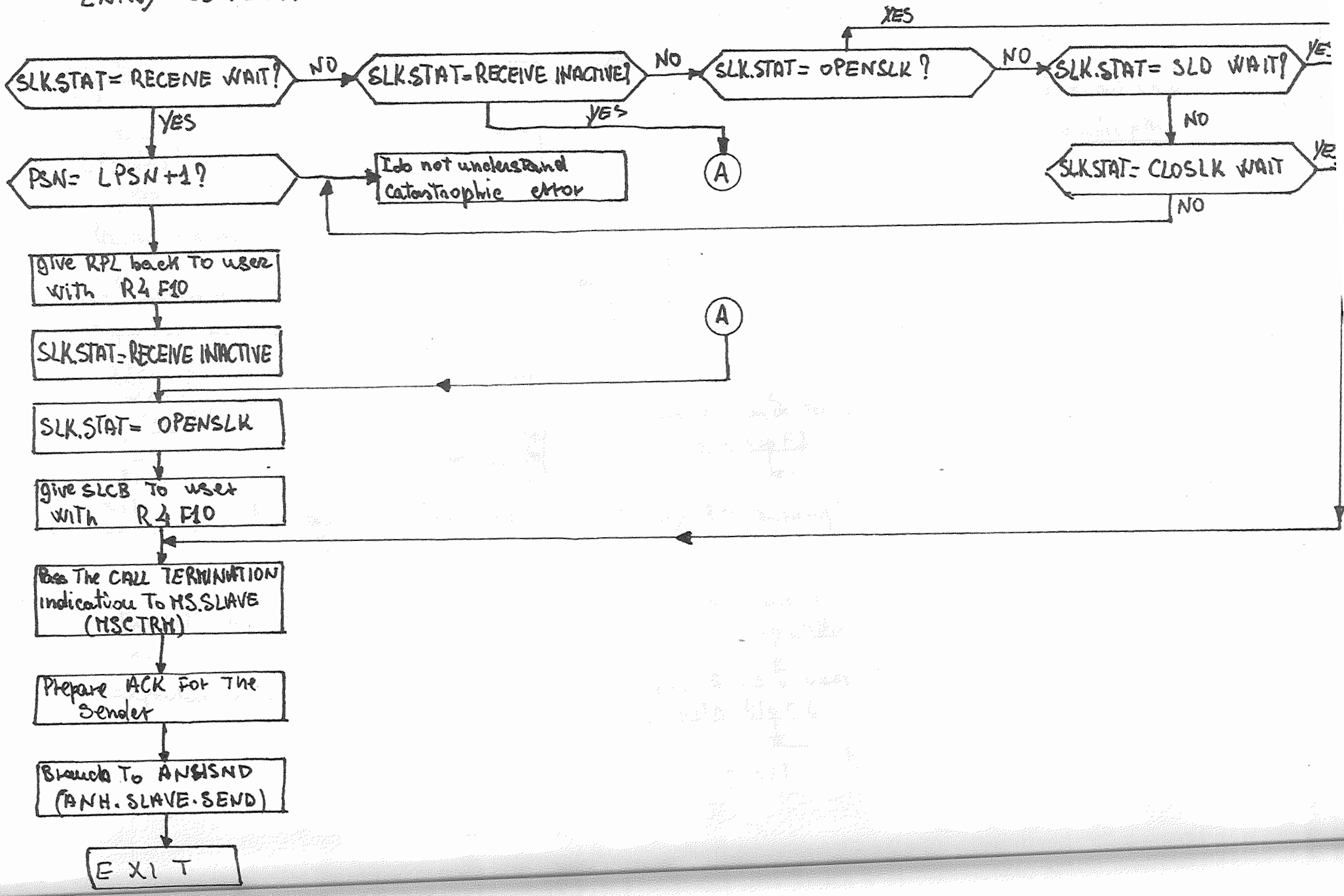


L PSN = Last Packet Sequence Number
(Espected PSN)


A
SLK.STAT=RECEIVE INACTIVE ?
NO
SLK.STAT=RECEIVE WAIT?
NO
SLK.STAT=SLD WAIT?
NO
SLK.STAT=CLOSLK WAIT?
N
YES
YES
YES
YES
Make sure The data transmission phase has not stepted yet
Give SLCB To user with R16 F2
give SLCB To user with R16 F4
EXIT
EXIT
LPSN=0 ?
NO
I do not understand catastrophic error
yes
NO
Branch To routine ASND
EXIT

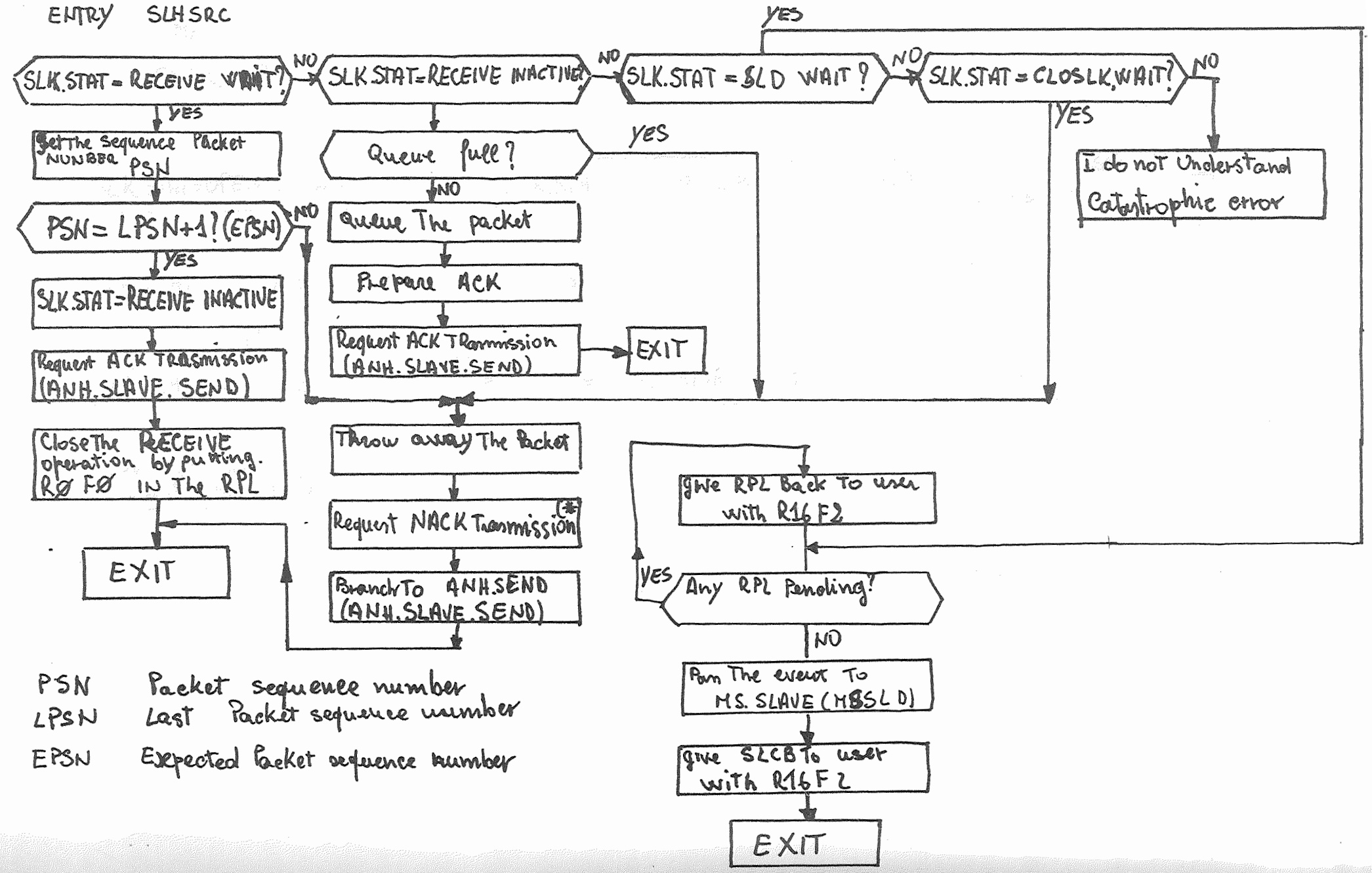
SSM.SLAVE.RECEIVE (Type 3 CALL TERMINATION)

ENTRY · SSMSTR



SLH.SLAVE.RECEIVE Type = 1 Data record (From ROUTER.SLAVE-RECEIVE)

ENTRY SLHSRC

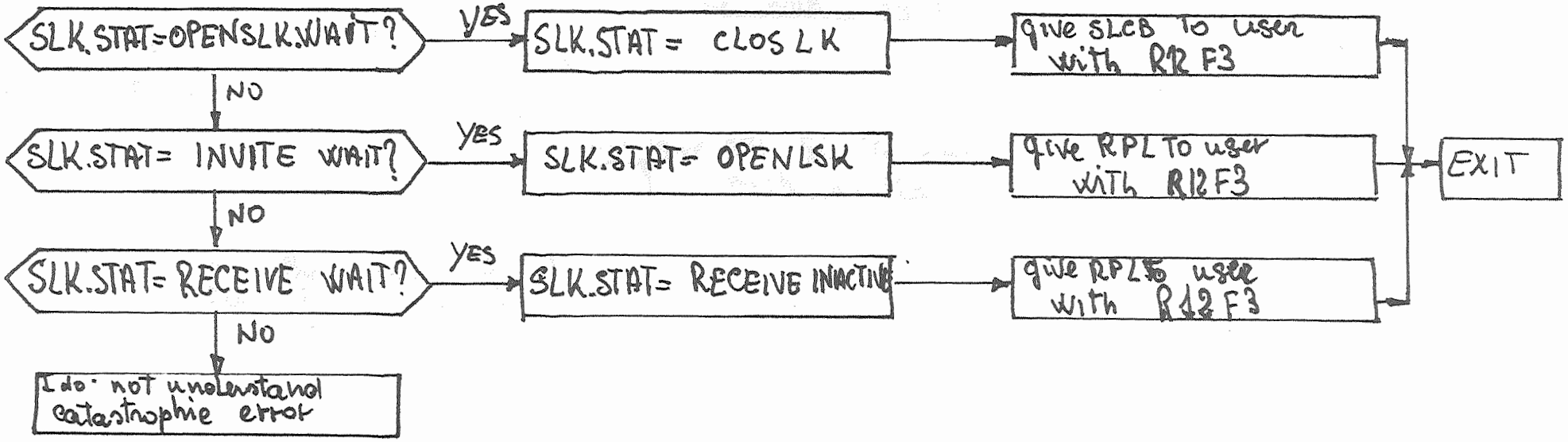


PSN Packet sequence number
LPSN Last Packet sequence number
EPSN Expected Packet sequence number

* The NACK under consideration has The sequence number of The last correct packet received (N). A NACK

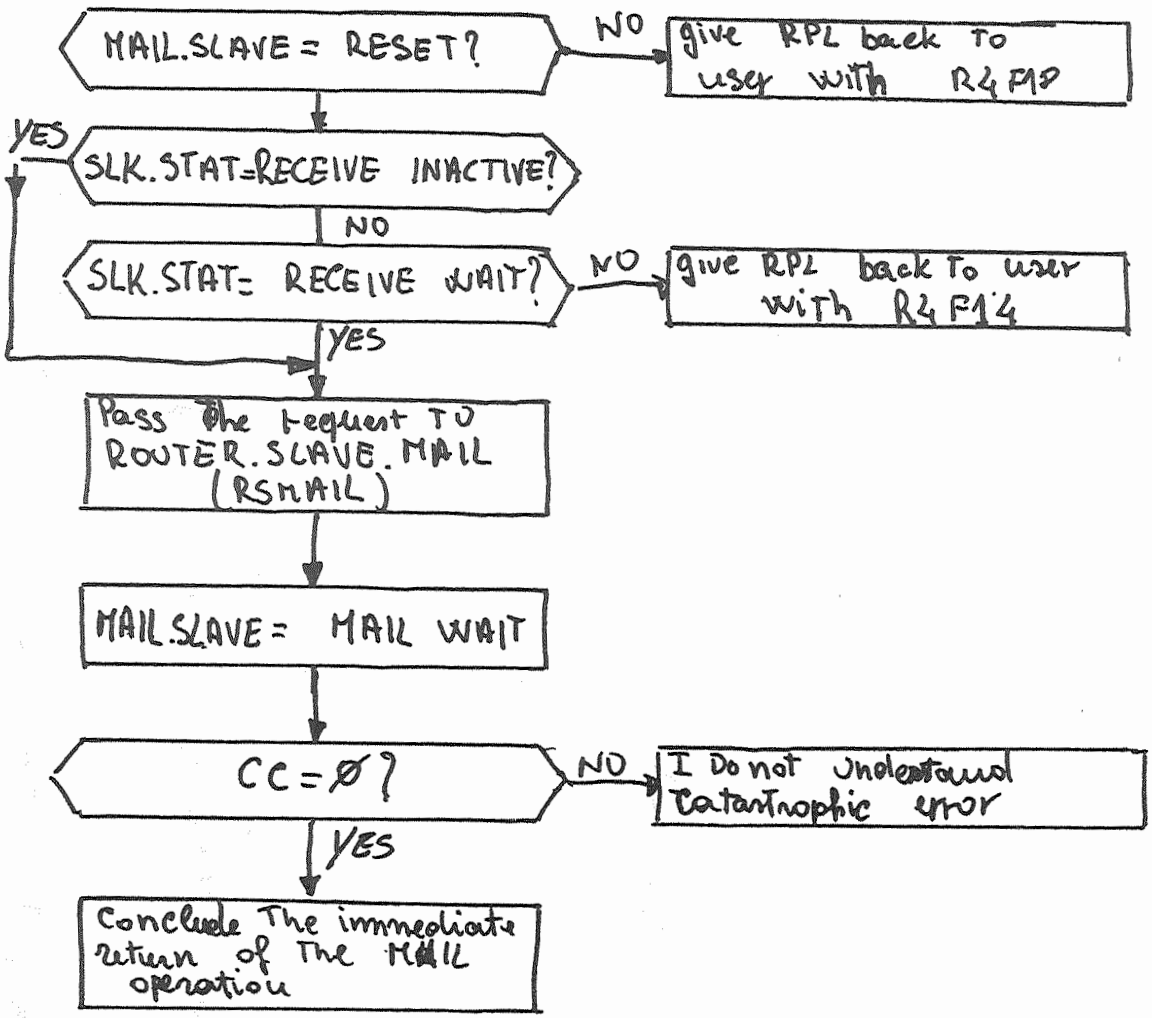
TIMEOUT.SLAVE

ENTRY TMOUTS

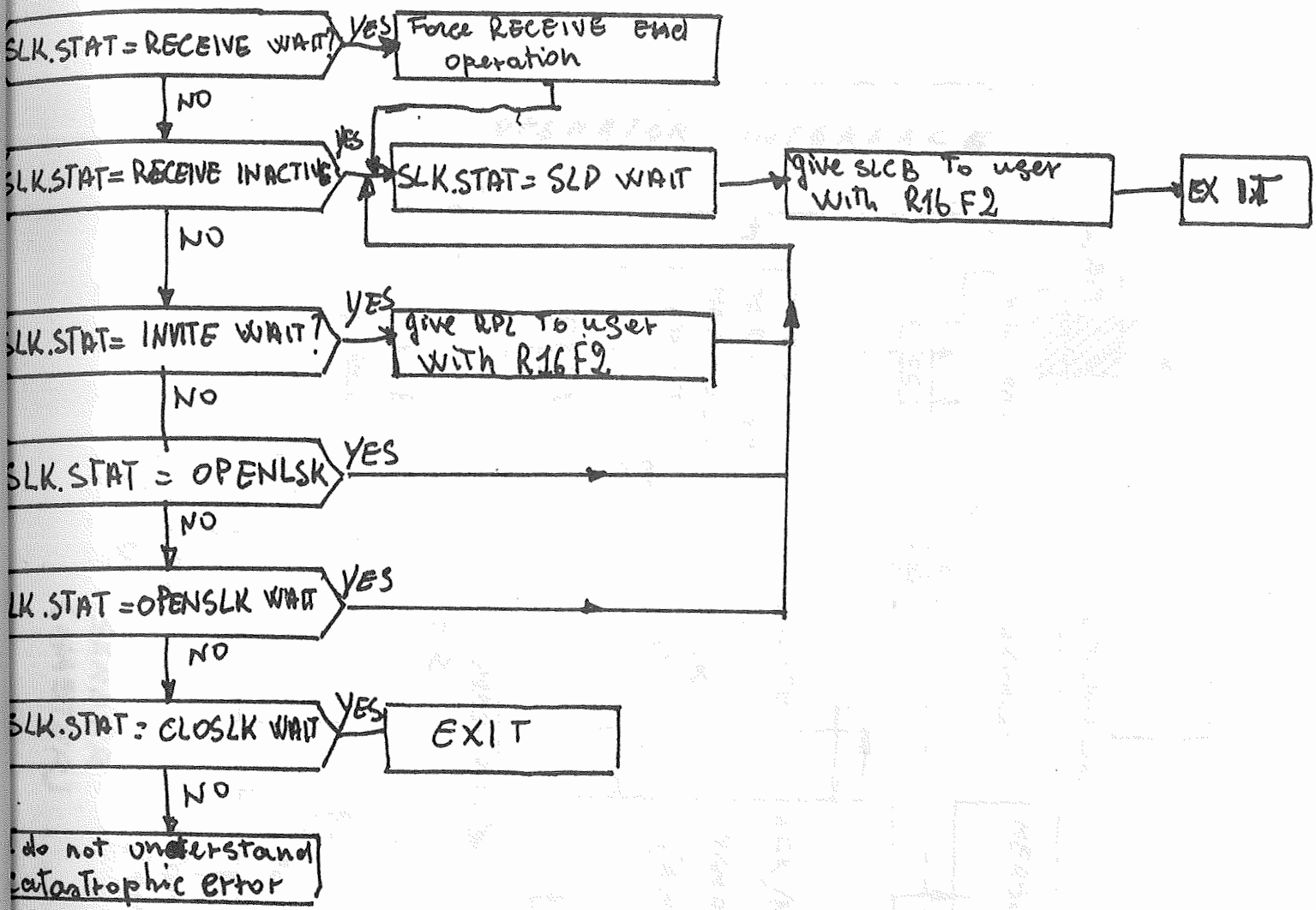


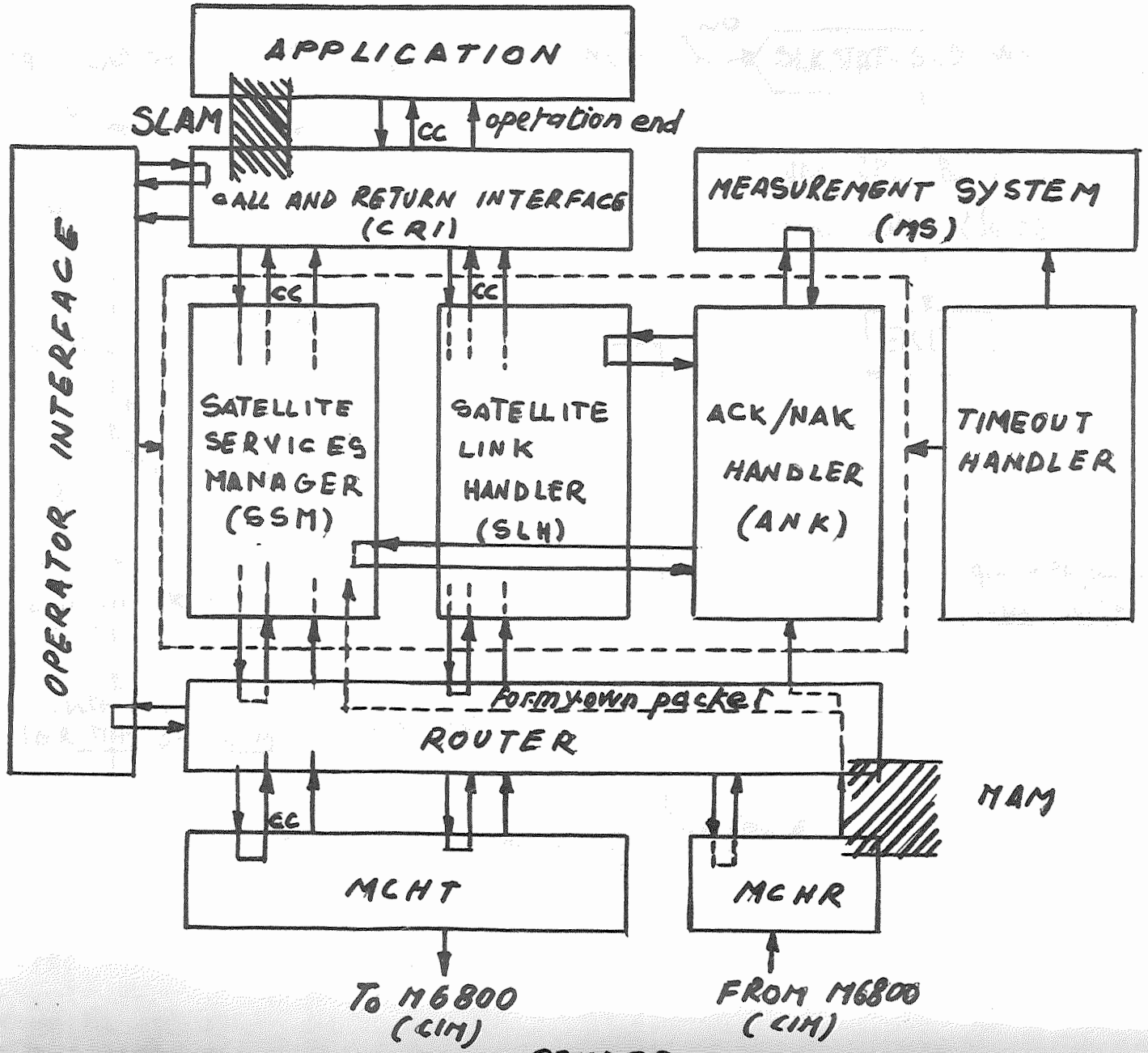
SSM.SLAVE.MAIL (MAIL OPERATION)

ENTRY SLMLS (Mail slave)



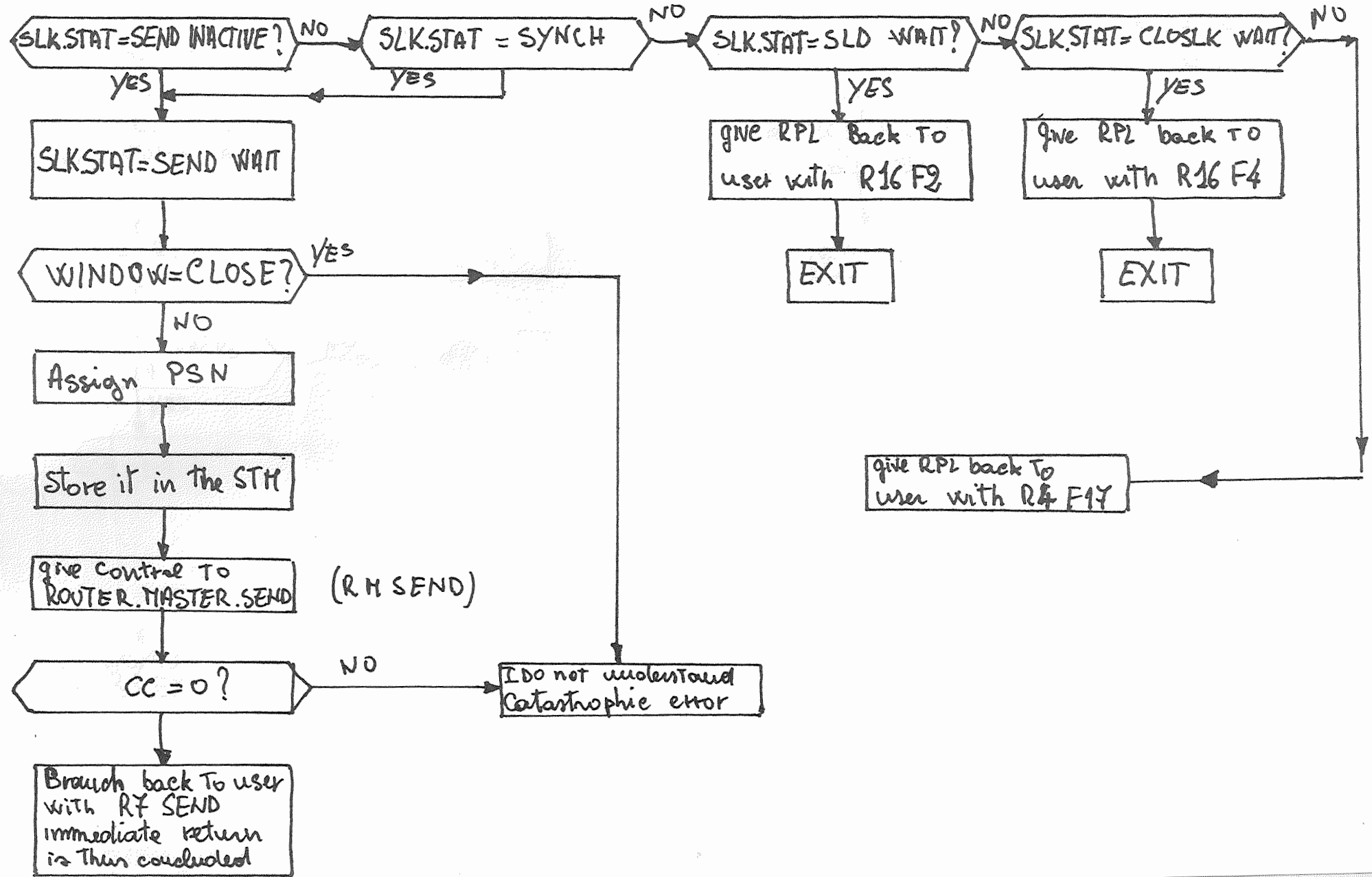
SSH.SLAVE
Entry SLDWNS (Satellite Link DOWN SLAVE)




APPLICATION
SLAM
CC
operation end
CALL AND RETURN INTERFACE (CRI)
MEASUREMENT SYSTEM (MS)
OPERATOR INTERFACE
CC
CC
SATELLITE SERVICES MANAGER (SSM)
SATELLITE LINK HANDLER (SLH)
ACK/NAK HANDLER (ANK)
TIMEOUT HANDLER
For my own packet
ROUTER
MAM
CC
MCHT
MCHR
To M6800 (CIM)
FROM M6800 (CIM)
SENDER

SLH.MASTER.SEND (senol)

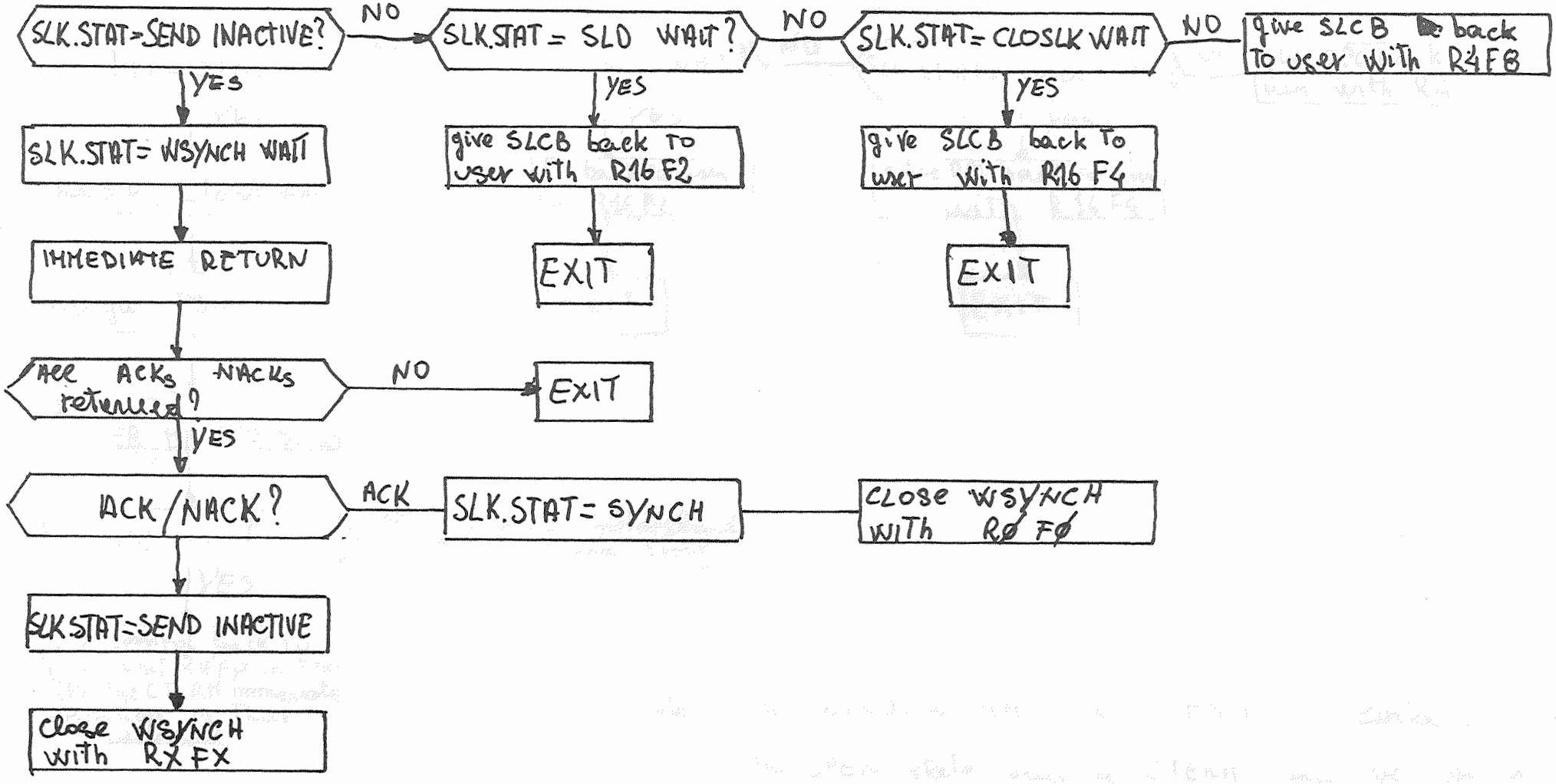
Entry SLSEND



24

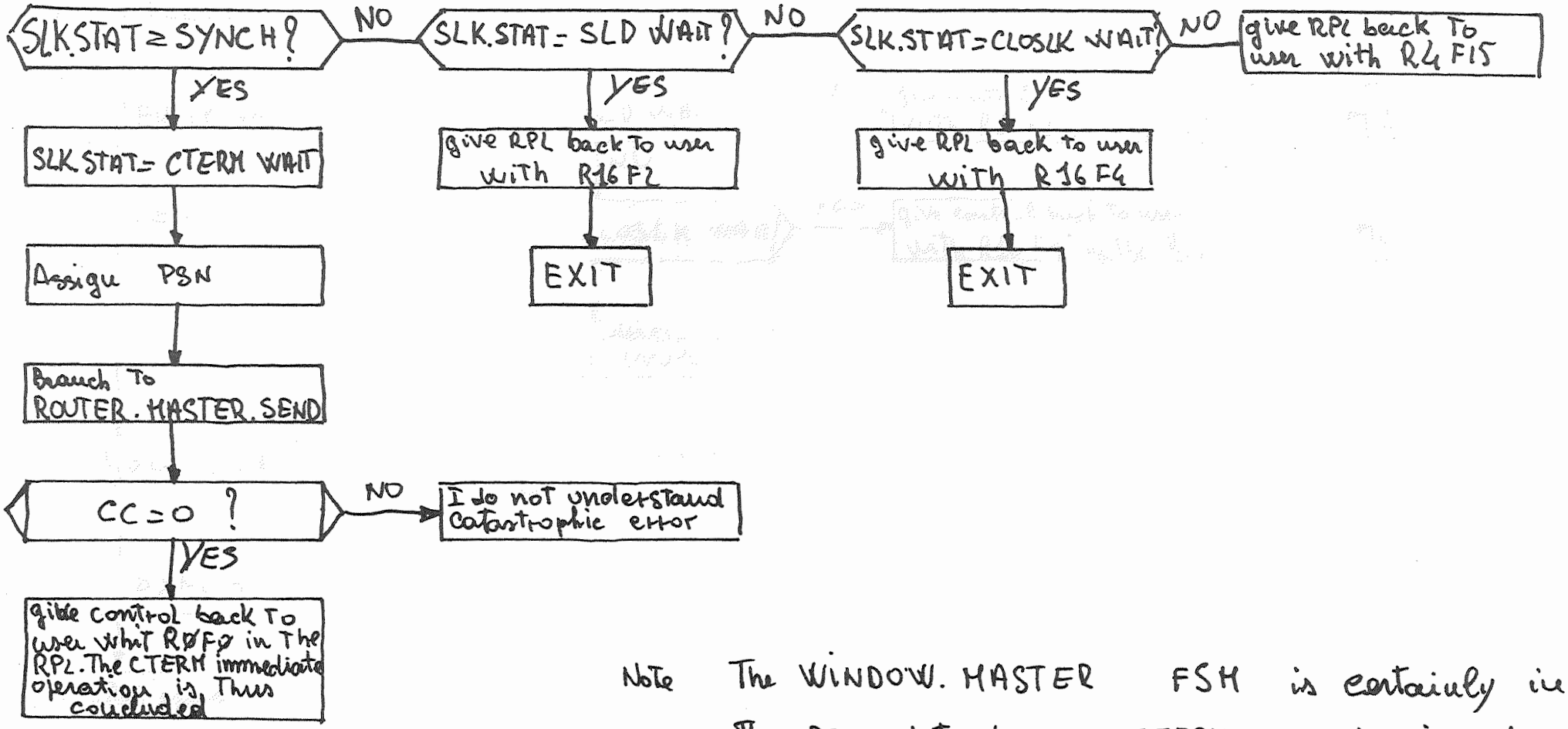
SLH.MASTER.SEND (WSYNCH)

Entry SLWSCH



SSM.MASTER.SEND (CTERM)

Entry SCLTR

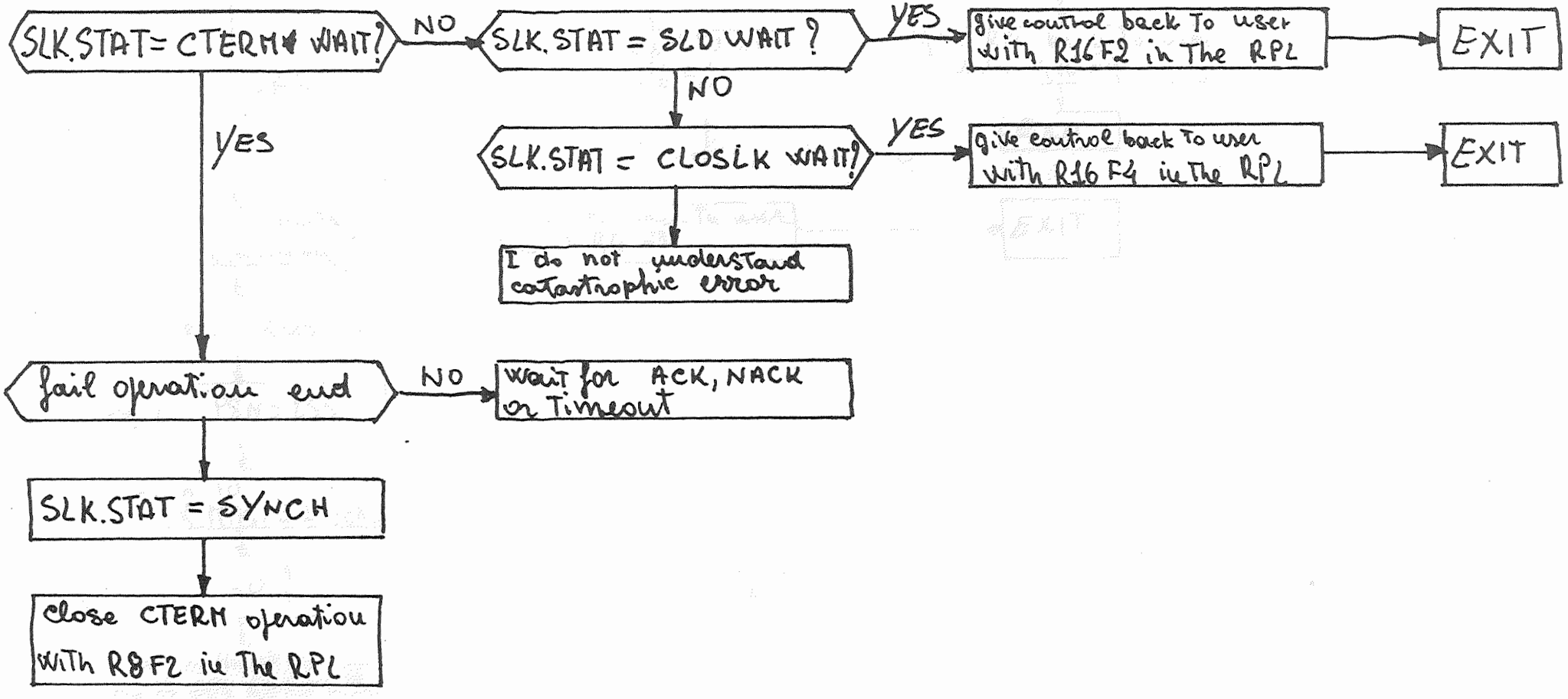


Note The WINDOW.MASTER FSM is certainly in The OPEN state, because CTERM can be issued only after sender ed receiver have sinchronized

SSM.MASTER:SEND (From ROUTER.MASTER.SEND)

Branch here when the CALL TERMINATION packet left the antenna.

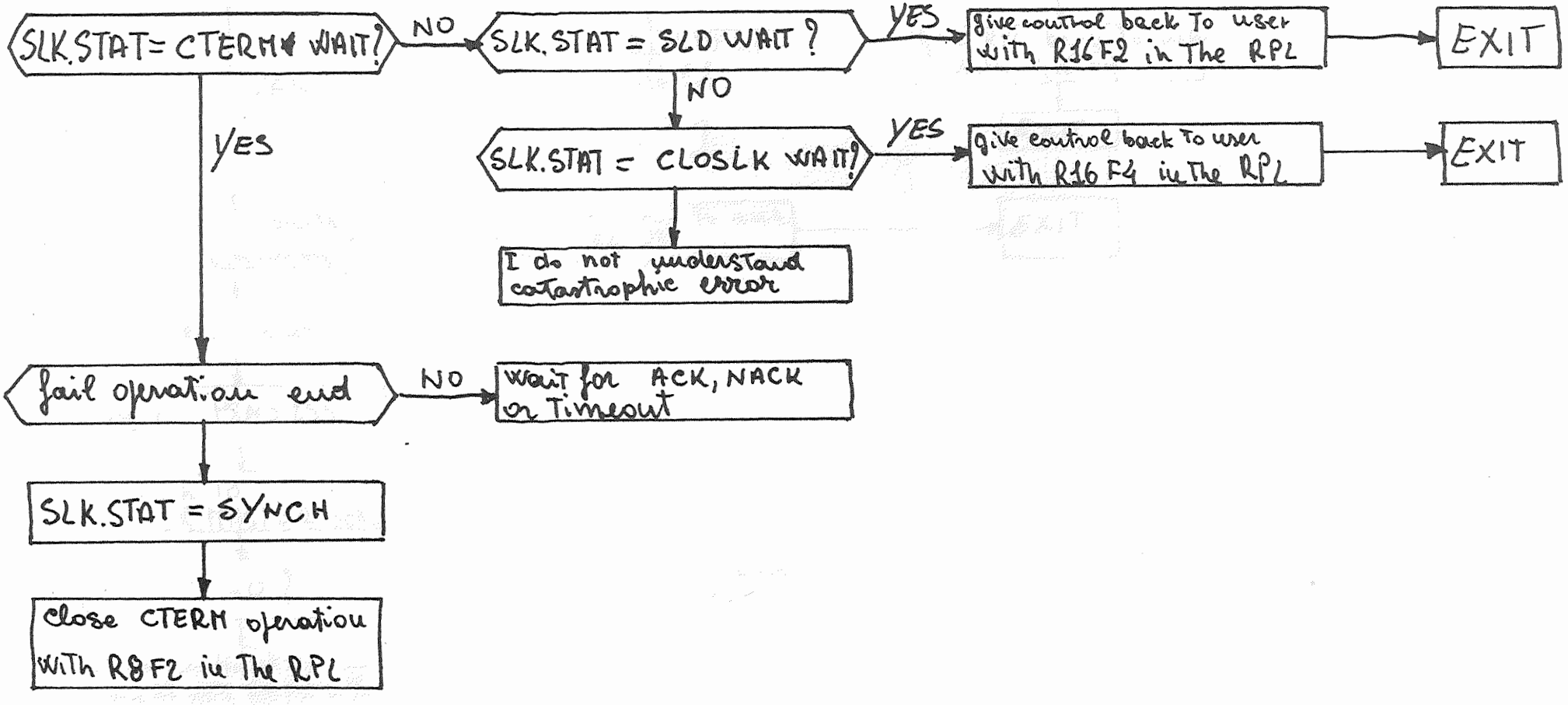
Entry CTLEFT



SSM.MASTER:SEND (From ROUTER.MASTER.SEND)

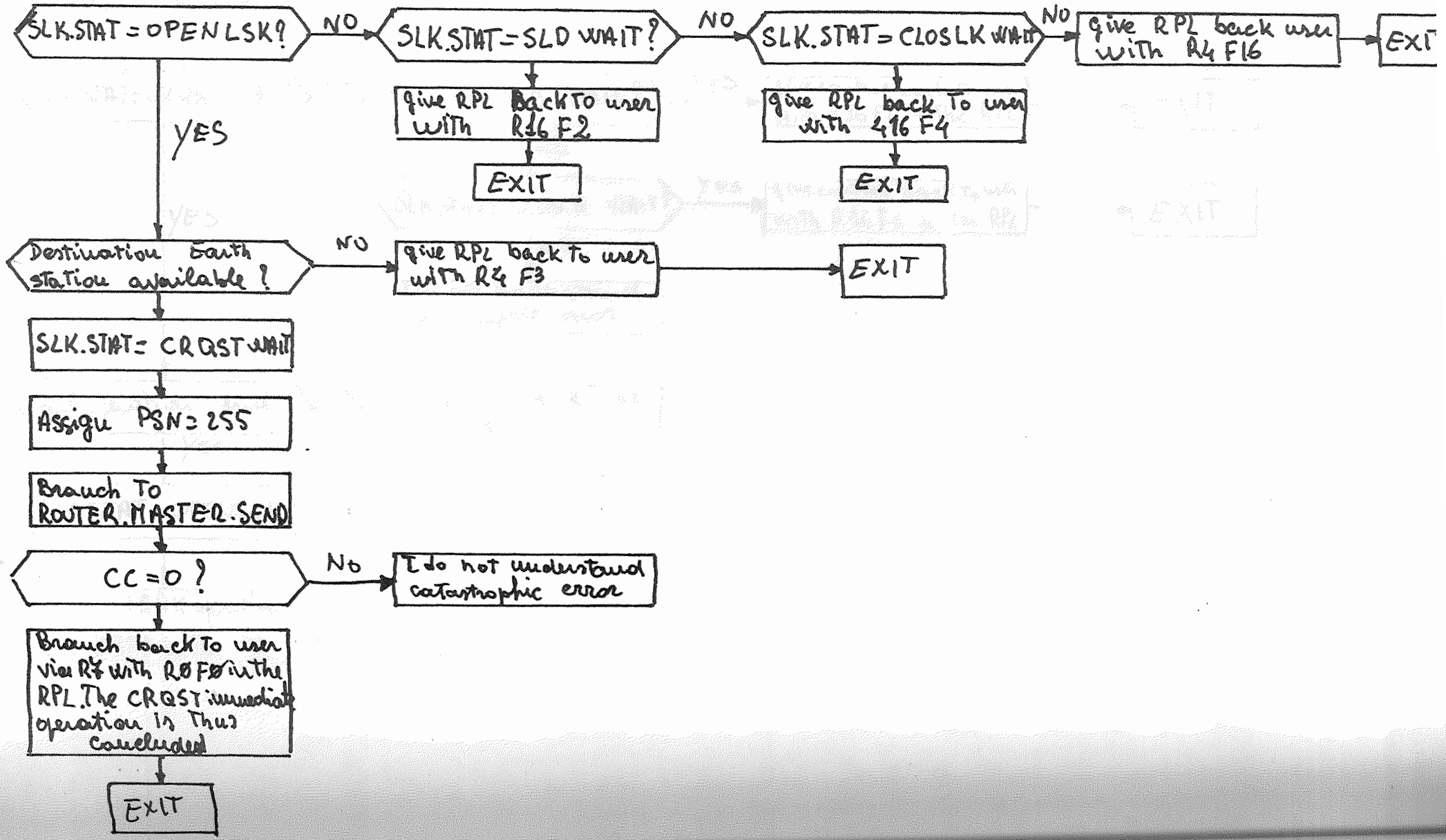
Branch here when the CALL TERMINATION packet left the antenna.

Entry CTLEFT



SSM.MASTER.SEND (CRQST)

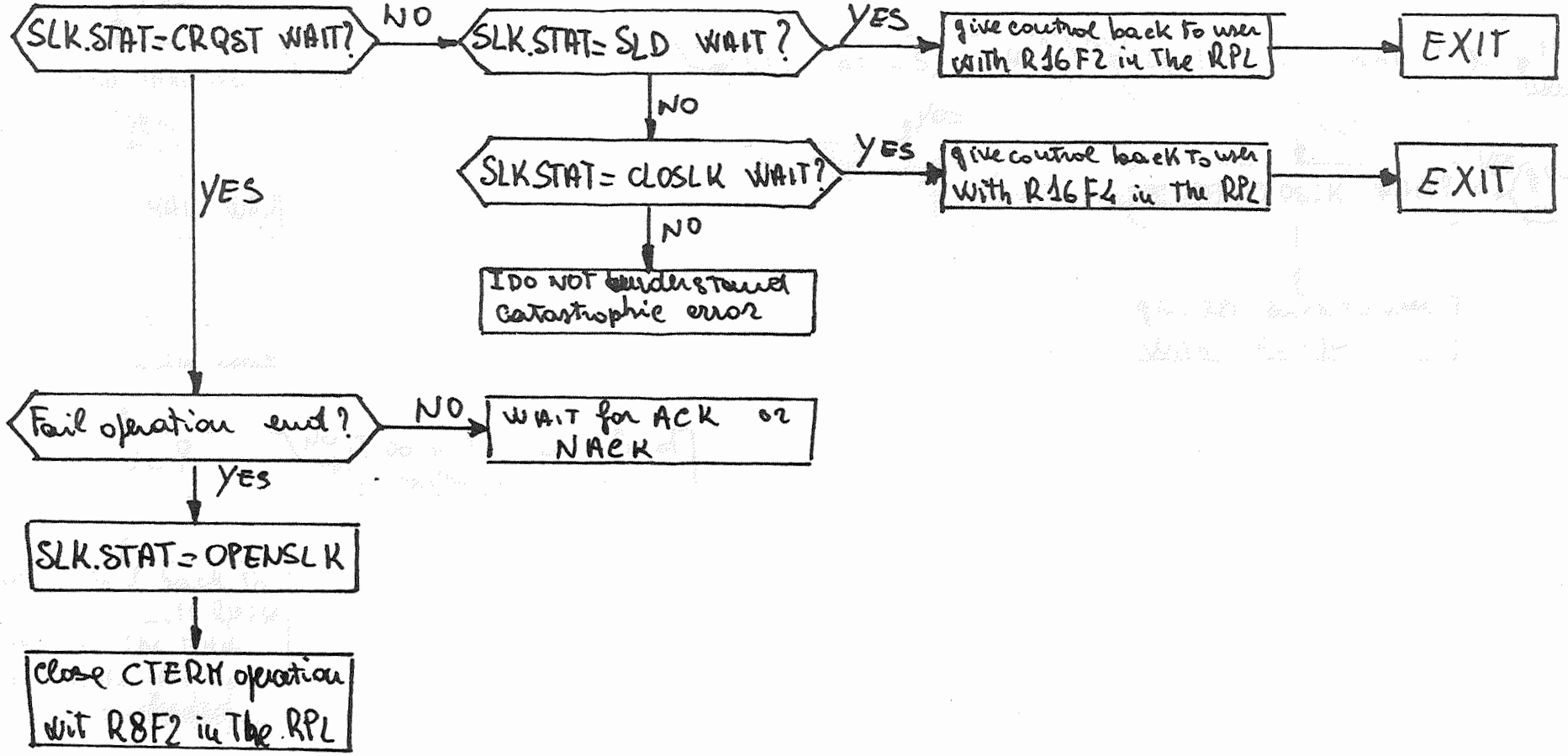
ENTRY SLCRQ



SSH.MASTER.SEND (FROM ROUTER.MASTER.SEND)

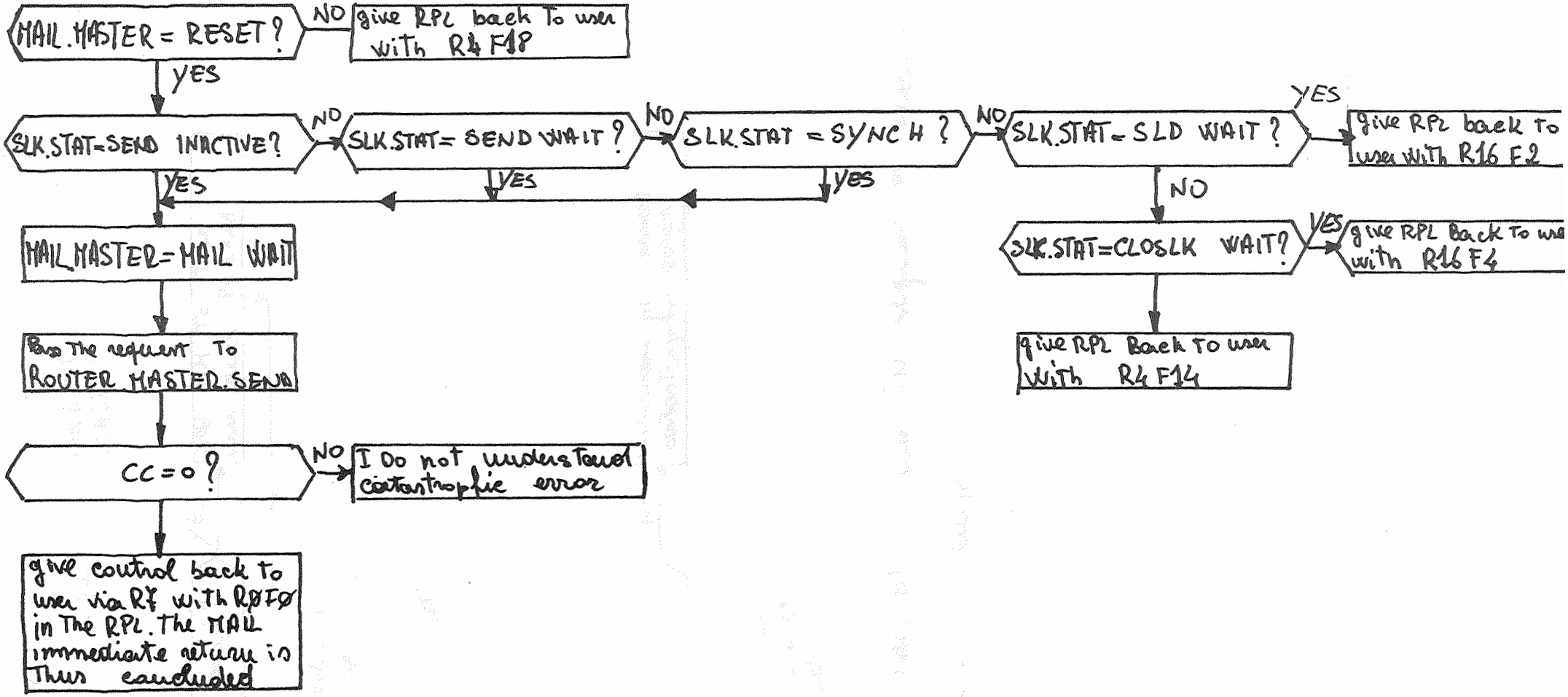
Branch here when the CALL REQUEST packet left the antenna

Entry CT.LEFT



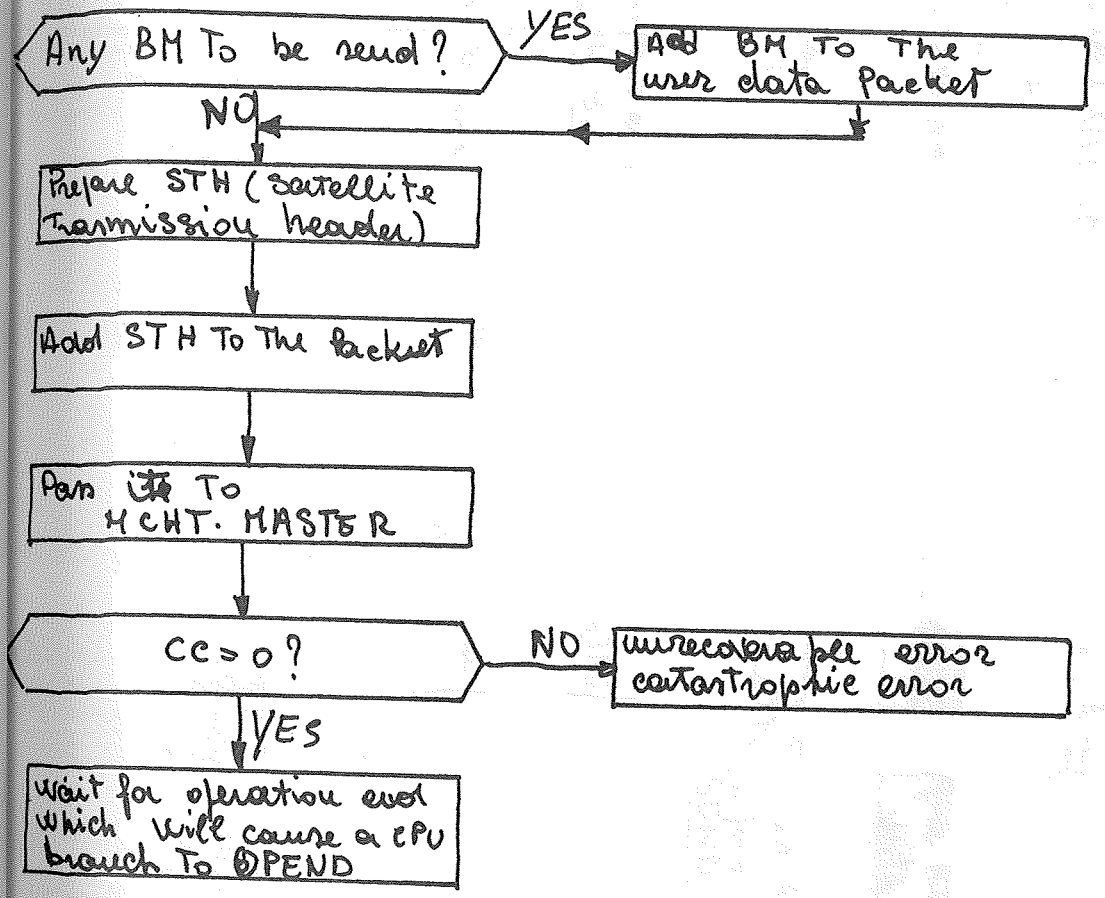
SSM.MASTER.MAIL

Entry SLMLM (MAIL MASTER)



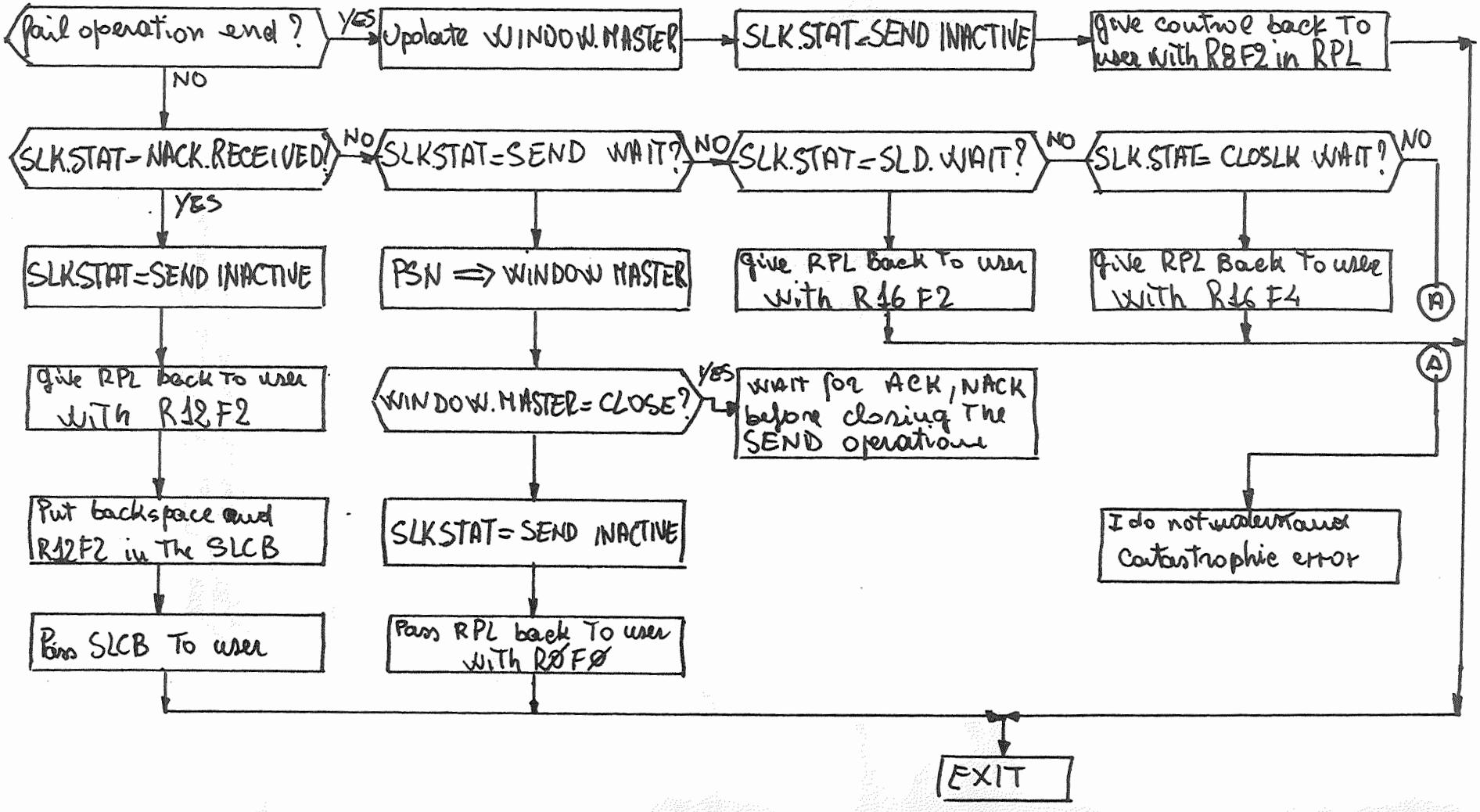
ROUTER.MASTER.SEND

Entry RMSEND (from SLH.MASTER
LSS.MASTER)



Note: MCHT.MASTER Queues The request or execute it if The queue is empty

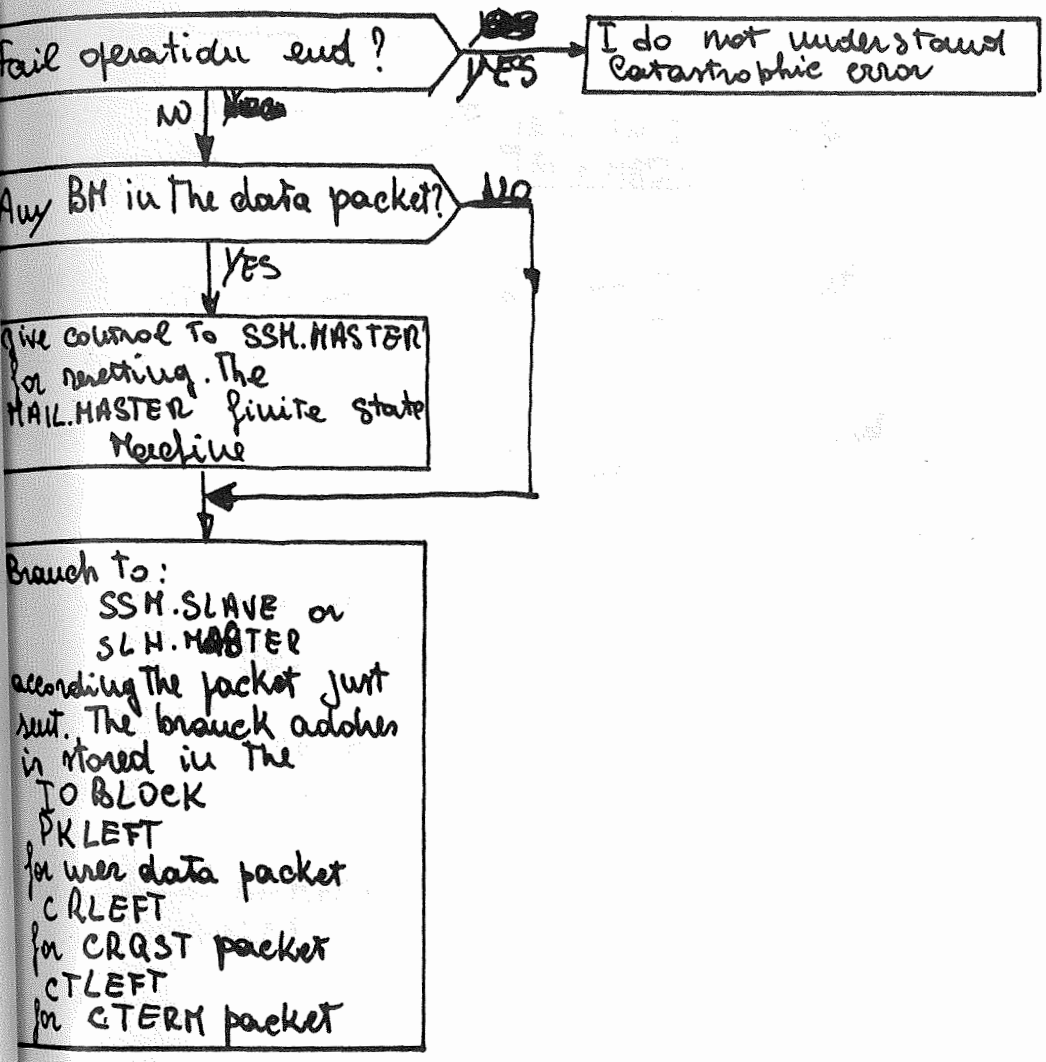
SLH.MASTER (From ROUTER.MASTER.SEND
Brouck here when the packet left the antenna
Entry PKLEFT



ROUTER.MASTER.SEND

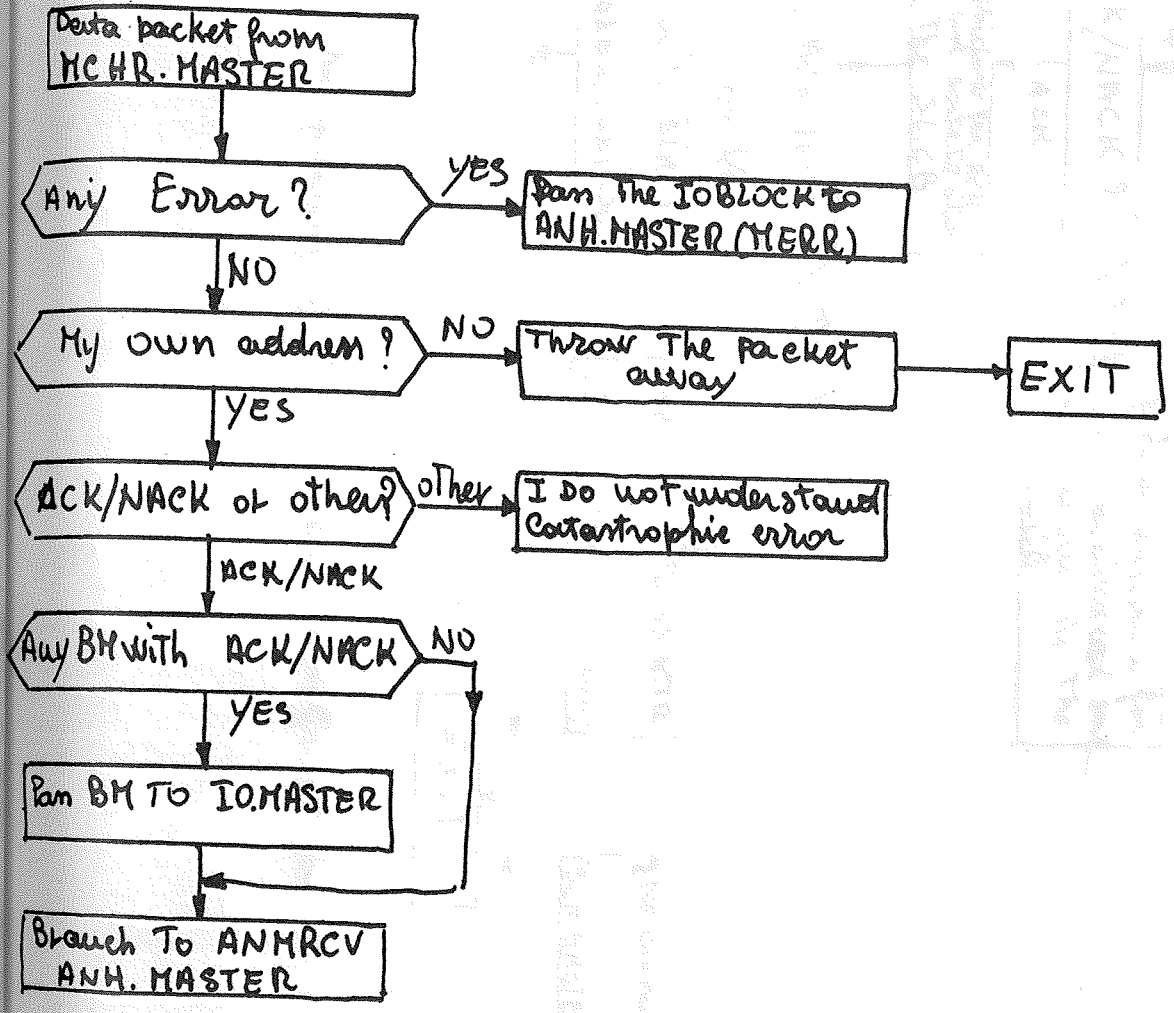
Entry OPEND (from MCHT.MASTER)

Branch here when the data packet left the Antenna



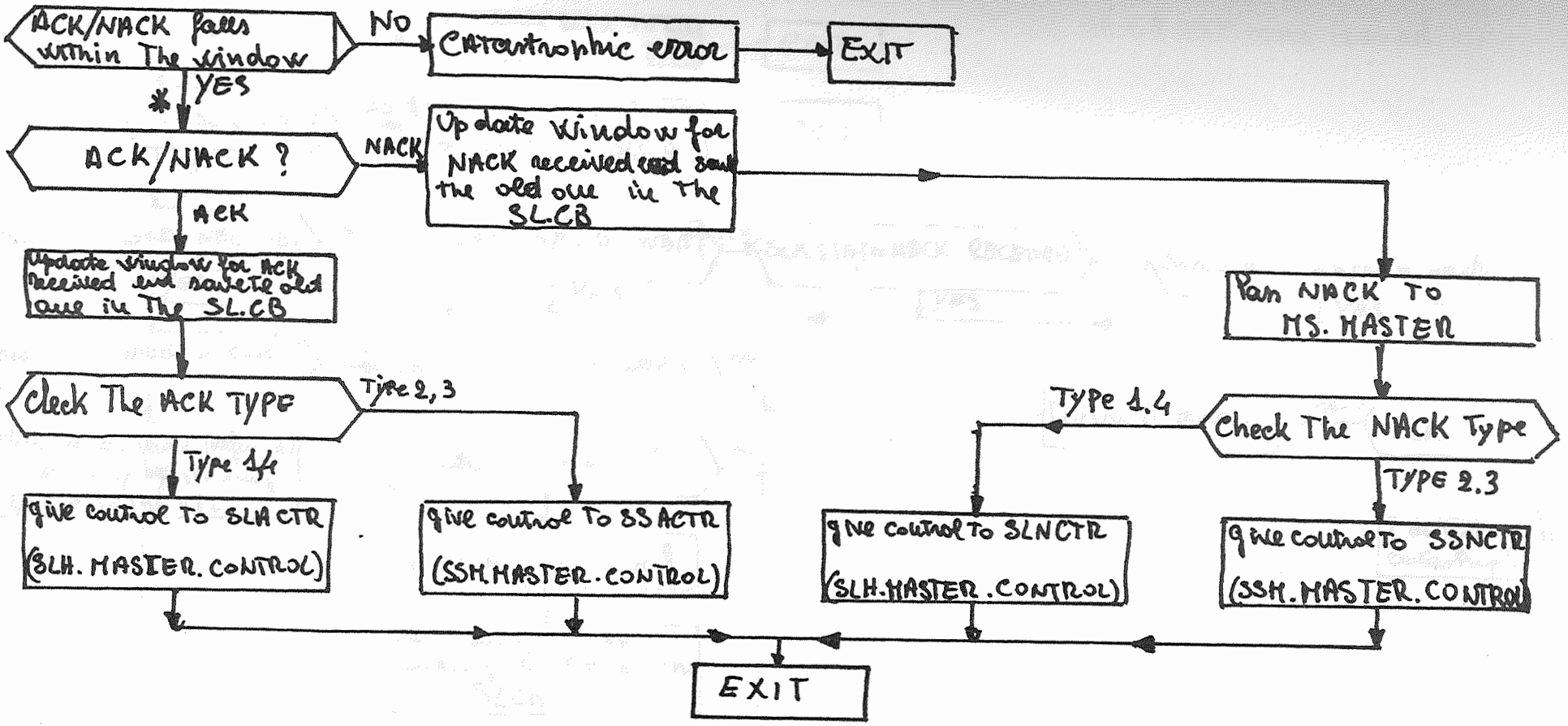
ROUTER.MASTER. RECEIVE

ENTRY RMRCV (from MCHR.MASTER)



ANH.MASTER.RECEIVE (from ROUTER.MASTER.RECEIVE)

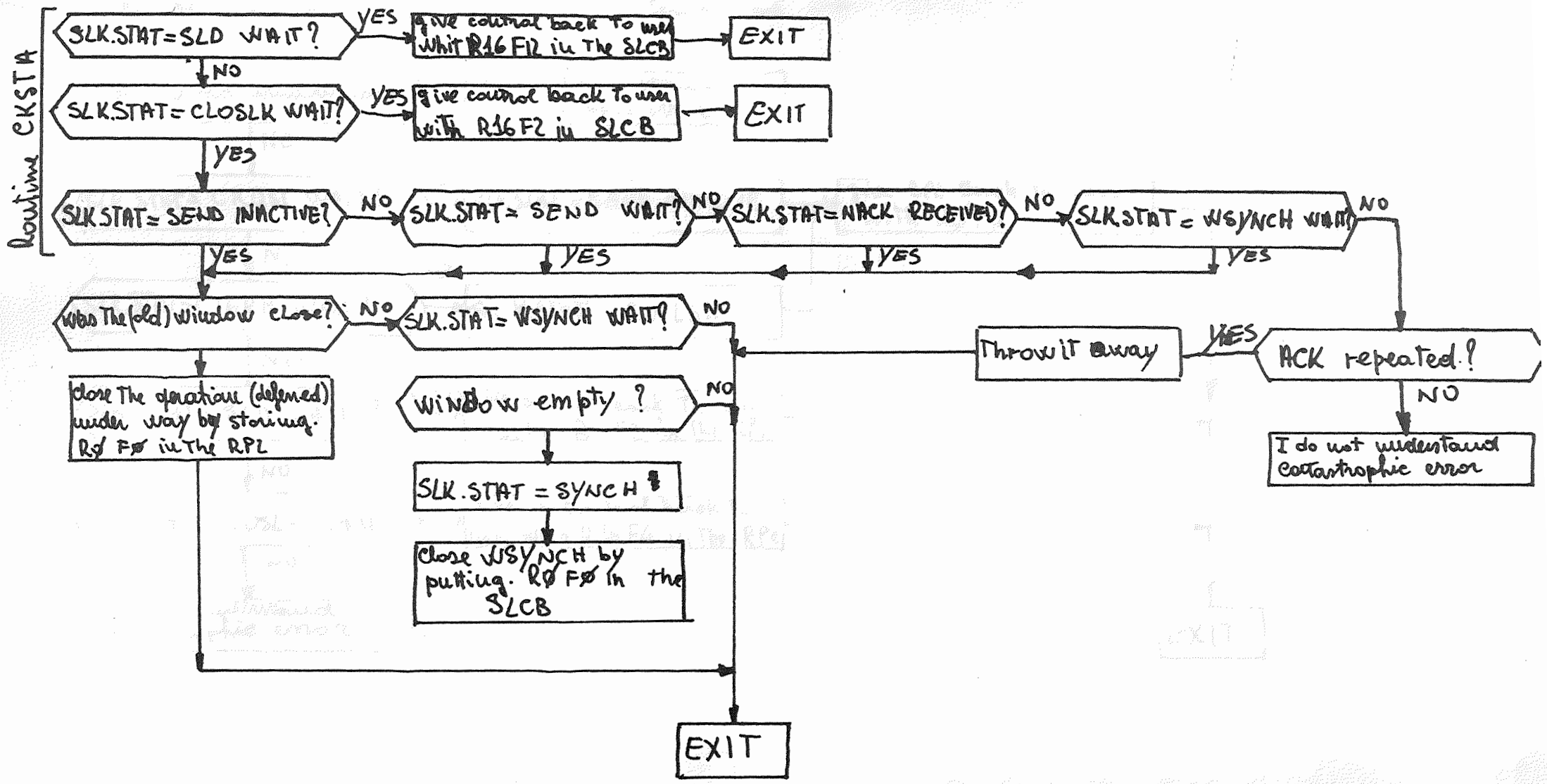
ENTRY ANMRcv



* The updating of the window is different for ACK or NACK received

SLH.MASTER.CONTROL (from ANM.MASTER.RECEIVE)
ENTRY SLACTR (SLH ACK CONTROL) ACK (Type 1,4)

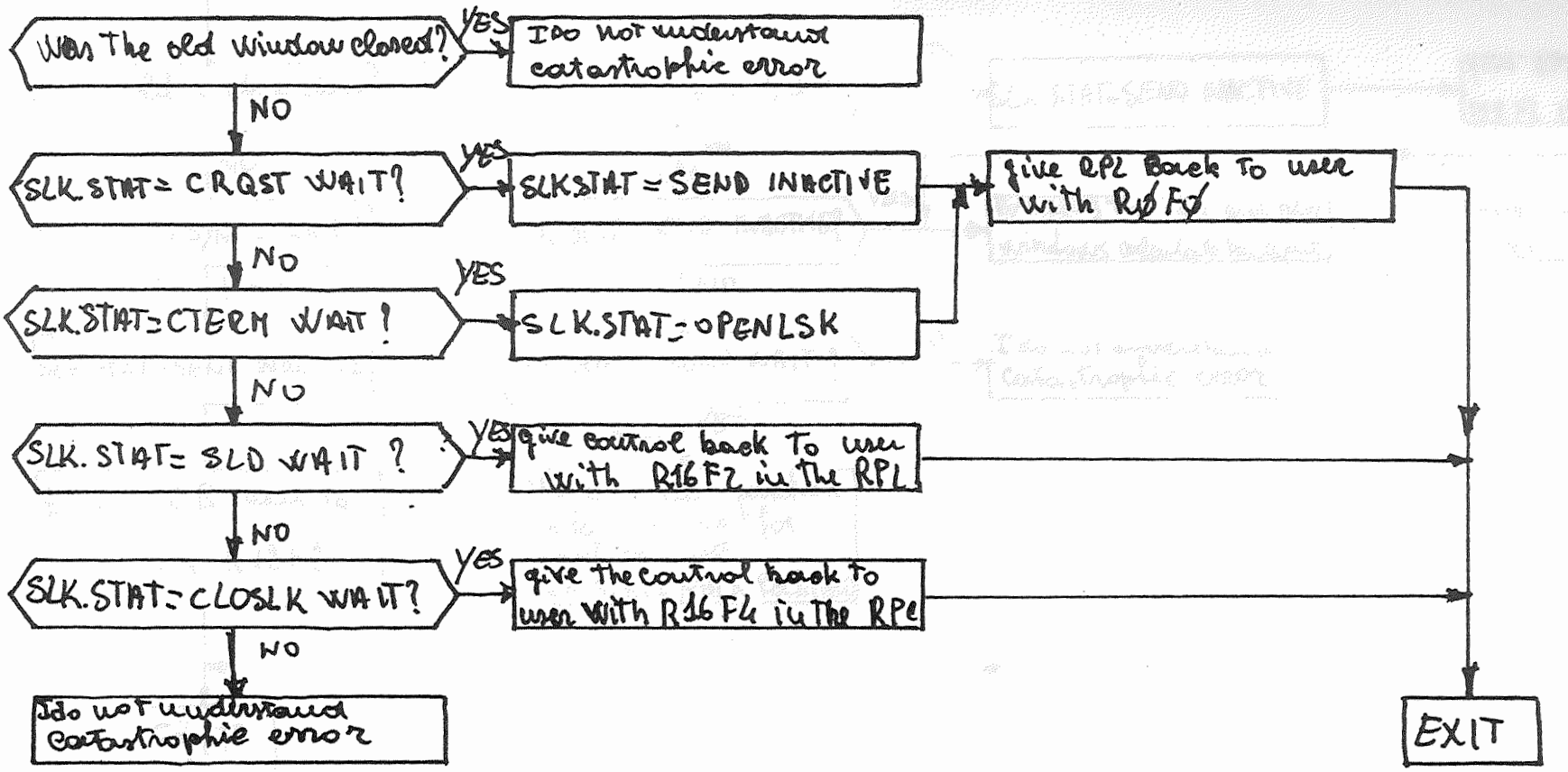
routine CKSTA



SSM.MASTER.CONTROL (from ANH.MASTER.RECEIVE)

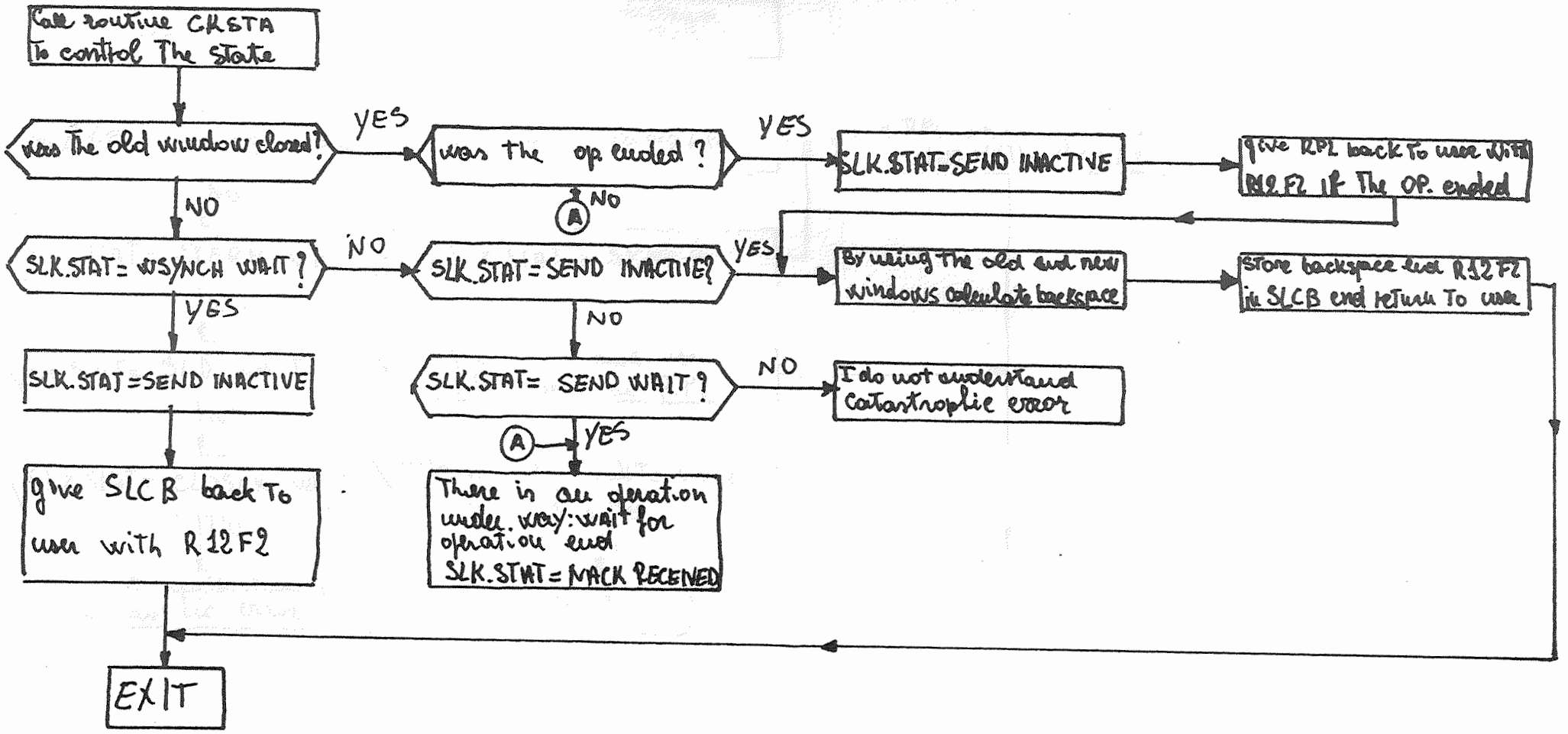
Entry SSACTR (SSM ACK CONTROL) ACK (Type 2,3)

The only possibilities SSACTR is entered are that CTERM or CRQST were issued



SLH.MASTER.CONTROL (from ANH.MASTER.RECEIVE)

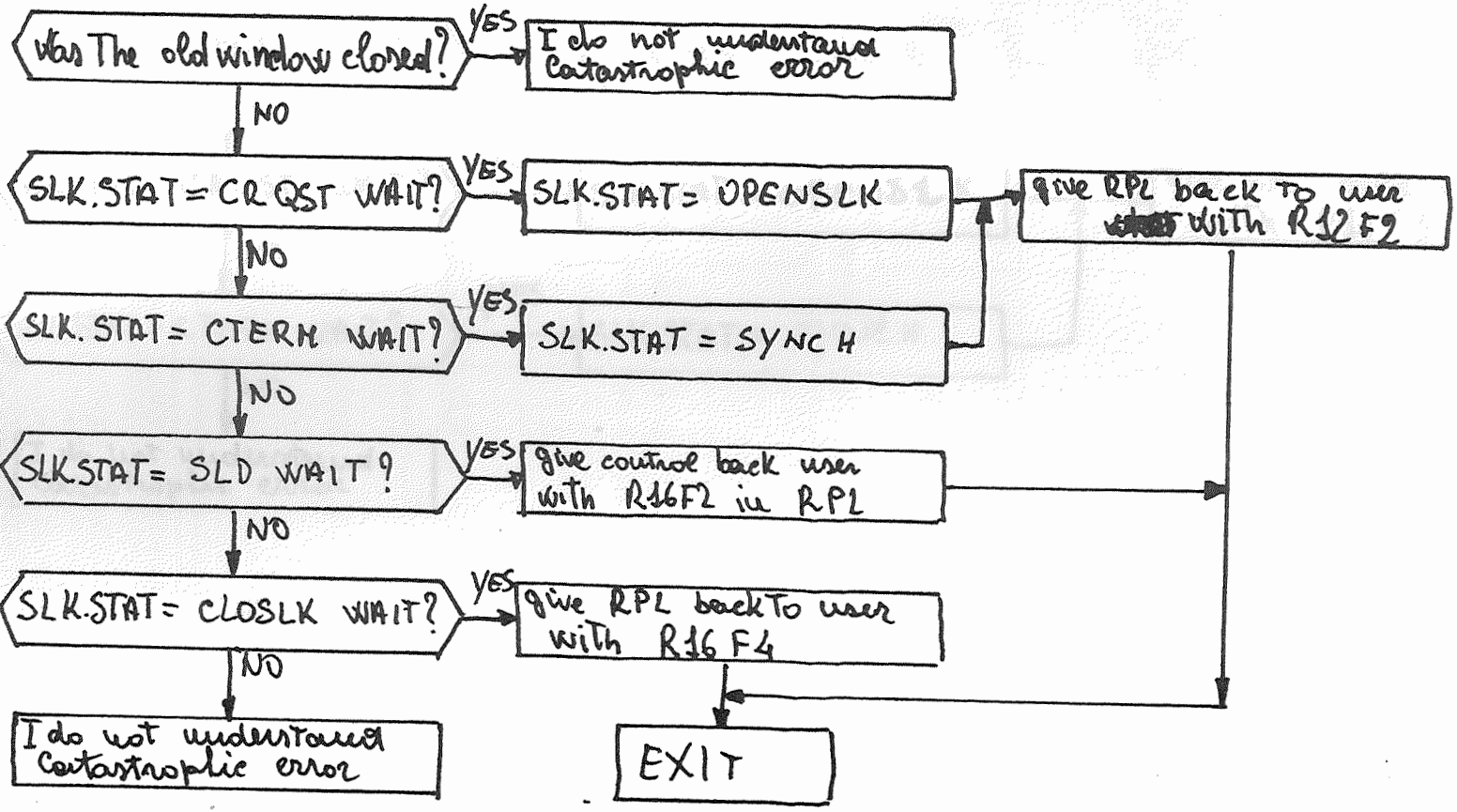
Entry SLNCTR (SLH NACK CONTROL) NACK (Type 1,4)



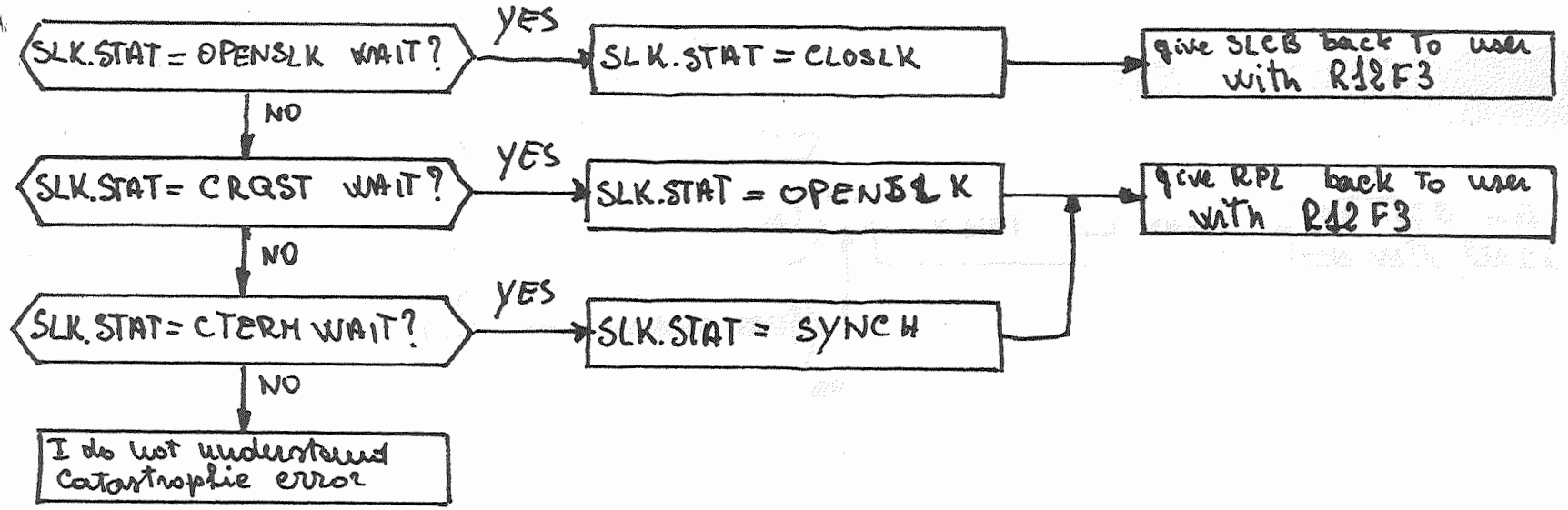
Note If. The old window was in the close state The NACK just received will find The SLK.STAT finite state Machine in The SEND WAIT State only

SSM.MASTER.CONTROL (from ANM.MASTER.RECEIVE)

Entry SSNCTR (SSM NACK CONTROL) NACK (Type 2,3)

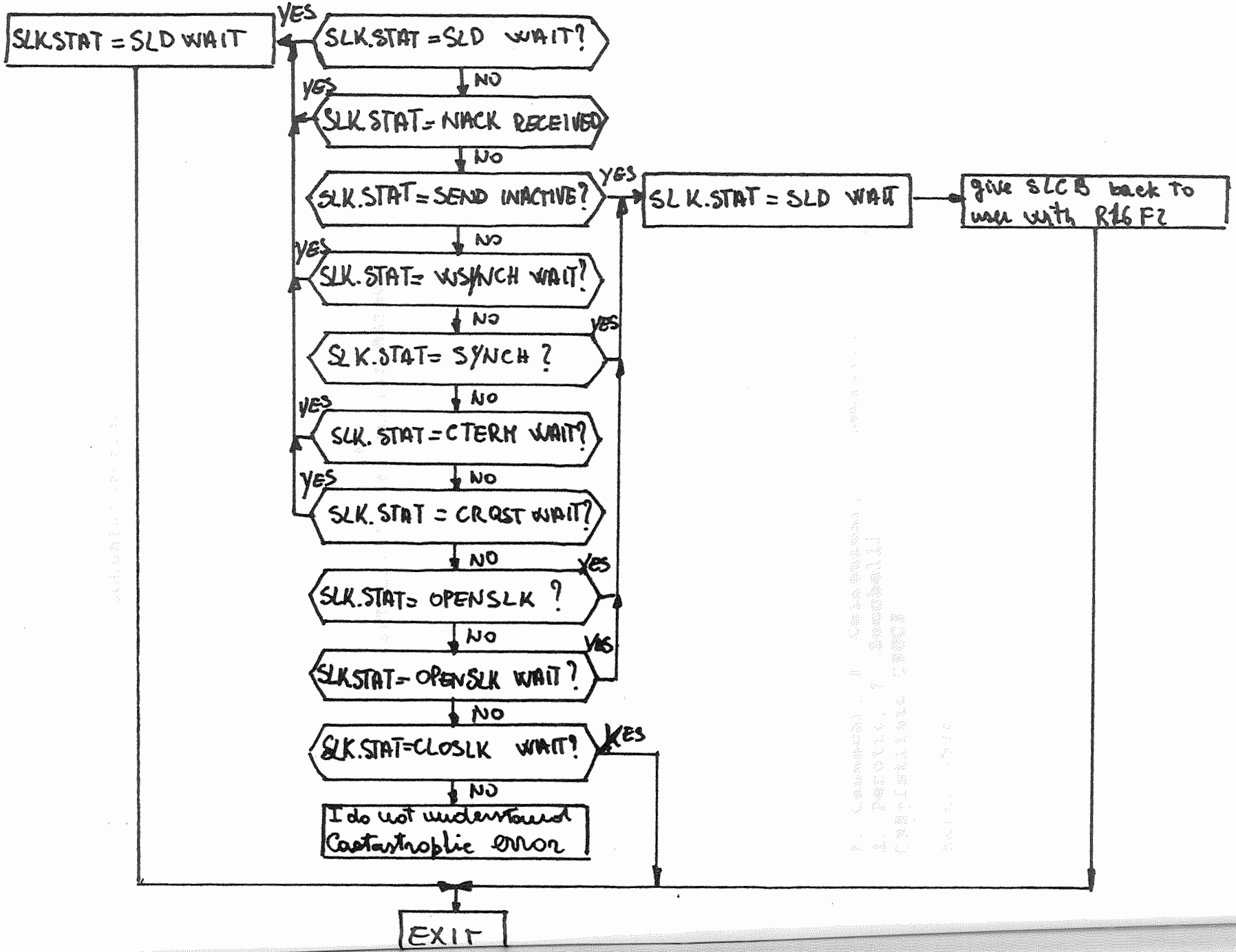


TIMEOUT MASTER



SSM.MASTER

ENTRY SLD WNM (Satellite Link DOWN Master)



STELLA

SATELLITE TRANSMISSION EXPERIMENT LINKING LABORATORIES

DOCUMENT NUMBER A/3/78

SLAM
SATELLITE LINK ACCESS METHOD

F. Caneschi, N. Celandroni, L. Lenzini,
E. Perotto, E. Zucchelli
CNR-Istituto CNUCE

March 1978

Abstract

The Satellite Link Access Method (SLAM), is a functional unit that allows two application programs to communicate by using the OTS Link facilities.

This document, in its first version, assumes that the reader is familiar with the RSX11M MACRO Language, and wishes to know specific details for the PDP 11 implementation of SLAM.

CONTENTS

1. INTRODUCTION

1.1 GENERAL CONSIDERATIONS

(1) On all macros that generate calls on the SLAM functions, there is an optional parameter LOGERAD which allows the user to specify a logical error return address. If this parameter is not specified, then return is to the location following the macro call.

(2) The various modes of returning to the caller of a SLAM macro are tailored to the PDP 11, and in general are very simple. In general, if an operation is not accepted, there is an "immediate" return to the location after the macro call, or to the logical error return address (LOGERAD) if the error is a logical error. If an operation is accepted by SLAM, then the user has three return options. For some macros not all of these options are available. The OPTION=SYN parameter specifies that there is no immediate return if the operation is accepted, and when the operation is completed there is a return to the user after the macro call, with all registers preserved except RX, which will contain the return code, and index register 7, which is destroyed for all macro calls. This option corresponds to an internal ECB with SLAM itself doing the CHECK operation. The second return option can be specified by OPTION=ASY, and will by default have an internal ECB in the RPL. If SLAM accepts the macro, an immediate return is made to the caller after the macro call, with RX set to zero. At a later time the user must do a CHECK operation for the RPL. When the operation is complete, an immediate return is made after the call of the CHECK operation. The third method of return is to provide a parameter of the form EXIT=address, where the address is a return address for the emt routine. If the operation is accepted, an immediate return occurs as for the second case. When the operation is complete, an EMT return is made to the address specified for the operation. In that case, the only registers whose contents are defined are register 5, which addresses the control block used for the operation (RPL). In summary, there is no user specified ECB available, and the EXIT option follows the EMT conventions.

(3) Asynchronous exits to the user are specified as a part of the SLCB information, and do not include a return for logical errors. The three exit addresses (for NACKs received, for Satellite Link going down, and for broadcast messages) are branched to directly from SLAM to the

application, with RX set to the return code, and register 5 set to the appropriate SLCB. Because there is a possibility of several events waiting to be given to the application at the same time (they are engueued in SLAM), there is a convention about clearing the return code in the SLCB so that it can be used for further asynchronous signals.

(4) The only parameters to the macros which can be in registers are the buffer address, and an implied control block address being in register 5 if its address is not mentioned in the macro call.

(5) Return codes are normally in RX (except for the EXIT return) and if appropriate, also in the control block being returned.

1.2 THE ORGANIZATION OF THE REST OF THE DOCUMENT

The rest of this document is organized as follows. There is a discussion of the control blocks and the operations, to put things into perspective, after which the details of the parameters which are used by more than one macro are described. This is followed by a description of the individual macros. Then the return codes are listed in detail with their PDP 11-oriented meanings.

2. MACRO PARAMETERS

In this section the macro parameters which are used by more than one macro are discussed. If their meaning can be different from the usual one discussed here, that meaning is given with the relevant macro description.

APPLID='application name'

The designated application name, normally enclosed in primes, is a one to eight character designation of an application.

AREA=address|{(index)}

The AREA parameter designates the place where the address of a message buffer may be found. The simple address form specifies a location where the buffer address is to be found. The (index) form specifies that the address is in

the designated index register. The index expression must result in a value from 0 to 7 only.

AREALEN=receive count limit

This parameter designates the maximum message length that can be received with a RECEIVE operation.

CRQSTMSG=YES|NO

This parameter designates whether a CALL message is to be sent or is required for an INVITE to complete. See the SLACB and the CRQST macros.

BLK=SLACB|SLCB|RPL

In the MODCB and GENCB macros, this designates the type of block.

BRMAD=broadcast message receive address

This parameter designates where control is to be given in the application when a broadcast message is received. See the section on asynchronous events for details.

DATALEN=maximum record length

This parameter specifies a maximum record length associated with the Satellite Link operations.

DESTID='destination name'

This parameter specifies the one to eight character name of the destination earth station.

ECB=0

The ECB, EXIT and OPTION parameters must be considered together for their combined meaning. ECB=0 designates that an internal event control block is used for the RPL operation for an asynchronous return (OPTION=ASY). The syntax of the ECB parameter is such that a future extention could allow user designated ECBs. The ECB parameter cannot be coded with the EXIT parameter. It is redundant when coded with OPTION=SYN. For the OPTION=ASY and ECB=0 combination, the following action occurs. After an RPL operation is accepted, there is an immediate return to the

application with return code zero. The application does a CHECK operation to designate when and where it is ready to receive the operation complete return. See the CHECK operation for more details.

NACKRAD=asynchronous error (NACK received) address

This parameter specifies an address in the application where control is given asynchronously if error information (backspace for tape to tape) is to be given to the application. See the section on asynchronous events for details.

EXIT=operation end return address

This parameter must be considered together with the ECB and OPTION parameters. It cannot be coded with OPTION=SYN or with ECB. It designates the address used to EMT the RPL back to the user at operation completion time. When control comes back, the return code in in the RPLRTNCD field and the RPL address is in register 5.

LOGERAD=logical error return address

This parameter designates where control is to be returned if a requested operation has a logical error. All logical errors are checked for before an operation is allowed to take place. For all logical errors, the return code will be 20, and return code and feedback will be in RX. The section on return codes and methods goes into more details. If the LOGERAD parameter is missing or zero, the return is to the first location following the macro call.

CRAD=Catastophic error exit address

This parameter designetes where control is to be given when an unrecoverable error occurs. See the section on asynchronous error exits for details.

SLCB=slcb address

This parameter designates the address of an SLCB. For RPL operations, it designates the SLCB to use with the RPL. For SLCB operations (OPENSLK and CLOSLK), it designates the SLCB to be open or closed. In the latter cases, if it is omitted or zero, the SLCB address is assumed to be in register 5 when the OPENSLK or CLOSLK call is made.

SLACB=slacb address

This parameter designates the Satellite Link Application Control Block address.

OPTION=SYN|ASY

This parameter specifies the return option for the case where an RPL operation has been accepted. It must be considered together with the ECB and EXIT parameters. The SYN option has an implied internal event control block in the RPL, and designates that SLAM must do a CHECK operation instead of an immediate return after accepting the operation request. When the operation completes, return is made after the macro call. For the ASY option, either the ECB or the EXIT parameters, but not both, may be designated. The EXIT option is explained under that parameter description. If EXIT is omitted, and ASY is specified, then the default is the ECB=0 designation, if the ECB parameter has not been specified. See the ECB parameter for more details.

PASSWD='password'

This parameter specifies a password to be used for making connections to an application. The password must be one to eight characters.

RPL=rpl address

This parameter specifies the address of the request parameter list (RPL). For ordinary RPL operations, if this parameter is omitted or zero in value, SLAM assumes that the RPL address is in register 5.

symbol

This parameter can appear in the name field of most macros. If it is the SLACB, SLCB or RPL macro, it represents the address of the control block. For other macros, it represents a label on the first instruction of an executable macro.

3. SLAM MACROS

The SLAM macros can be classified in several ways. The first section defines macros that are declarations, that is, they have their effect at assembly time by generating control blocks or definitions which are useful or necessary for the application. The remaining macros are executable by the application, and can be grouped into ordinary RPL macros, control block modification macros, and miscellaneous macros. Although the strict syntax implies that the parameters can only be coded in the order shown with each macro definition, in fact the keyword parameters can be coded in any order, separated by commas. Optional parameters are denoted by square brackets. Besides the macro names in the following sections, certain other macro names are reserved for the use of SLAM. They are: xxxxx,.....,..

3.1 DECLARATION MACROS

3.1.1 SLACB

```
symbol   SLACB APPLID='application name'
              [,PASSWD='password']
              [,CRQSTMSG=YES|NO]
```

This macro generates a Satellite Link Application Control Block. The PASSWD and CRQSTMSG parameters refer to requirements that a received CALL request must match before an INVITE for that application will accept the CALL request from another application. The symbol parameter addresses the first word of the block, and is the name of the block.

3.1.2 SLCB

```
symbol   SLCB [SLACB=slacb address]
              [,CRAD=catastrophic error exit address]
              [,BRMAD=broadcast message receive address]
              [,NACKRAD=asynchronous exception address]
              [,DATALEN=maximum record length]
```

This macro creates a Satellite Link Control Block. The symbol parameter is the name of the block, and is the address of its first word.

3.1.3 RPL

```
symbol   RPL [SLCB=slcb address]
             [,AREALEN=receive data count limit]
             [{,OPTION=SYN}|
              {[,OPTION=ASY]
               {,ECB=0}|{,EXIT=op. end return address}}]
             [,DESTID='destination earth station name']
             [,APPLID='application name']
             [,PASSWD='password']
```

The RPL macro assembles a request parameter list control block. The symbol parameter is the label on the first word of the block, and is the name of the block to use in other macros. If omitted, the default for OPTION is SYN, unless ECB or EXIT has been specified.

3.1.4 RETCODES

RETCODES

This macro defines a number of EQUs which define useful symbols for return codes of SLAM. The symbols all have the form RxFy, where x is either 0, 4, 8, 12, 16 or 20, and y ranges from 0 to around 20. Rx represents the major return code and Fy the feedback code. In the PDP 11 Rx is in the left byte and Fy the right byte of any return code. Thus RETCODES provides a convenient symbolic definition of the return codes.

3.2 THE EXECUTABLE MACROS

There are several classes of executable macros, the normal RPL macros, the control block modification macros, and the miscellaneous macros. Each macro has its parameter syntax shown, then a description of the macro operation itself, then the return codes that can occur for the macro. See the parameter description section for the parameters that are not described for that specific macro. See the section on return codes for their meanings.

3.2.1 THE NORMAL RPL MACROS

3.2.1.1 CRQST

```
[symbol] CRQST [LOGERAD=logical error return address]
               [,RPL=rpl address]
               [,SLCB=slcb address]
               [,AREA=buffer address specification]
               [,CRQSTMSG=YES|NO]
               [{,OPTION=SYN}|
                {[,OPTION=ASY]
                 {,ECB=0}|{,EXIT=op. end return address}}]
               [,DESTID='destination earth station name']
               [,PASSWD='password']
```

The CRQST and INVITE macros are used for making connections. CRQST initiates a CALL request to the application and earth station designated by DESTID, APPLID and PASSWD.

After an INVITE has successfully completed an ACK is sent back.

If the application does not wish to accept the connection (after the inspection of the PASSWD, for example) or is not in the appropriate state then it sends back a NACK.

The return codes for CRQST are as follows:
(1) Immediate return codes:
LOG R20F1 R20F2 R20F5 R20F7 R20F8
IMMSTO R0F0 R4F3 R4F16 R16F1 R16F4 R8F1
(2) ENDOP return codes:
R0F0 R4F4 R4F5 R4F6 R8F1 R12F3 R16F1 R16F2 R16F3 R16F4

3.2.1.2 INVITE

```
[symbol] INVITE [LOGERAD=logical error return address]
                [,RPL=rpl address]
                [,SLCB=slcb address]
                [,BRDCST=NO|YES]
                [,DESTID='destination earth station']
                [{,OPTION=SYN}|
                 {[,OPTION=ASY]
                  {,ECB=0}|{,EXIT=op. end return address}}]
```

The INVITE operation waits for a CALL request from somewhere. If the application and password agree with the SLACB of the application, and if there is a broadcast message when the SLACB requires it, then the INVITE operation completes. If there was a broadcast message, the RPLAREA field of the RPL has its address, otherwise it has a zero. ******** The next statement must still be checked out!!!!!!! When the operation completes successfully, the application and location are stored in the RPLAPLID and RPLDEST fields of the RPL. If the application does not wish to accept the connection a NACK is sent back. When the INVITE is first requested, the BRDCST option may be used to designate that a broadcast message be sent to the requesting earth station (DESTID option).

The return codes for INVITE are as follows:
(1) Immediate returns:
LOG R20F1 R20F2 R20F5 R20F7
IMMSTO R0F0 R16F2 R16F4
(2) ENDOP returns:
R0F0 R16F1 R16F2 R16F4

3.2.1.3 CTERM

```
[symbol] CTERM [LOGERAD=logical error return address]
               [,RPL=rpl address]
               [,SLCB=slcb address]
               [{,OPTION=SYN}|
                {[,OPTION=ASY]
                 {,ECB=0}|{,EXIT=op. end return address}}]
```

This macro is used to end a session that connects two applications.

The CTERM return codes are as follows:
(1) Immediate return codes:
LOG R20F1 R20F2 R20F5 R20F7
IMMSTO R0F0 R4F15 R8F1 R16F2 R16F4
(2) ENDOP return codes:
R0F0 R12F3 R16F1 R16F2 R16F4

3.2.1.4 SEND

```
[symbol] SEND [LOGERAD=logical error return address]
              [,RPL=rpl address]
              [,SLCB=slcb address]
              [,AREA=buffer address designation]
              [{,OPTION=SYN}|
               {[,OPTION=ASY]
                {,ECB=0}|{,EXIT=op. end return address}}]
```

This macro designates the send operation. If no buffer is specified with AREA, its address is assumed to already be in the RPLAREA field of the RPL.

The return codes for SEND are as follows:
(1) Immediate returns:
LOG R20F1 R20F2 R20F5 R20F7 R20F8
IMMSTO R0F0 R4F17 R16F2 R16F4
(2) ENDOP return codes:
R0F0 R12F3 R16F1 R16F2 R16F4

3.2.1.5 RECEIVE

```
[symbol] RECEIVE [LOGERAD=logical error return address]
                 [,RPL=rpl address]
                 [,SLCB=slcb address]
                 [,AREALEN=data count limit]
                 [{,OPTION=SYN}|
                  {[,OPTION=ASY]
                   {,ECB=0}|{,EXIT=op. end return address}}]
```

This macro designates the receive operation. When the receive completes successfully, the buffer address is in the RPLAREA field of the RPL. The receive operation will use the AREALEN parameter as a limit on how long the received message is allowed to be. If the parameter is not specified, it is taken from the RPLALEN field of the RPL.

Return codes for the RECEIVE macro:
(1) Immediate returns:
LOG R20F1 R20F2 R20F5 R20F7 R20F9
IMMSTO R0F0 R4F9 R16F2 R16F4
(2) ENDOP return codes:
R0F0 R12F3 R16F1 R16F2 R16F4

3.2.1.6 MAIL

```
[symbol] MAIL [LOGERAD=logical error return address]
              [,RPL=rpl address]
              [,SLCB=slcb address]
              [,AREA=buffer message address designation]
              [{,OPTION=SYN}|
               {[,OPTION=ASY]
                {,ECB=0}|{,EXIT=op. end return address}}]
```

This macro causes a broadcast message to be sent. The AREA parameter is described in more detail in the section on parameters. It designates the message to be sent. If omitted, SLAM assumes the buffer address is already in the RPLAREA field of the RPL.

Return codes for the MAIL macro.
(1) Immediate returns:
LOG R20F1 R20F2 R20F5 R20F7 R20F8
IMMSTO R0F0 R4F18 R4F14 R16F2 R16F4
(2) ENDOP return codes:
R0F0 R16F1 R16F2 R16F4

3.2.2 CONTROL BLOCK MODIFICATION MACROS

3.2.2.1 GENCB

```
[symbol] GENCB BLK=SLACB|SLCB|RPL
               [,WAREA=address of space to use]
               [,LENGTH=length of space to use]
               [,LOGERAD=logical error return address]
               [,APPLID='application name']
               [,PASSWD='password']
               [,CRQSTMSG=YES|NO]
               [,SLACB=slacb address]
               [,CRAD=catastrophic error exit address]
               [,BRMAD=broadcast message receive address]
               [,NACKRAD=asynchronous exception address]
               [,DATALEN=maximum data count]
               [,SLCB=slcb address]
               [,AREALEN=maximum rpl receive data count]
               [{,OPTION=SYN}|
                {[,OPTION=ASY]
                 {,ECB=0}|{,EXIT=op. end return address}}]
               [,DESTID='destination earth station'
```

The GENCB macro is used to generate a control block when executed at run time. If WAREA and LENGTH are designated, it must be long enough to hold the requested block. If the WAREA is not designated, LENGTH has no meaning. Only a single copy of a block may be gotten with one call. If WAREA is not specified, the block address is returned in register 5. The remaining parameters have the same significance as for the SLACB, SLCB and RPL macros. The symbol parameter is a label on the GENCB macro call rather than a label (name) of the generated control block. For the first version the GENCB operation is not complete when generating an SLCB, and further work must be done before this option can be used. The GENCB operation makes only a single return, either to the logical error address or to the location following the macro call. Thus the OPTION, ECB and EXIT parameters only have meaning for setting information in a RPL.

Return codes for the GENCB macro:
(1) Immediate returns:
 LOG R20F1
 IMM R0F0
(2) There are no ENDOP returns for GENCB.

3.2.2.2 MODCB

```
[symbol] MODCB BLK=SLACB|SLCB|RPL
               [,LOGERAD=logical error return address]
               [,ADDR=address of block to modify]
               [,SLACB=slacb address]
               [,SLCB=slcb address]
               [,APPLID='application name']
               [,PASSWD='password']
               [,CRQSTMSG=YES|NO]
               [,CRAD=catastrophic error exit address]
               [,BRMAD=broadcast message receive address]
               [,NACKRAD=asynchronous exception address]
               [,DATALEN=maximum data count]
               [,AREA=message buffer address specification]
               [,AREALEN=receve data count message]
               [{,OPTION=SYN}|
                {[,OPTION=ASY]
                 {,ECB=0}|{,EXIT=op. end return address}}]
```

The MODCB macro modifies the contents of the type of block specified by the BLK parameter. The control block address is specified by the ADDR parameter, and if omitted, the address is assumed to be in register 5 upon call of the macro. The parameters shown only apply to the type of block for which they are defined. Most of these can be determined from the definitions of the SLACB, SLCB and RPL macros, and the remainder, such as AREA, are those that can be set in the normal executable RPL macros, such as SEND or MAIL. This macro makes an immediate return only, either to the logical error return address, or immediately after the macro call. Thus the OPTION, ECB and EXIT parameters refer to what may be set in an RPL.

Return codes for the MODCB macro:
(1) Immediate returns:
 IMM R0F0 R8F1
(2) There are no ENDOP returns for this macro.

3.2.3 MISCELLANEOUS EXECUTABLE MACROS

3.2.3.1 OPENSLK

```
[symbol] OPENSLK [LOGERAD=logical error return address]
                 [,SLCB=slcb address]
```

This macro opens a Satellite Link Control Block. If its address is not given by a SLCB parameter, it is assumed to be in register 5.

Return codes for the OPENSLK macro:
(1) Immediate return codes:
 LOG R20F1
 IMMSTO R0F0 R16F2
(2) ENDOP return codes.
 R0F0 R12F3

3.2.3.2 CLOSLK

```
[symbol] CLOSLK [LOGERAD=logical error return address]
                [,SLCB=slcb address]
```

This macro closes the Satellite Link identified with the SLCB parameter. If the SLCB parameter is omitted, the SLCB address is assumed to be in register 5. There is only one return made by CLOSLK, either an immediate return or an asynchronous return. The immediate return is when the close cannot be done, and will be either to the location after the macro call or to the logical error return. In some cases, the return is made asynchronously to the location after the macro call, and these can be determined by the return code in RX. In the case of the asynchronous return no registers are preserved, and RX is set to the return code, and register 5 is set to the address of the SLCB.

Return codes for the CLOSLK macro:
(1) Immediate returns:
 LOG R20F1
 IMM R4F13 R8F1 R16F4
(2) ENDOP return codes:
 R0F0

3.2.3.3 WSYNCH

```
[symbol] WSYNCH [LOGERAD=logical error return address]
                [,SLCB=slcb address]
```

This macro synchronizes the sender with the receiver. WSYNCH completes successfully when all ACKs, referring to messages previously sent are returned and correctly received by the sender. The operation fails if a NACK is received. If the SLCB parameter is omitted, the SLCB address is assumed to be in register 5. There is only one return made by WSYNCH, either an immediate return or an asynchronous return. The immediate return is when the WSYNCH cannot be done, and will be either to the location after the macro call or to the logical error return. In some cases, the return is made asynchronously to the location after the macro call, and these can be determined by the return code in RX. In the case of the asynchronous return no registers are preserved, and RX is set to the return code, and register 5 is set to the address of the SLCB.

Return codes for the WSYNCH macro:
(1) Immediate returns:
 LOG R20F1
 IMM R4F8 R16F2 R16F4
(2) ENDOP return codes:
 R0F0

3.2.3.4 CHECK

```
[symbol] CHECK [LOGERAD=logical error return address]
               [,RPL=rpl address]
```

This operation is only meaningful if an RPL is used with the OPTION=ASY and ECB=0 parameters. With those options an RPL operation (for example, SEND or CRQST) is requested by a macro such that an asynchronous return will be made at the point where the application does a subsequent CHECK. When the previously designated RPL operation completes, a return is made after the CHECK macro, with registers restored as for the CHECK operation. The only register values changed are RX, which has either the return code for the RPL operation or else a return code for the CHECK operation itself, and register 7, whose value is destroyed. If the return code is for the CHECK, it is either due to a logical error, a lack of save area, or an incorrect RPL designation. For the CHECK macro, if the RPL parameter is not given, the RPL address is assumed to be in register 5.

Return codes for the CHECK macro:
(1) Immediate returns:
 LOG R20F1 R20F4
(2) ENDOP returns:
 R8F1 can occur as an ENDOP, as well as any ENDOP return defined for the operation being CHECKed.

3.2.3.5 CANCEL

```
[symbol] CANCEL [LOGERAD=logical error return address]
                [,RPL=rpl address]
```

For certain RPL operations, the operation may be cancelled. Examples are RECEIVE and INVITE. An operation is cancellable only if it is in the right state. For example, a RECEIVE may not be cancellable if it is already receiving a message. CANCEL makes just one return, either an immediate return after the CANCEL macro call or a logical error return. The ACC is set to the result of the CANCEL attempt, and the remaining registers are as before the macro call is made. If the cancel is successful, no return is made for the RPL operation that was originally specified and is being cancelled. If the cancel is not successful, the original operation is not affected. The cancel can be unsuccessful for several reasons, such as logical errors, the operation being already complete (it could be on its way back via EMT in either the ECB or EXIT case), or the operation could be in a state where it could not be cancelled.

Return codes for the CANCEL macro:
(1) Immediate returns:
 LOG R20F1
 IMM R4F1 R4F12 R16F2 R16F4
 IMMSTO R4F2
(2) There are no ENDOP returns for this macro.

3.2.3.6 EXECRPL

```
[symbol] EXECRPL [LOGERAD=logical error return address]
                 [,RPL=rpl address]
```

In a few cases RPL operations may be rejected by SLAM but they may be retryable. EXECRPL may be used to retry such an operation, if desired. The returns (except via the CHECK macro or the EXIT parameter) will be made to the EXECRPL logical error return address or to the location following the EXECRPL macro, instead of the original macro.

Return codes for the EXECRPL macro:
(1) Immediate returns:
 LOG R20F7
 Any immediate return that can occur for the operation being retried can occur for this macro.
(2) Any ENDOP return that can occur for the operation being retried can also occur for EXECRPL.

4. RETURN TYPES AND RETURN CODES

The application gets control from SLAM in several ways. The two major ways are returns from requests made by the application, and asynchronous branches made to the application not associated with any request of the user.

4.1 RETURNS FROM APPLICATION REQUESTS

There are several return methods from application requests, and the application has some control over the methods of return. See the writeup of the ECB, EXIT and OPTION parameters for details of control methods. All requests have immediate returns and some requests have operation end returns. There are several classes of immediate returns, here called LOG (logical error return), IMM (normal immediate return), and IMMSTO (immediate return with stored return code). The normal method of handling return codes is as follows; exceptions to this exist for certain operations.

(1) If there is a logical error in a request, control is given to the address specified by the LOGERAD parameter. If it is missing, the location after the macro receives control. The return code is always in RX in this case.

(2) If any other error is discovered before the operation is detected then the error situation is signalled with an immediate return to the location following the macro call. The value of the return code will always be in RX in this case. In addition, if the return type is IMMSTO (not IMM) the code is stored in the status field of the control block involved in the operation (RPLRTNCD or SLCBRTNC).

(3) In all cases, if an operation is not accepted, only an immediate return is done (LOG, IMM, or IMMSTO). If an operation is accepted, normally an immediate return of type IMM with a return code of zero is done, followed at some later time with an operation end return (ENDOP). If the OPTION=SYN parameter is in effect, the IMM with zero return code is suppressed, and control is returned as discussed below for ENDOP returns.

(4) The operation end return can occur only after a level (wait) exit has been done and is possible only if the operation was accepted in the first place. The return code is always placed in the control block status field, and depending upon the return type chosen by the ECB, EXIT, and OPTION parameters, the return code may also be in RX. See these parameter writeups for more details.

(5) For OPTION=SYN, the ENDOP return restores all registers except index register 7, and set RX to the return code. The return is to the location following the macro that did the request.

(6) For OPTION=ASY and ECB=0, the ENDOP return is immediately after the CHECK macro, with all registers restored as before the CHECK macro except index 7, and with RX set to the return code.

(7) For OPTION=ASY and EXIT= exit address, the ENDOP return is the to exit address. In this case, the only register that is predictable is index register 5, which will address the RPL control block involved.

(8) At this point, some exceptions to the above will be discussed. In the case of CLOSLK, CHECK, and CANCEL only one return will occur, not two. For CLOSLK, there is no immediate return if the operation is accepted, and when it is completed, return is to the location following the macro, with RX zero, and index register 5 set to the SLCB address, all other registers being destroyed. For CHECK, LOG and IMM immediate returns are possible, as well as the normal return when the CHECKed operation ends, also appearing to be an immediate return, but in that case a level (wait) exit having taken place before the return to the application. CANCEL only has immediate returns.

4.2 SPECIFIC RETURN CODES

In SLAM the return code is represented as a 16 bit value, returned in RX and/or the status field of an SLCB or an RPL. The left byte is the return code, with values of 0, 4, 8, 16, or 20, to designate the class of the return code. The right byte has a so-called feedback code of value 0 through 20 or so. The codes may be defined with symbolic names of the form RxFy using the RETCODES macro. Such names will be used here in the descriptions of the return codes.

4.2.1 OPERATION END RETURN CODE

R0F0

This return code is used for immediate returns and ENDOP returns to designate the successful end of an operation, or to indicate, for an immediate return, that an operation request has been accepted and the application may expect a later ENDOP return.

4.2.2 SPECIAL CONDITION RETURN CODES

R4F1

The CANCEL of the operation is not possible, either

because CANCEL is not defined for that operation, or the operation is not in a cancellable state. The operation is not affected.

R4F2
The CANCEL of an operation was issued successfully.

R4F3
The destination earth station name is not defined.

R4F4
The CRQST operation could not complete because a broadcast message required by the other application was not sent by the CALL request.

R4F5
The CRQST operation was not successful because the password was incorrect.

R4F6
The CRQST operation was received in error by the requested application.

R4F7
An in-session request has been made for an SLCB that is not in session.

R4F8
A request that can only be made in send inactive state has been made when not in send inactive state.

R4F9
A request that can only be made in receive inactive state has been made when not in receive inactive state.

R4F10
A CTERM request was received while the SLCB was in receive wait state.

R4F11
A CTERM operation did not complete successfully.

R4F12
A CANCEL has been issued for an inactive RPL. It is possible for this to happen legally if the operation to be cancelled already has its RPL being returned by EMT, a case that can occur either for an EXIT or an ECB or the OPTION=SYN return.

R4F13
CLOSLK has been issued while an active RPL has not been CHECKed. It implies that the RPL cannot be returned to the application until a CHECK is done. Then the CLOSLK may be reissued by the application.

R4F14
MAIL has been rejected because of incorrect state.

R4F15
A request that can only be made in SYNCH state has been made when not in that state.

R4F16
A request that can only be made in OPENSLK state has been made when not in that state.

R4F17
A request that can only be made in SEND INACTIVE or SYNCH state has been made when not in those states.

R4F18
A MAIL was already under way.

4.2.3 RETRY RETURN CODES

R8F1
The operation could not be carried out because of temporary lack of storage. The operation may be retried after freeing some storage.

R8F2
The operation could not be carried out because of loss of access to the local earth station due to hardware malfunctions.

R8F3
This code is not supported by SLAM.

4.2.4 DATA LOSS RETURN CODES

R12F1
This code is not supported by SLAM.

R12F2
A negative response to a request has been received.

R12F3
Timeout.

R12F4
This code is not supported by SLAM.

4.2.5 ENVIRONMENT ERROR RETURN CODE

R16F1
The session partner has become unreachable, and the session is therefore closed. This can occur because the partner issued a CTERM or CLOSLK or because of loss of access to the other earth station due to physical loss of connection.

R16F2
The Satellite Link System is shutting down. The application should close its SLCB.

R16F3
The requested application is not available at the requested earth station location.

R16F4
The requested operation cannot be done since the application has issued a CLOSLK.

4.2.6 LOGICAL ERROR RETURN CODES

R20F1
The control block is invalid. This can occur, for example, when an RPL does not point to an open SLCB (if it should) or an SLCB is already open when an OPENSLK is attempted, and for similar errors, including the discovery that the control block address is invalid (for example, if it is zero). Not all such errors can be detected by SLAM but many can be.

R20F2
The EXIT option has been selected, but the address is zero in value.

R20F3
This return code is not used in SLAM.

R20F4
A CHECK was attempted for an RPL where EXIT=exit address was in effect, or where OPTION=SYN was in effect.

R20F5
An RPL operation was attempted when that RPL was still active, or while the designated SLCB of the RPL had another RPL active.

R20F6
This return code is not used in SLAM.

R20F7
An option of the request is invalid, or the request code itself is invalid (EXECRPL only), or the SLCB is in an incorrect state for the operation (and no special return code is defined for that situation).

R20F8
The specified output buffer was discovered to be invalid. For example, its address could be zero, or its message might have a negative length.

R20F9
The RPLALEN field has an invalid length value (zero or negative) for a RECEIVE operation.

5. CONVENTIONS FOR GIVING CONTROL TO THE APPLICATION

There are three classes of events where SLAM will give control to the application in an asynchronous fashion. They are (1) when a broadcast message is received (see the BRMAD parameter), (2) when error information (NACK) is received (see the NACKRAD parameter), and when errors are detected asynchronously by SLAM (see the ERAD parameter). In each possible case, control is given to the application at the appropriate address, with R5 set to the address of the SLCB, and with the status field of the SLCB (SLCBRTNC) set non-zero. In some cases, this non-zero value specifies the reason for the event in more detail. When SLAM gives control to the application with R5=SLCB, there are conventions to observe about the status field of the SLCB. The only way that SLAM can know when the SLCB can again be scheduled to be given to the application is for the application to clear the status field. It is preferable that this is done before the application does a level exit (wait).

5.1 ASYNCHRONOUS ERROR EXITS

The only events that cause the ERAD exit to be given control are represented by the return codes R16F1 (session partner became unreachable) and R16F2 (Satellite Link going down). For both cases, if there is an RPL active it will also be returned with that same return code. Note that the Satellite link going down indication will have priority over the session broken indication, if both occur at the same time. In the case of session broken, the SLCB is left in the OPENSLK state as if a CTERM had successfully completed. In the case of the Satellite Link going down, the application must issue a CLOSLK.

5.2 BROADCAST MESSAGE ASYNCHRONOUS EXIT

This exit to the BRMAD address specifies that there is a broadcast message in the SLCBBRKA field of the SLCB.

5.3 ERROR RESPONSE (NACK) ASYNCHRONOUS EXIT

This exit to the NACKRAD address specifies an error response has been received by SLAM for the application, and

further information is stored in the SLCB.

6. CSECTS FOR SLAM CONTROL BLOCKS

This section describes the format of the control blocks that the application uses to communicate with SLAM. Normally the fields are controlled by various SLAM macros, but there are times when the application needs to access these control blocks to change or to test the contents of certain fields. The NAMDSECT macro can be used to define DSECTs for these control blocks. Note that the application should not change any field contents or rely on field contents unless the application has well defined information about that field and how it may be used by the application.

6.1 SATELLITE LINK APPLICATION CONTROL BLOCK (SLACB) CSECT

SLACB Satellite Link Application Control Block

SBFLAG 2 bytes SLACB Flags
* CALL REQUEST message required

SBAPID 8 bytes Application Name
SBKWRD 8 bytes Password Required for the Application

SLCB Satellite Link Control Block

SLSLAC	2 bytes	Pointer to SLACB
SLRPL	2 bytes	Pointer to RPL
SLRTCD	1 byte	Return code
SLFDBK	1 byte	Feed Back code
SLFLAG	1 byte	SLCB Flags

* Broadcast menage received
* Recoverable (NACK) error received
* Catastroplic error detected (Satellite link Going Down)
* CALL REQUEST received
* CALL TERMINATION "

SLBSP	1 byte	Backspace
SRTNAD	2 bytes	Address of the following application entry
SBMRT	2 bytes	Broadcast Menage Routine
SNACKA	2 bytes	Backspace Routine
SCERR	2 bytes	Catastroplic Error Routine
SCRQST	2 bytes	CALL REQUEST Routine
SCTERM	2 bytes	CALL TERMINATION Routine
SMAXLT	2 bytes	Maximum Data Transfer Lenght

STH Satellite Transmission Header

SDES	1 byte	Destination Earth Station
SPSN	1 byte	Packet Sequence Number
SLBM	1 byte	Lenght of Broadcast Menage
SLPACK	1 byte	Last Packet Acknowledged
SLPRCV	1 byte	Last Packet received Correctly
SLWORK	n bytes	Work Area

RPL Request Parameter List

RSLCBA	2 bytes	Pointer to SLCB Block
RTNCD	1 byte	Return Code (General Return Code
RFDBK	1 byte	Feed Back Code (Specific Error Return Code
RPLREQ	2 bytes	RPL Request Codes

* OPENSLK
* INVITE
* CLOSLK
* CRQST
* SEND
* RECEIVE
* MAIL
* WSYNCH
* CTERM

RAREA	2 bytes	Pointer to data area
RLEN	2 bytes	Data Area Lenght
ROPT	2 bytes	Options for RPL Usage
SYN		Synchronous Handling
ASY		Asynchronous Handling
ECB		ECB Used
EXIT		Exit Used
PSN	1 byte	Packet Sequence Number
	1 byte	Reserved for future use
DES	8 bytes	Destination Earth Station
KEY	8 bytes	Authorization Keyword