FODA-TDMA satellite access scheme for mixed traffic. Implementation and testing features at 2Mbit/sec bit rate.
Final Report

Dr. Erina Ferro
Ing. Nedo Celandroni

CNUCE INSTITUTE
Via S.Maria 36 - 56100 Pisa (Italy)
Tel. +39-50-593246
Telex 500371 CNUCE

CNUCE Report C88-10 April 1988

Consiglio Nazionale delle Ricerche

CNUCE

PISA

Acknowledgements

The author would like to thank all the people involved in the realization of the entire system, i.e. people from the Rutherford LAB (UK), people from Marconi Reserch Center (U.K.), people from the Technical University of Graz. A thank to Dr. Mervyn Hine from the DD Division at Cern for his precious cohordination work among the partners of the project and with respect to the European Community.

I. The hardware environment

A satellite TDMA controller, named "satellite bridge" has been designed and developed which consists of four major elements:

a) The system bus, process, memory and communication support hardware. This system has been given a flexible design which allows most of the parameters of a TDMA system to be selected and altered under process control. The actual transmission and reception of bursts is handled by the communication support hardware and requires no intervenction by the process.
This flexible scheme ensures speed of operation with the ability to dynamically change the time frame, the number and the sizes of the transmission windows, the buffering and priority requirements of the data and the algorithms used in managing the satellite access scheme.
A most important feature of the design allows the modem and encoding rate to be changed dynamically during a transmission burst.

b) A variable rate digital burst modem which is capable of operating at 1, 2, 4 or 8 Mbit/sec inside a 5 Mhz frequency band. Different modulation schemes are used for different rates in this design. As the modem is mainly digital (after the first stage of analogue filtering), it is possible to implement these schemes using different tables in ROM (Read Only Memory). The modem is capable of dynamically changing its transmission rate.

c) An encoder/decoder implemented in VLSI and which is specifically adapted to the modulation scheme. The encoding rate may be changed on each sub-burst within a burst allowing different BER services for voice, video or data.
The combination of the modem and the encoder will deal with most fade conditions and will also allow a wide variety of BERs, between 10**-4 and 10**-9, for user services.

d) A satellite access scheme which provides both "stream" virtual circuits of known bandwidth to regular services such as voice and video and a "datagram" service for bursty traffic from the distributed computing system. The scheme allows for dynamic allocations of "stream" channels in a few hundred milliseconds and for FIFO ordering of "datagram" transmissions which normally allow for the immediate transmission of a datagram packet.

The satellite bridge can be summarized in such a way:

```
                   | From the                    /|\ To the
                   | satellite                    |   satellite
                   |                              |
                   V                              |
              +---------+                    +---------+
              |  Modem  |                    |  Modem  |
              +---------+                    +---------+
                   |                              |
              +---------+                    +---------+
              | Decoder |                    | Encoder |
              +---------+                    +---------+
                   |                              |
          +----------------+    PIO      +----------------+
          | Down process   |-------------|  Up process    |
          |(Motorola 68000)|-------------|(Motorola 68000)|
          +----------------+             +----------------+
                   | To the                      /|\ From the
                   |  LAN                         |    LAN
                   |                              |
                   V                              |
```

The mentioned satellite access scheme, named FODA-TDMA (Fifo Ordered Demand Assignment-Time Division Multiple Access), has been enterely developed by CNUCE and implemented in C language under the C-EXEC operating system. This report refers to the 2Mbit/sec implementation.

The reading of the CNUCE Report C85-3 is strongly recommended to well understand the FODA-TDMA acting, the network environment and the considered types of traffic.

II. The FODA system architecture on Motorola 68000

The system on both the Up and Down machines runs basically under C-EXEC V2.1b, a Unix like multitasking operating system.
In order to speed up the operations requiring very quick responses, they are performed at interrupt level. It means that the CPU works in system mode, at rather high priority level. It consents to avoid the overhead due to the operating system and the use of the entire set of machine istructions other than a dynamical management of the CPU priority, according to the operation under execution.
More precisely, all the operations relative to the satellite access method are performed by several processes entered by either hardware or software interrupts. The other processes called at interrupt level are the line drivers (terminal, Cambridge Ring interface) of the operating system.

All the other operations, such as the interface versus the operator, the LAN interface and the messages handling are performed by 3 different tasks named "upuser", "upring" and "msghandler" respectively for the UP process, and "downuser", "downring" and "msghandler" respectively for the DOWN process.
In this implementation the considered LAN is a Cambridge Ring.
In the currently used version of the C-EXEC operating system, tasks always work at priority level zero.

The interrupt level processes are in the following listed by decreasing priority order.

On the Up machine:

- DMA end of block: the DMA board has finished the transfer of a burst on satellite (interrupt level 6).

- PIO read: the transfer of the first word is completed (interrupt level 5).

- Timer: this interrupt occurs once per frame, in a fixed position. At the timer interrupt, the timer is reset for the next frame and its position is computed with respect to the frame counter (interrupt level 3).

- Terminal input: commands from the operator are entered (interrupt level 2). The task "upuser" handles the entered commands.

- LAN packet ready: the LAN handler has finished the read of a packet which is set up ready to be sent on satellite (interrupt level 1).

On the Down machine:

- DMA end of block: the DMA board has finished the read of a burst from the satellite (interrupt level 6). The system must begin to analyse the received data lowing the priority to 3.

- Timer: it occurs once per frame in a fixed position (interrupt level 5). In order to keep the timer cycle hanged to the frame cycle, the timer is set at the timer interrupt and it is tuned up on receiving of the reference burst. Such a procedure is needed because the frame-counter is not readable in the Down machine.

- RB UW detection: the unique word relative to the receiving of a reference burst is detected (interrupt level 4).

- PIO read: the transfer of the first word is completed (interrupt level 3. Unused).

- Terminal input: commands from the operator are entered (interrupt level 2). The task "downuser" handles the entered commands.

- LAN read: interrupt level 1. Unused.

Communications between interrupt level processes and C-EXEC tasks are needed in both ways. They are made possible according to the following procedures:

- Interrupt level process to C-EXEC task:

  the process uses the C-EXEC "ttyin" routine to put a byte in the pipe of the task. After the return from exception (RTE) instruction the system will alert the receiving task by its priority.

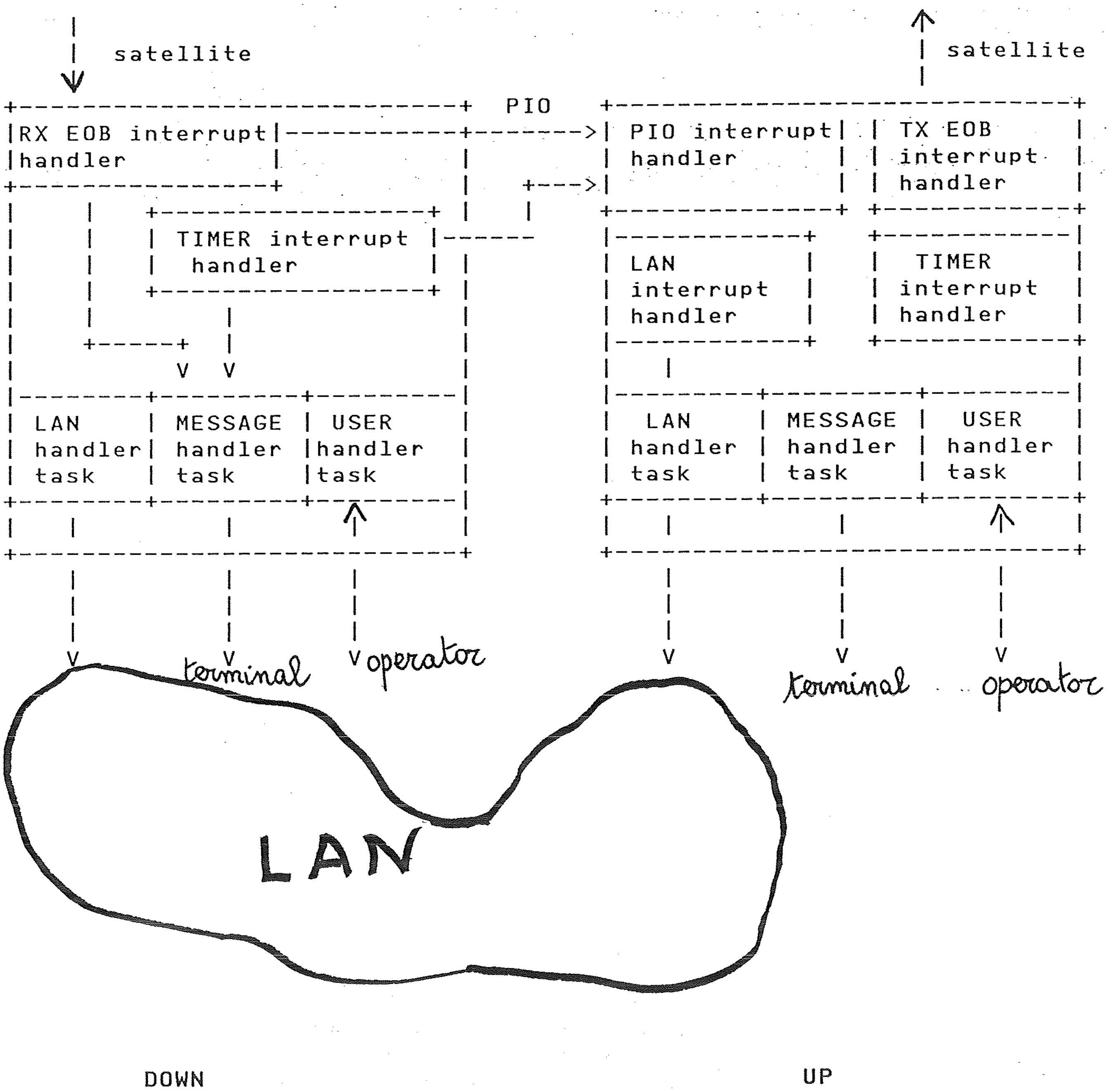
- C-EXEC task to interrupt level process:

  as a C-EXEC task is always running at system level, it can call directly routines which are normally executed at interrupt level.

A remark has to be pointed out about the PIA interface: the interrupt provoked by the reads of the first word is shared between the access method process and the C-EXEC driver. The two directions of the fork are taken according to the most significative bit of that word: the message belongs to the process if the bit is set on (in our implementation, the most significative bit), otherwise it belongs to the driver.

The PIO write is performed by a routine called at task or at system level. No interrupt is generated after the transfer of each word at the writing machine. As the writing provokes an interrupt and a switching to high priority at the reading machine, it has been taken care of performing also the write operation at rather high priority level, in order to make fast enough the message transfer.

The FODA software situation can be reassumed in the following picture:



With the terminology "DOWN process" all the FODA software running on the down box of the satellite bridge is referred to. "UP process" means the same on the up box.

III. The message handler task ("msghandler")

Most of the FODA software runs at interrupt level, so it is impossible to use the standard routines (errfmt, putfmt, etc.) furnished by the C language to print messages on terminal.
To solve this problem, the following solution has been adopted. A new task, named "msghandler", has been created on both the up and down processes and its entry point is the routine MSGhdl(). The definition of this new task has been added to the UPROC table (the user process table) defined respectively in the files TXCONFIG.C and RXCONFIG.C.

This task is started by means of the "pstart (process number in UPROC)" routine invoked by the user handler task (which is the only one automatically started by the operating system). The name of the user handler task is "upuser" and "downuser" respectively for the Up and the Down process, and its functions are defined respectively in the files UBU100.C and UBD100.C.

It must be underlined that the user handler task is only able to receive commands from the input terminal and, as conseguence of these commands, eventually to display information on the output terminal, but it is not able to display messages sent from the interrupt level software or from other tasks.

The "msghandler" task uses the device "wake2" as standard input, "tty0" as standard output and "tty1" as standard error output, allowing particular error or statistic messages be printed on a dedicated terminal and/or collected in a file.

From the interrupt level software, the msghandler task is waken up by means of the "w2sendchar(char)" routine, defined in the WAKE.C file, whose object module is included in the SATCXLIB.68K system library. This routine uses the "ttyin" system routine, which processes the input character "char" for the specified device unit. Any process waiting for read from the device specified in the ttyin routine is activated at the appropriate time.

The MSGhdl() routine performs different functions on the base of the input character "char", depending on which side it is implemented. This routine is defined in the files TXMSGH.C and RXMSGH.C respectively for the Up and the Down process.

IV. The user handler task

A communication is estabilished between the operator and each side of the satellite bridge by means of the user handler task, named "upuser" and "downuser" respectively on the Up and on the Down process. The definition of this task is in the files UBU100.C and UBD100.C respectively.

The logic of this task is the same on both the sides, i.e. generally to accept commands from the operator and to reply to them displaying some areas, but the accepted commands depend on the side on which the task is running.
The LAN interface task and the message handler task are activated by this task.

DOWNUSER
On the down side, the operator is requested to enter a valid physical address for the station. After that, the "downuser" task enters in a wait state for commands from the operator.
Available input commands are:

C : to display the rxcb area containing also the statistics.

E : to end the down process.

UPUSER
On the up side, the operator is requested whether or not the test fixed rate traffic generator (see third part) can be started. If yes, the traffic parameters are requested. In any case, the traffic generator cannot be started till when the station physical address is received via PIO from the down process. After that, the task enters a wait state for commands from the operator.
Available input commands are:

A : to change the traffic parameters of the fixed rate traffic generator ot to start it if not initially selected.

B : to create bulk bursts with the burst generator.

C : to display the "txstat" statistic area.

D = to display the areas containing the definition parameters for the delay measurement packet (DMP).

E : to end the up process.

F : to finish the traffic generator.

G : to create bulk + stream + interactive data bursts with the burst generator.

H : to stop the transmission of a DMP every "n" (specified) frames.

K : to create bulk + stream data bursts with the burst generator.

I : to create interactive data bursts with the burst generator.

L : to release the stream channels requested by the burst generator test application and by the Cambridge Ring voice application.

M : to create and transmit just one DMP.

N : to create and transmit one DMP every specified "n" frames.

Q : to display the number of data buffers waiting for transmission in the bulk, interactive and stream data queues.

R : to create and send a bulk buffer 4 chunks long.

S : to create stream bursts with the traffic generator.

T : to start the fixed rate traffic generator or to enter the number of stream channels (after opening the STREQ_AREA) in the case of the burst generator.

V : to display the stream channels requested and those really assigned.

W : to create stream + interactive bursts with the burts generator.

The DMP, the burts generator and the traffic generator facilities have been created to test the system performances. They are described in the following.

First section: the Down process implementation

The behaviour of the receiving side (Down process) of the satellite bridge, running the FODA-TDMA satellite access scheme, is documented in this section.

The Down process working areas and some particular behaviours are described.

The flow-charts of the most significative routines are presented in order to make easier the understanding of the process and the reading of the software.

1. The Down process areas

1.1 SCBTAB (Station Control Block TABle)

```
+---------------------+
|                     |
|       SCB #0        |
|                     |
+---------------------+
|                     |
|       SCB #1        |
|                     |
+---------------------+
|                     |
|       SCB #2        |
|                     |
+---------------------+
.                     .
.                     .
.                     .
.                     .
+---------------------+
|                     |
|       SCB #63       |
|                     |
+---------------------+
```

To each station a station control block (SCB) is associated which contains information regarding the particular active station.

SCBTAB is the table of all the SCBs. As the maximum allowed number of simultaneously active stations is 64, SCBTAB is constituted by 64 SCBs.

1.2 SCB (Station Control Block - 44 bytes long)

```
 bit-wide
           +-----------------------+
    8      |      status           |   scb_status
           |   (bit format)        |
           +-----------------------+
    8      |   station physical    |   scb_phyaddr
           |    address            |
           +-----------------------+
   32      |   datagram enqueuing  |   dnextlink
           |   forward pointer     |
           +-----------------------+
   32      |   datagram enqueuing  |   dbacklink
           |   back pointer        |
           +-----------------------+
   32      |   stream enqueuing    |   snextlink
           |   forward pointer     |
           +-----------------------+
   32      |   stream enqueuing    |   sbacklink
   -------------------------------------------16 bytes boundary
           |   back pointer        |
           +-----------------------+
    8      |   in which frame numb.|   drfn
           |   datag. req. received|
           +-----------------------+
    8      |   in which frame numb.|   srfn
           |   stream req. received|
           +-----------------------+
   16      |   datagram request    |   backlog
           |    (backlog)          |
           +-----------------------+
   16      |   stream channel      |   scb_streq
           |     request           |
           +-----------------------+
   16      |   hello messages      |   helloc
           |     counter           |
           +-----------------------+
    8      |   interactive traffic |   iexpfn
           |   expexted fragment # |
           +-----------------------+
    8      |   interactive buffer  |   ibid
           |     identifier        |
           +-----------------------+
   32      |   interactive data    |   ielstart
           |   incomplete buff.addr|
   --------+-----------------------+-------------16 bytes boundary
```

```
            +----------------------+
   8        | datagram traffic     |     dexpfn
            | expected fragment #  |
            +----------------------+
   8        | datagram  buffer     |     dbid
            |   identifier         |
            +----------------------+
  32        | datagram incomplete  |     delstart
            | buffer address       |
            +----------------------+
   8        | stream   traffic     |     sexpfn
            | expexted fragment #  |
            +----------------------+
   8        |   stream  buffer     |     sbid
            |   identifier         |
            +----------------------+
  32        | stream   incomplete  |     selstart
            | buffer address       |
            +----------------------+
```

1.3 CCB (Channel Control Block- 62 bytes long)

```
 bit-wide
          +-----------------------+
     32   |  datagram request     |
          |  queue head pointer   |   DRQ head
          +-----------------------+
     32   |  datagram request     |
          |  queue tail pointer   |   DRQ tail
          +-----------------------+
     16   |  datagram request     |
          |  queue counter        |   DRQ counter
          +-----------------------+
     32   |  stream  request      |
          |  queue head pointer   |   SRQ head
          +-----------------------+
     32   |  stream  request      |
     --------------------------------------16 bytes boundary
          |  queue tail pointer   |   SRQ tail
          +-----------------------+
     16   |  stream  request      |
          |  queue counter        |   SRQ counter
          +-----------------------+
     32   |  new born queue       |
          |    head pointer       |   NBQ head
          +-----------------------+
     32   |  new born queue       |
          |    tail pointer       |   NBQ tail
          +-----------------------+
     16   |  new born queue       |
          |     counter           |   NBQ counter
          +-----------------------+
     32   |  down station queue   |
     --------------------------------------16 bytes boundary
          |    head pointer       |   DSQ head
          +-----------------------+
     32   |  down station queue   |
          |    tail pointer       |   DSQ tail
          +-----------------------+
     16   |  down station queue   |
          |      counter          |   DSQ counter
          +-----------------------+
      8   |  coefficient of       |
          |  proportionality      |   cp (*)
          +-----------------------+
      8   *    alignment          *
          *                       *
          +-----------------------+
     16   | max. numb. of already|
          |assigned datag. slots |   ncsmax
          +-----------------------+
```

```
         +-----------------------+
  16     | min. numb. of already|
         |assigned datag. slots |   ncsmin
         +-----------------------+
   8     | stream upper bound    |
         |      limit            |   stubl
         +-----------------------+
   8     | stream total requests|
         |  (on 4 frames)        |   sttot
------+-----------------------+------------16 bytes boundary
   8     |timer interr. counter |
         |to prepare the SRB     |   srbtimer
         +-----------------------+
   8     |timer interr. counter |
         |to scan the hello msg |   hellotimer
         +-----------------------+
   8     |  number of the last   |
         |  received ref. burst |   lastrbn
         +-----------------------+
   8     *    alignment          *
         *                       *
         +-----------------------+
  16     |  already assigned     |
         |  datagram slots       |   alrass
         +-----------------------+
  16     |  datagram starting    |
         |  for frames ending #0|   dstart(0) *
         +-----------------------+
  16     |  datagram starting    |
         |  for frames ending #1|   dstart(1) (**)
         +-----------------------+
  16     |  datagram starting    |
         |  for frames ending #2|   dstart(2) (**)
         +-----------------------+
  16     |  datagram starting    |
         |  for frames ending #3|   dstart(3) (**)
         +-----------------------+
```

(*) The coefficient of proportionality is used by the master station to calculate the percentage of the datagram requests which must be allocated.

(**) These areas are initialized with a value of the datagram starting time computed as no stream allocations present.

1.4 RXCB (RX Control Block)

```
 bit-wide
              +-----------------------+
        8     |     my current        |    mycln (*)
              |   logical number      |
              +-----------------------+
        8     |     my previous       |    mypln
              |    logical number     |
              +-----------------------+
        8     |  my counter of the    |    framen
              |     frames            |
              +-----------------------+
        8     |  master or slave      |    role (**)
              |      role             |
              +-----------------------+
        8     |  initial temporary    |    rcb_status
              |  status (bit format)  |
              +-----------------------+
        8     |  flag indicating the  |    refbflag
              | receiving of a R.B.   |
              +-----------------------+
        8     |counter of consecutive|    missrbcounter
              | missing Ref. Bursts   |
              +-----------------------+
        8     *      alignment        *
              *                       *
              +-----------------------+
      800     |     STATISTIC         |    statarea
 (100 bytes)  |      AREA             |
              |                       |
              |                       |
              +-----------------------+
```

(*) 0 = master station
    0 < any slave station <= 64

(**) 0 = slave role
     1 = master role

The statistic area is 50 16-bits words long, each word being a counter of a particular event. The code number by which the routine is called is used as word offset inside the "rxcb.statarea" area to increase the corresponding counter.
The meaningful counters are the following:

```
                      RXCB.STATAREA
 bit-wide                                   code/word offset
                 +----------------------+
      16         | R.B. with a wrong    | WRONG_VERSION (0)
                 | version number       |
                 +----------------------+
      16         | R.B. received with   | WRONG_RB (1)
                 |  errors              |
                 +----------------------+
      16         | station erroneously  | ALREADY_ACTIVE (2)
                 | redeclared active    |
                 +----------------------+
      16         | station erroneously  | DECLARED_DEAD (3)
                 | declared dead        |
           ------+----------------------+------------16 bytes boundary
      16         | wrong traffic types  | WRONG_TRAFFIC (4)
                 |                      |
                 +----------------------+
      16         | headers with data    | GOOD_HDR_WITH_DATA (5)
                 | correctly received   |
                 +----------------------+
      16         | headers received     | WRONG_HDR (6)
                 | with errors          |
                 +----------------------+
      16         | R.B. correctly       | GOOD_RB (7)
                 | received             |
                 +----------------------+
      16         | control sub-bursts   | WRONG_CS (8)
                 | received with errors |
                 +----------------------+
      16         | headers without data | GOOD_HDR_WITHOUT_DATA (9)
                 | correctly received   |
                 +----------------------+
      16         | incompleted buffers  | INCOMPLETE_BUFFER (10)
                 |                      |
                 +----------------------+
      16         | buffers enqueued to  |  ENQUEUED_TO_CR (11)
                 | the LAN (Camb. Ring) |
           ------+----------------------+------------16 bytes boundary
      16         | data not for me      | DATA_NOT_FOR_ME (12)
                 |                      |
                 +----------------------+
      16         | data for me          | DATA_FOR_ME (13)
                 |                      |
                 +----------------------+
```

```
        +-----------------------+
16      | number of bursts      | B_BURST  (14)
        | of bulk data          |
        +-----------------------+
16      | number of bursts of   | I_BURST  (15)
        | interactive data      |
        +-----------------------+
16      | number of bursts of   | S_BURST  (16)
        | stream data           |
        +-----------------------+
16      | bad sub-bursts        | BAD_SCB  (17)
        |   counter             |
        +-----------------------+
16      | total missed ref.     | TOTAL_RBMISSED  (18)
        |   bursts              |
        +-----------------------+
16      | total (re)starts of   | TOTAL_INIDMA  (19)
        | the DMA               |
------+-----------------------+------------16 bytes boundary
16      | buffers generated by  | FRTG_DATA  (20)
        | the traffic generator |
        +-----------------------+
16      | HDR wrong len in the  | IS_NOT_HDRLEN (21)
        | control sub-burst     |
        +-----------------------+
16      | corrupted "rellen"    | INV_HDR_RELLEN (22)
        | field in good HDR     |
        +-----------------------+
16      | negative datagram     | NEGATIVE_MYDALLOC (23)
        | allocation            |
        +-----------------------+
```

Some other counters may be added in the future. In any case, their offset values may be found in the RXDEFINE.H file.

1.5 STATEL (STATion ELement)

```
bit-wide
             +--------------------+
   32        |    address of      |    s_elfwd
             |  next element      |
             +--------------------+
    8        |  logical number of |    logical
             | the station (or 0) |
             +--------------------+
    8        |  physical address  |    phyaddr
             |  of the station    |
             +--------------------+
    8        *     alignment      *
             *                    *
             +--------------------+
```

The STATEL blocks are used by the master Down process to be en-queued in the station new born queue (NBQ) and in the down station queue (DSQ).
NBQ and DSQ are contained in the CCB area (see 1.3).

1.6 FSQ (Free Station element Queue)

```
            s_head       s_tail     s_counter
          +----------+----------+----------+
          |  Head    |  Tail    |  Counter |
          | pointer  | pointer  |          |
          +----------+----------+----------+
               |            |
               |            |
               V            |
    +-----+----------+      |
    |     |  STATEL  |      |
    |     +----------+      |
    |                       |
    +---------+             |
              |             |
              V             |
    +-----+----------+      |
    |     |  STATEL  |      |
    |     +----------+      |
    |         .             .
    |         .             .
    |         .             .
    |         .             .
    |                       |
    +---------+             |
              |             |
              V             |
           +---------+<-----+
           |  STATEL |
           +---------+
```

1.7 Areas for the PIO communications with the Up process

The use of the following areas is described in chapter 6.

1.7.1 RBMYSTA (Reference Burst MY STream Assignment)

bit-wide		
8,8	stream starting slot number (from 1) in frames ending with 0	number of stream slots assigned in frames ending with 0
8,8	stream starting slot number (from 1) in frames ending with 1	number of stream slots assigned in frames ending with 1
8,8	stream starting slot number (from 1) in frames ending with 2	number of stream slots assigned in frames ending with 2
8,8	stream starting slot number (from 1) in frames ending with 3	number of stream slots assigned in frames ending with 3

This area is a sub-field of the SRBINF area (see 1.7.3).

1.7.2 RBMYDA (Reference Burst MY Datagram Allocations)

```
bit-wide
         +---------------------+---------------------+
8,8      | code=MY_DATAG_ALLOC | word length of the  |  code, len
         |                     | following data      |
         +---------------------+---------------------+
8,8      |R.B. number where the| number of the       |  dw_frame, dw_count
         |allocations were    | datagram entries    |
         +---------------------+---------------------+
16       |  1st datagram assignment starting time    |   myda(0)
         |                                           |
         +-------------------------------------------+
16       |  1st datagram assignment length           |   myda(1)
         |   (in multiple of 8-bits)                 |
         +-------------------------------------------+
16       |  2nd datagram assignment starting time    |   myda(2)
         |                                           |
         +-------------------------------------------+
16       |  2nd datagram assignment length           |   myda(3)
         |   (in multiple of 8-bits)                 |     .
         +-------------------------------------------+     .
16       |                 etc                       |     .
         |                                           |     .
         +-------------------------------------------+     .
16       |                 etc                       |
         |                                           |
         +-------------------------------------------+
         |                                           |
         .                                           .
         .                                           .
         .                                           .
         +-------------------------------------------+
```

This area is used by the slave Down process to pass to the Up process its own datagram allocations for the next frame.
Till to a maximum of 10 different datagram allocations are allowed in a frame to the same station (MYMAXDA) at 2Mbit/sec.

1.7.3 SRBINF (Superframe Reference Burst INFormations)

```
 bit-wide
           +-------------------------------+
     8     | code = SLAVE_SRB              |    code
           |                               |
           +-------------------------------+
     8     | word length of the following|    len
           |      data                     |
           +-------------------------------+
     8     | reference burst frame number|    frame
           |  (new superframe)             |
           +-------------------------------+
     8     | number of actual active       |    naas
           |   stations                    |
           +-------------------------------+
     8     | next available logical        |    naln
           | number                        |
           +-------------------------------+
     8     |  logical number assigned      |    lognumb
           |     to the station            |
           +-------------------------------+
     8     | current speed used            |    speed
           |                               |
           +-------------------------------+
     8     *     alignment                 *
           *                               *
           +-------------------------------+
    16     | satellite adjustment          |    adjustment
           |  (in bytes)                   |
           +-------------------------------+
    64     |    my stream assignment       |
(8 bytes)  |    (on 4 frames) table        |
           |                               |    rbmysta
           |      RBMYSTA                  |     area
           |      (see 1.7.1)              |
           |                               |
           +-------------------------------+
```

This area is used by the slave Down process to pass to the Up process information regarding the received new superframe reference burst.

1.7.4 MDALLOC (Master Datagram ALLOCations)

```
bit-wide
            +-------------------+--------------------+
  8,8       |code=MASTER_DALLOC |   word length of   | code, len
            |                   |the following area  |
            +-------------------+--------------------+
  16        |  station physical address              |
            |                                        |
            +----------------------------------------+
  16        |  starting time (in multiple of 8-bits) |
            |  of the datagram assignment            |
            +----------------------------------------+
  16        |  station physical address              |
            |                                        |
            +----------------------------------------+
  16        |  starting time (in multiple of 8-bits) |
            |   of the datagram assignment           |
            +----------------------------------------+
            .                                        .
            .                                        .
            .                                        .
            +----------------------------------------+
  16        |                0                       |
            |                                        |
            +----------------------------------------+
  16        |         frame size                     |
            |                                        |
            +----------------------------------------+
```

This area is used by the master Down process to pass to the Up process the global datagram allocations for the next frame.
Till to a maximum of 10 different datagram allocations are available in total in a frame (MAX_DALLOC) at 2 Mbit/sec.

1.7.5 REFBURST (prepared by the master Down process)

bit-wide		
8	code = MASTER_SRB	code
8	word length of the following area	len
8	destination station physical address	dest
8	destination sub-address	destsa
8	source station physical address	source
8	source station logical number	slogic
8	control information for the master (bit format)	control
8	reference burst length (multiple of 8-bits)	length
8	version number of the current FODA implementation	version
8	spare	
8	number of the frame which this REF. BURST refers to	framenumb
8	number of frames constituting the superframe	supframlen
8	next available logical number	rb_naln
8	number of actual active stations	rb_naas
8	new born station physical address	nbphya

```
              +-----------------------------------+
    8         | logical number assigned to the    |    nbln
              |     new born station              |
       -------+-----------------------------------+--------16 bytes boundary
    8         |   down station physical address   |    dsphya
              |                                   |
              +-----------------------------------+
    8         |  logical number of the down       |    dsln
              |      station                      |
              +-----------------------------------+
    8         |  logical number which has to be   |    hole
              |    filled  (hole)                 |
              +-----------------------------------+
    8         |  current speed used               |    rb_speed
              |                                   |
              +-----------------------------------+
   16         |   master round trip delay         |    mrtdoff
              |    offset (in bits)               |
              +-----------------------------------+
    8         |   number of stream assignments    |    sentries
              |   rounded to next multiple of 4   |
              +-----------------------------------+
    8         |  number of datagram assignments   |    rb_dentries
              |                                   |
              +-----------------------------------+----------------
    8         |   station physical address        |      beginning
              |                                   |        of
              +-----------------------------------+        the
    8         |   station physical address        |       STREAM
              |                                   |     assignments
              +-----------------------------------+         on
              .                                   .          4
              .                                   .       frames
              .                                   .
              .                                   .
              .                                   .
              +-----------------------------------+
    8         | physical address or 0 (rounded    |
              | to the next multiple of 4)        |
              +-----------------------------------+
```

This area is used by the master Down process to pass (at the beginning of a new superframe) to the Up process the new superframe reference burst to be transmitted. Only the fixed part of the reference burst, constituted by the first 22 8-bits bytes (from "dest" to "rb_dentries" included) plus the stream allocations for the whole superframe are contained in REFBURST. The datagram allocations, which must be passed at every frame, are communicated to the Up process in the MDALLOC area.
Till to a maximum of 16 stream channels can be allocated per frame. Therefore, the stream assignments contained in the reference burst can be at maximum 4 * 16 = 64 bytes long.

1.7.6 CLNAREA (send the Change Logical Number control message)

```
bit-wide
             +------------------------------+
   8         |     code = SEND_CLN          |    code
             |                              |
             +------------------------------+
   8         |     word length of the       |    len
             |     following area ( =1 )    |
             +------------------------------+
   8         |            spare             |
             |                              |
             +------------------------------+
   8         |     new logical number       |    cln_newlog
             |                              |
             +------------------------------+
```

1.7.7 SCLNAREA (stop sending the Change Logical Number control message)

```
bit-wide
             +------------------------------+
   8         |     code = STOP_CLN          |    code
             |                              |
             +------------------------------+
   8         |     word length = 0          |    len
             |                              |
             +------------------------------+
```

1.7.8 SITEINF (SITE INFormation)

```
bit-wide
          +-----------------------------+
   8      |     code = PHY_SITE         |    code
          |                             |
          +-----------------------------+
   8      |     word length of the      |    len
          |     following area (2 words)|
          +-----------------------------+
   8      |     my physical address     |    phya
          |                             |
          +-----------------------------+
   8      |         spare               |    site_spare
          |                             |
          +-----------------------------+
  16      |  station offset for the     |    tx_stoff
          |  slave TX                   |
          +-----------------------------+
```

The SITEINF area is filled at the very starting of the down process, when the operator is requested to enter the physical address of the station. The input value is stored in the "thissite" global area and copied in the "phya" field of the SITEINF area. The entered physical address is then checked for validity in the "site_tab" table, whose entries are as many as found before the final zero row.

Each entry in site_tab has the following format:

```
bit-wide
          +-----------------------------+
   8      |  station physical address   |    addr
          |                             |
          +-----------------------------+
   8      *       alignment             *
          *                             *
          +-----------------------------+
  16      |   station offset for        |    txstoff
          |    slave TX                 |
          +-----------------------------+
  32      |   station offset for        |    rxstoff
          |   master RX                 |
          +-----------------------------+
```

Therefore, the format of the site_tab table is the following:

addr	txstoff	rxstoff
addr	txstoff	rxstoff
.		.
.		.
.		.
addr	txstoff	rxstoff
0	0	0

Entries can be added simply adding new lines to site_tab, which is allocated in the RXALLOC.C file.
If the specified station physical address is invalid (not found in site_tab), the operator is requested to enter again the station physical address. If valid, the corresponding "rxstoff" value is stored in the STATION_OFFSET global field of the down process and the "txstoff" value is copied in the "tx_stoff" field of the "siteinf" area. The whole "siteinf" area is then passed to the Up process via the PHY_SITE PIO message.

1.7.9 FRAME_AREA

This area has been devoted to pass to the Up process, once per frame, the current frame number of the Down process for synchronization purposes.

```
bit-wide
         +-----------------------------+
   8     |  code = DOWN_FRAME          |    code
         |                             |
         +-----------------------------+
   8     |  len = 1 word               |    len
         |                             |
         +-----------------------------+
   8     |  current frame number used  |    rx_frame
         |  on the down process        |
         +-----------------------------+
   8     |  spare (PIO transmission    |    rx_spare
         |  requirement)               |
         +-----------------------------+
```

1.8 "STATUS" BYTE DEFINITIONS

The following is the bit format of the "status" byte in both the SCB and the RXCB areas.

```
+----+----+----+----+----+----+----+----+
| x  | x  | IN | SG | CL | ST | DR | AC |
+----+----+----+----+----+----+----+----+
 +7    6    5    4    3    2    1    0-
```

AC : IS_ACTIVE bit: the station is active;
DR : IS_DREQP bit: a datagram request is enqueued in DRQ;
ST : IS_SREQP bit: a stream request is enqueued in SRQ;
CL : IS_CLNP bit: the station is changing its logical number;
SG : IS_STARTING bit: the station is in the starting phase;
IN : IS_INITIALIZING bit: the station decided to start as master but the sent null reference burst has not yet been received back. As soon as it will be, this bit will be set off and the IS_ACTIVE bit will be finally set on;
x : spare bit.

1.9 Format of some areas as received from satellite

1.9.1 The Reference Burst

bit-wide		
8	destination station physical address	dest
8	destination sub-address	destsa
8	source station physical address	source
8	source station logical number	slogic
8	control information for the master (bit format)	control
8	reference burst length (multiple of 8-bits)	length
8	version number of the current FODA implementation	version
8	spare	
8	number of the frame which this REF. BURST refers to	framenumb
8	number of frames constituting the superframe	supframlen
8	next available logical number	rb_naln
8	number of actual active stations	rb_naas
8	new born station physical address	nbphya

8	logical number assigned to the new born station	nbln
8	down station physical address	dsphya
8	logical number of the down station	dsln
8	logical number which has to be filled (hole)	hole
8	current speed used	rb_speed
16	master round trip delay offset (in bits)	mrtdoff
8	number of stream assignments rounded to next multiple of 4	sentries
8	number of datagram assignments	rb_dentries
8	station physical address	beginning of the STREAM assignments on 4 frames
8	station physical address	
	. . . . .	
8	physical address or 0 (rounded to the next multiple of 4)	
	datagram allocations . . see 1.7.4	beginning of the DATAGRAM allocations for next frame
16	0	
16	frame size	

The format of the datagram allocations is exactly that one described in 1.7.4 when the datagram allocations are prepared by the master station to be transmitted in the reference burst.

1.9.2 The Header (Satellite Header + LAN header)

```
bit-wide  7+                      0-
          +-----------------------+------------------------+
   8      | destination physical  |   dest      SATELLITE  |
          |   address             |               HEADER   |
          +-----------------------+                        |
   8      |      destination      |   destsa               |
          |      sub-address      |                        V
          +-----------------------+
   8      |  source physical      |   source
          |    address            |
          +-----------------------+
   8      |  source logical       |   slogic
          |   number              |
          +-----------------------+
   8      |  control information  |   control
          |       field           |
          +-----------------------+
   8      | data information      |   datainf
          |    field              |
          +-----------------------+
   8      | data fragmenting      |   fragment
          |    information        |
          +-----------------------+
   8      |   data buffer         |   sh_bufid
          |   identification      |
          +-----------------------+
  16      |    stream request     |   sh_streq
          |                       |
          +-----------------------+
  16      |   datagram request    |   datareq
          |                       |
          +-----------------------+
  16      | byte length of the    |   datalen
          | following data        |
          +-----------------------+
   8      | HDR word length       |   rellen  (*)
          |                       |
          +-----------------------+
   8      |  new logical number   |   newlog
          |                       |
          +-----------------------+
```

(*) This byte must contain the word length of the satellite header. It is checked to be sure about the reception of just an header (if in the control sub-burst the word length corresponding to this sub-burst is 0xC).

```
              +-----------------------+------------------------+
   16         |  liason label         |h_port    LOCAL NETWOK  |
              |                       |           ROUTING      |
              +-----------------------+          INFORMATION   |
   16         |byte len. of the data  |h_length     (*)        |
              |part of the packet     |                        V
              +-----------------------+
   16         |   check-sum           |h_check
              |                       |
              +-----------------------+
   16         |   packet type         |h_head
              |                       |
              +-----------------------+
```

(*)
The local area network routing information field is strictly dependent by the local are network itself. The one presented is related to the Cambridge Ring local area network.

1.9.2.1 The satellite header subfields

The **control byte** format is the following:

```
+--+--+--+--+--+--+--+--+
|N |C |H |X |R |X |S |D |
+--+--+--+--+--+--+--+--+
+7  6  5  4  3  2  1  0-      bit position
```

where:

X = spare bit.
D = IS_DGREQ bit; datagram request is present (value = 1);
S = IS_STREQ bit; stream request is present (value = 2);
R = IS_RESERVED bit; reserved to indicate a master reference burst (value = 8);
H = IS_HELLO bit; hello control message is present (value = 32);
C = IS_CLN bit; change logical number control message is present (value = 64);
N = IS_NEWBORN bit; new born control message is present (value = 128);
X = spare bit.

The **data information** field format is the following:

```
+--+--+--+--+--+--+--+--+
|D |T |T |T |W |W |W |W |
+--+--+--+--+--+--+--+--+
+7  6  5  4  3  2  1  0-      bit position
```

where:

D = data follow the header;

T = type of traffic (3 bits):
    000 = undefined
    001 = bulk data traffic
    010 = undefined
    011 = undefined
    100 = undefined
    101 = undefined
    110 = interactive traffic
    111 = stream traffic

W = where (in which window subframe) data have been sent (4 bits):
    0001 = unused (Ref.Burst subframe);
    0100 = stream subframe used;
    1000 = datagram subframe used;
    0010 = control subframe used;

The **data fragmenting** byte format is the following:

```
+--+--+--+--+--+--+--+--+
|E |F |F |F |F |F |F |F |
+--+--+--+--+--+--+--+--+
+7  6  5  4  3  2  1  0-    bit position
```

where:

F = fragment number (7 bits);
E = End bit. If on, this is the last fragment of a data buffer.

2. Interrupts and routines organization

The Down process is initialized by means of the RX_INI routine called by the "downuser" task. It has to handle only two types of interrupts: the timer interrupt and the satellite EOB interrupt. In the following, the main routines called at each interrupt event are listed together with the other sub-routines invoked (tree structure):

- Initialization phase:
  RX_INI routine entered

```
A) RX_INI              --------->  ALLOC
                                   INI_FREE_SELQ
                                   INI_REFBURST
                                   INI_CCB
                                   INI_RXCB
                                   INI_RXDMA
                                   rx_tini (assembler routine)
                                   PIA_init
```

- Timer interrupt:
  RX_TE routine entered

```
B) RX_TE               --------->  RX_RECOVERY
                                   SCAN_HELLO_MSG
                                   SRB_PREPARE
                                   DATAG_ASSIGNMENT
                                   SEND_TO_TX  --->pia_bout

B.1) SCAN_HELLO_MSG--------->      DOWN_ENQUEUE
                                   DELINK_REQUEST

B.1) SRB_PREPARE       --------->  FILL_GODOWN_SRB
                                   FILL_NEWBORN_SRB
                                   STREAM_ASSIGNMENT
                                   DATAG_ASSIGNMENT
                                   SEND_TO_TX

B.2) DOWN_ENQUEUE      --------->  DEQUEUE
                                   ENQUEUE

B.2) FILL_NEWBORN_SRB------->      DEQUEUE
                                   ENQUEUE

B.2) DATAG_ASSIGNMENT------->      DELINK_REQUEST
                                   LINK_REQUEST
```

- Satellite EOB interrupt:
  BURST_ANALIZE routine entered

```
C) BURST_ANALIZE       --------->  ENQUEUE_TO_CR
```

```
                                           RB_ANALIZE
                                           ANALIZE_CONTROL
                                           INI_RXDMA
                                           cut_chunks
                                           UB12getel

C.1) RB_ANALIZE       --------->           MY_DALLOC_COMPUTE
                                           SEND_TO_TX
                                           SRB_ANALIZE

C.1) ANALIZE_CONTROL-------->              RX_CLN
                                           DGREQ_RECEIVE
                                           RX_NEWBORN
                                           STREQ_RECEIVE

C.1) ENQUEUE_TO_CR     -------->           SEND_TO_TX
                                           UB16qonfree
                                           UB14qatend
                                           wakering

C.2) SRB_ANALIZE       -------->           MY_SALLOC_COMPUTE
                                           SEND_TO_TX
                                           SRB_NEWBORN_HANDLING
                                           SRB_GODOWN_HANDLING

C.2) DGREQ_RECEIVE     -------->           LINK_REQUEST

C.2) wakering          -------->           ttyin

C.2) RX_NEWBORN        -------->           ENQUEUE
                                           DEQUEUE

C.2) STREQ_RECEIVE     -------->           LINK_REQUEST
                                           DELINK_REQUEST

C.3) SRB_GODOWN_HANDLING--->               SEND_TO_TX
```

3. The Down process starting

To start the down process, the operator is requested to enter the station physical address using the terminal connected to the down box of the satellite bridge.
As soon as the value is entered, it is checked for validity and, if accepted, a PIO message is sent to the Up process to communicate the station physical address and the relative tx slave station offset. This communication between the Down process and the operator is handled by the "downuser" task which activates also the LAN interface handler task "downring". Then the routine RX_INI is entered.
The "downuser" task from now on remains in a wait state for reading from terminal other operator commands.

When the RX_INI routine is entered, the following actions are done:

a1) the area of the free station element queue (INI_FREE_SELQ); is initialized;

a2) the REFBURST area is initialized containing the null reference burst, independently whether or not the station is master;

a3) (where it is possible) the "code" and "length" bytes of the areas normally used by the Down process to pass (via PIA) information to the Up process are initialized;

a4) the CCB and the RXCB areas are initialized, setting in RXCB the "role" value equal to "slave" (default) with logical number (current and previous) equal to -1 and the "temporary status" equal to starting;

a5) the RX DMA is initialized;

a6) the timer for the timer interrupts is initialized;

a7) the PIA interface is initialized;

a8) some global areas are set to zero.

4. The Down process master/slave behaviour

As soon as a timer interrupt occurs the RX_TE routine is entered. If the station is in the starting phase (i.e. the IS_STARTING bit is on in the RXCB temporary status), the following actions are performed:

a) If no reference burst has been received:

- a1) a counter of the consecutive missing reference bursts is increased;
- a2) as soon as the counter reaches a fixed value, the Down process assumes to start as master (nothing is on the air). So, the station logical number is set to zero;
- a3) the counter of the missing reference bursts is cleared;
- a4) the temporary status in RXCB is cleared;
- a5) in the SCB # 0 (relative to the master station) the IS_INITIALIZING bit is set on in the "status" field, and the physical address of the station is copied in the "phyaddr" field;
- a6) the hardware is alerted to start as master;
- a7) a null superframe reference burst is sent for transmission to the Up process.

As soon as the sent null superframe reference burst is received back by the master station (BURST_ANALIZE routine) and recognized as a null reference burst (RB_ANALIZE routine), the following actions are taken:

- in the refburst area, the reference burst is marked as valid;
- the adjustment of the station with respect to the satellite movements is calculated and written in the refburst area (for next sending);
- in the status of the SCB number zero the IS_INITIALIZING bit is set off and the IS_ACTIVE bit is set on to indicate that now the station is really active.

b) If a reference burst has been received:

- b1) some other station is master, so the station must really start as slave (default). In this case, the information regarding the received superframe reference burst (SRB) has already been passed to the Up process (on receiving of SRB, in the SRBINF area) with the indication that the current station logical number is -1. The Up process detects that the logical number must still be assigned by the master to the station and sends the NEWBORN control message.

5. The change logical number behaviour

The "change logical number" procedure is entered when a slave station is declared dead and the station with the current highest logical number must assume that one of the dead station to fill the hole created by the dead station in the logical number sequence.
Here it must be reminded that in a superframe only one station at a time can be declared dead.

- Slave Down process detection that its current logical number must be changed (at the S.R.B. receiving time):

  a) the field "hole" in the S.R.B. is not zero;

  b) its current logical number is equal to the next available logical number -1. It means that the station has the highest logical number.

- Slave Down process actions:

  a) to clear the SCB related to the station having "hole" as logical number (dead station);

  b) to set on the IS_CLNP bit in the status of its SCB (related to its current logical number);

  c) to set the current logical number equal to "hole" in the RXCB;

  d) to inform the Up process to send the Change Logical Number control message. The logical number the station is going to assume (hole) is passed to the Up process;

  e) to pass to the Up process the information related to the received S.R.B. (SRBINF area), writing the "previous" logical number in the "logical number" position.

- Master Down process detection of the presence of the Change Logical Number control message:

  a) in the received satellite header, the IS_CLN bit is on in the "control" field.

- Master Down process actions on receiving of a Change Logical Number control message:

  a) to check whether or not the IS_ACTIVE bit is on in the SCB related to the "new logical number" the slave station is going to assume. If yes, this is a duplicated message and therefore it is discarded;

  b) to copy the SCB relative to the "slog" field (in the satellite header) in the SCB relative to the "newlog" field;

c) to adjust, eventually, the datagram and stream request queues;

d) to clear the SCB related to the "slog" station (the old logical number of the slave station);

e) to clear the "hole" field in the Superframe Reference Burst area;

f) to set the "next available logical number" field equal to "slog" in the Superframe Reference Burst area.

- Slave Down process detection that the Change Logical Number control message has been received by the master station:

  a) the IS_CLNP bit is on in the SCB relative to its "previous" logical number;

  b) the "hole" field in the new received S.R.B. is not equal to the "current" logical number.

- Slave Down process actions:

  a) to copy the "previous" SCB in the "current" SCB;

  b) to clear its "previous" SCB;

  c) to set the "previous" logical number equal to the "current" logical number value in the RXCB;

  d) to set off the CLNP bit in the status of its current SCB;

  e) to pass to the Up process the command to stop sending the Change Logical Number control message.

6. PIO messages from the Down toward the Up process

The following codes are used for the PIO communications from the Down to the Up processs:

-MASTER_SRB : if the station is master, the REFBURST area is passed to the Up process in order to transmit the superframe reference burst. As REFBURST does not contain the datagram allocations, which are contained in the MDALLOC area, two sequential PIO operations are necessary to communicate to the Up process the whole superframe reference burst.
The REFBURST area is passed to the Up process only every superframe (64 frames) just to reduce the activity on PIO.

-MASTER_DALLOC : if the station is master, at every frame the MDALLOC area is passed to the Up process, containing all the datagram allocations for next frame.

-MY_DATAG_ALLOC: when a reference burst is received and datagram allocations are present for the specific station, the RBMYDA area is filled with the datagram assignment starting times and the assignments lengths. Then this area is communicated to the Up process in order to set up the datagram transmission windows.

-SLAVE_SRB : when a superframe reference burst is received, the SRBINF area is filled and passed to the Up process. If no stream allocations are present on 4 frames for the specific station, the RBMYSTA part of SRBINF is filled with zeroes.

-SEND_CLN : the CLNAREA area is passed to the Up process in order to send the "change logical number" control message.

-STOP_CLN : the SCLNAREA area is passed to the Up process in order to stop sending the "change logical number" control message.

-PHY_SITE : the SITEINF area is passed to the Up process in order to communicate to it the station physical address and the (slave) tx station offset.

-DELAY_MEASUREMENT: this special code is used to alert the Up process about the receiving of the Delay Measurement Packet (DMP), a very packet whose use is to measure the delay of the entire system on the base of the traffic conditions of the station (see Appendix A).

-DOWN_FRAME : master or slave, the FRAME_AREA area is passed, once per frame, to the Up process to maintain synchronized the frame numbers of the processes. Command ready but nor used.

All the previous areas are passed from the Down to the Up processs via the PIA interface. The first byte of each area contains the operation code (code) and the second byte (len) the word length of the data to be passed, "code" and "len" not being included in this count.

7. The receiving chain from satellite

On receiving from satellite, data are put by the TDMA controller hardware (of the Down box) in a chain of pre-existing **rx chunks** each one having the following structure:

```
+-------------------------+
|    address of the next  |  c_ptr (32 bits)
|    chunk in the chain   |
+-------------------------+
|                         |  c_data
|      data part of the   |
|      chunk              |  (CHUNK_SIZE = 128
.                         .     16-bits words)
.                         .
+-------------------------+
| word length of the data |  c_len (16 bits)
| contained in this chunk | (used only for transmission
+-------------------------+  of the data on the LAN)
```

It must be noted that the RX hardware considers the receiving chunk 128 words long in total: 2 words for the next chunk address and 126 words (HARDWARE_DATACKSIZE = 126 16-bits words) of data. The "c_len" field is not considered by the receiving DMA hardware; it is filled by the software for the transmission of the data on the LAN.

The data part of a receiving chunk may contain:

```
              a reference burst      (see 1.9.1)
                   or
              an header              (see 1.9.2)
                   or
              a frame counter information
                   or
              a control sub-burst
                   or
              user data.
```

The frame counter information is 4 8-bits bytes long and it is filled by the hardware at each satellite DMA end-of-burst (EOB) interrupt.

The receiving chunks are organized in a queue, called the FREE_CHUNKS queue, so constituted:

```
+-----+-----+-----+-----+
|  H  |  N  |  T  |  D  |
+-----+-----+-----+-----+
  32    16    32    32      bit-wide
```

where:

H = pointer to the first chunk in the queue;
N = number of the chunks in the queue;
T = pointer to the last valid (for writing) chunk in the queue;

D = pointer to the "dummy" chunk. The dummy chunk is the very last chunk in the queue and its pointer to the next chunk points to itself. Normally it has not to be used (it would mind that in the queue there are no more usable chunks). If used, the hardware will rewrite on the same (dummy) chunk the data received from the satellite.

The used chunks must be released by means of the cut_chunks routine which moves the used chunk(s) from the current position to the bottom of the free_chunks queue. So the chunks are used in a circular manner.

At each receiving from satellite, the chunk chain is filled by the hardware:

```
+---------+   +---------+   +---------+   +---------+   +---------+
|   F     |-->|   CS    |-->|  RB/H   |-->|  H      |-->|  D      |..
+---------+   +---------+   +---------+   +---------+   +---------+

-->+---------+   +---------+            +---------+   +---------+
   |  D      |-->|   H     |-->....-->|   H      |-->|    D    |-->..
   +---------+   +---------+            +---------+   +---------+
```

where:

F = frame counter chunk;
CS = control sub-burst chunk;
RB = reference burst chunk. It may or may not be present;
H = satellite header chunk;
D = user data chunk.

In any case, the first chunk after the control sub-burst chunk will always contain a reference burst or a satellite header. After a satellite header chunk, a user data chunk may or may not be present (i.e. if the control slot is used, user data may be not present).

The data part of the control sub-burst chunk has the following format:

```
+-----+-----+-----+-----+-----------+-----+-----+
|     |  L  |  L  |  L  |           |  L  |extra|
| N   |-----|-----|-----|...........|-----|word |........
|     |  C  |  C  |  C  |           |  C  |     |
+-----+-----+-----+-----+-----------+-----+-----+
   0     1     2     3                 N    N+1
```

Each field is 16-bits long, with the following meaning:

N   = number of the received sub-bursts. N is not inclusive of the control sub-burst itself;
L/C = length in words of the sub-burst and its coding. L is constituted by the less significative 12 bits, C by the most significative remaining 4 bits.

The extra word of the control sub-burst has to be checked in order to verify whether or not the control sub-burst itself has been correctly received (it must be zero!). If no, all the filled chunks must be removed from their current position and put at the bottom of the free_chunks queue, together with the frame counter chunk and the control sub-burst chunk themselves.

The minimum length of the control sub-burst is 3 16-bits words. Therefore, the minimum allowable number of transmitted sub-bursts is 2. In this case, the control sub-burst field is so filled:

```
+-----+-----+-----+-----+
|     |  L  |     |extra|
| 2   |-----|  0  |word |.........
|     |  C  |     |     |
+-----+-----+-----+-----+
   0     1     2     3
```

which means that the second sub-burst is a dummy one, with 0 length and coding values.

Dividing the sub-burst word length (L) by the number of words constituting the data part of a chunk really filled in by the hardware (HARDWARE_DATACKSIZE), the number of chunks constituting the sub-burst is computed, adding 1 in case that the rest of the division is not zero. If the receiving is correct, this sub-chain of chunks is maintained in memory in an appropriate queue (on the base of the data type) in order to be enqueued to the LAN later on, when all the data constituting the original buffer will be received.
If any consistency error is detected, the sub-chain of chunks is returned to the bottom of the free_chunks queue (by the cut_chunks routine).

8. Interface between the Down process and the LAN software

When data are received from satellite, buffers are created for the incoming data and, when completed, they are passed to the the local area network handling process ("downring" task).
A buffer is ready to be passed to the LAN when all the possible fragments constituting the buffer are received.

The type of the incoming data (which fill the "chunks") may be: stream, bulk or interactive. So, three different queues are internally maintained by the Down process before to enqueue the completed buffers to the unique queue of the LAN.

When a buffer is ready to be enqueued, the data structure is the following:

```
+---------+    +-----------+
| Buffer  |-->|           |
| element |   |           |  +--------+
+---------+   |  Buffer   |-->| Header |
              |  Proper   |  | chunk  |
              |           |  +--------+
              |           |
              |           |  +--------+    +--------+      +--------+
              |           |-->|  data  |-->| data   |..-->|  data  |
              |           |  |  chunk |   | chunk  |      |  chunk |
              +-----------+  +--------+   +--------+      +--------+
```

and the buffer element address is used to pass the whole structure to the LAN.

The header chunk linked to the buffer proper must be filled in keeping the data from the first header transmitted related to the buffer. This header must contain the fields:

h_port
h_length
h_check
h_head

relative to the data originating LAN.

The task handling the LAN must be alerted about the data incoming by writing a byte into its pipe (inter-task communication as used in the C-EXEC operating system). It is done by means of the "wakering" routine.

8.1 The buffer element format

```
 bit-wide
             +-----------------------+
     32      |  forward Pointer      |  elfwd
             |                       |
             +-----------------------+
     32      |  backward pointer     |  elback
             |                       |
             +-----------------------+
     16      |  status/port          |  elstat
             |                       |
             +-----------------------+
     16      |  dest. site or host   |  eldest
             |                       |
             +-----------------------+
     32      | pointer to the buffer|  bufstart
             |     proper            |
             +-----------------------+
     16      | no. of timer counts   |  eltimect
             | since retry           |
             +-----------------------+
     16      | no. of retries        |  elsendct
             | sending to ring       |
             +-----------------------+
```

8.2 The buffer proper format

bit-wide		
32	pointer to the first data byte in a chunk	pkt
16	number of bytes of data	bytecount
16	first minipacket	head
16	routing	route
16	type of data	type
16	destination host	desthost
16	destination port	destport
32	pointer to the buffer element	buffel
16	checksum value passed along	checksum
16	whether checksum was wrong	checked
32	pointer to the header chunk	b_head
32	pointer to the first data chunk	b_dat
16	count of the number of chunks	chunk_count
32	pointer to the first out chunk	b_posn
16	encoding rate for data	b_code

The buffer proper format is the same for the Down and the Up processes. Only the chunk structure is different, due to different hardware requirements.

When a buffer is get, the buffer element address is returned. It points to the buffer proper which automatically includes a chunk for the header.

9. The Timer Interrupts Handling

Every frame a timer interrupt occurs in the Down process.
In normal running (i.e. when the station status is ACTIVE), at every timer interrupt the Down process checks whether or not a reference burst has been received. If no, a counter of the consecutive missed reference bursts is increased and the recovery routine for master fault is entered as soon as a certain value of the counter is reached.

On the other hand, every time a reference burst is received, this counter is cleared and the indication that a reference burst has been received is set on.

If the station is master and its status is not yet ACTIVE but only INITIALIZING, the receiving of the null reference burst is waited for. In this meantime, the timer interrupts do not provoke any action, a part the eventual increasing of the counter of the missed reference bursts.

As soon as the null reference burst is received by the master station (the slave station simply throws away the null reference burst), the status of the station changes from INITIALIZING to ACTIVE and the valid superframe reference burst is prepared in the REFBURST area. Its frame number is always equal to zero and in contains the adjustment of the master station with respect to the satellite movements.

If the station is master, the superframe reference burst is prepared and passed to the Up process every 64 timer interrupts. Every superframe, the scan of the received hello messages is also executed.

On the other hand, at each timer interrupt (each frame), the datagram allocations for the next frame are computed and passed for transmission to the Up process.

10. The scanning of the hello messages

At the beginning of every superframe the master station scans the hello messages received from those slave stations declared "active" at that moment.
It happens exactly after 63 timer interrupts, when the superframe reference burst has been already prepared and passed via PIO to the Up process. The hello counter field (helloc) of every SCB is incremented by the master process every time something (datagram request, stream request, hello message itself, etc) is received from the associated slave station.

If, scanning all the SCBs in SCBTAB, a station is declared active (the IS_ACTIVE bit is on) but its associated hello counter is zero, the station is assumed to be dead in the meantime. The master Down process dequeues all the eventual stream and/or datagram requests of the dead station from SRQ and DSQ. The status of the dead station in the associated SCB is set to Down and a STATEL, containing the dead station physical address and logical number is enqueued in the DSQ.

On the other hand, if a station is active and its hello counter is greater than zero, the station is still considered active for the incoming superframe, but its hello message counter is reset to zero in order to verify whether or not something will be received from that slave station in the superframe is going to begin.

If the master station is the only active station, only reference bursts are in the aer. Therefore, at each receiving of a reference burst, the hello message counter of the SCB associated to the master is increased.

When a new slave station becomes active, the master makes a present of a 1 in the hello message counter of the SCB related to the new born station. This is to avoid that the new slave station be immediately declared dead when the hello message counters are scanned if the slave come up too close to the end of the previous superframe (and it had no time to send anything).

11. The RX files organization in the C language implementation

The routines implementing the FODA-TDMA satellite access scheme on the new satellite bridge Down process are organized in files in the following way:

RXDEFINE.H	header file; it defines variables used by the Down process only.
RXSTRUCT.H	header files; it defines the structures of the areas used by the Down process.
RXSTRUCT.X	it contains the external references.
RXALLOC.C	it contains the real allocations of the Down process areas.
RXINI.C	it contains all the initialization routines:

RX_INI	main initialization program;
INI_FREE_SELQ	to initialize the FSQ queue;
INI_REFBURST	to initialize the refburst area;
INI_CCB	to initialize the CCB area;
INI_RXCB	to initialize the RXCB area;
INI_RXDMA	to initialize the RX DMA interface.

RXPIO.C	it contains the SEND_TO_TX routine for the PIO communications with the Up process.
RXTIMINT.C	it contains all the routines invoked when a timer interrupt occurs. They are:

RX_TE	entered when a timer interrupts occurs;
SCAN_HELLO_MSG	to scan the received hello control messages;
SRB_PREPARE	to prepare the superframe reference burst;
FILL_GODOWN_SRB	to fill the "godown" field in the superframe reference burst in course of preparation;
FILL_NEWBORN_SRB	to fill the "newborn" field in the superframe reference burst in course of preparation;
STREAM_ASSIGNMENT	to compute the stream assignments for the entire superframe;
DATAG_ASSIGNMENT	to compute the datagram assignments which will be

	sent with the next superframe ref. burst;
RX_RECOVERY	entered when the master station faults down.

RXBURST.C — it contains the routines invoked when an End-Of-Burst (EOB) interrupt occurs:

BURST_ANALIZE	entered at the DMA EOB interrupt to analize the received burst;
ENQUEUE_TO_CR	to enqueue the data received from satellite to the LAN (in this implementation a Cambridge Ring).
GO_IN_TRAP	to force a system trap.

RXRBANAL.C — it contains all the routines invoked when a reference burst has been received (after an EOB interrupt occurred):

RB_ANALIZE	to analize the received reference burst;
SRB_ANALIZE	to analize the received superframe reference burst;
SRB_NEWBORN_HANDLING	to handle the "newborn" field in the received superframe reference burst;
SRB_GODOWN_HANDLING	to handle the "godown" field in the received superframe reference burst;
MY_DALLOC_COMPUTE	to compute my datagram allocations on the base of the datagram allocations in the received reference burst;
MY_SALLOC_COMPUTE	to compute my stream allocations for the entire superframe.

RXCONTROL.C — it contains all the routines invoked when a satellite header has been received (after an EOB interrupt occurred):

ANALIZE_CONTROL	to analize the control byte in the received header;
RX_NEWBORN	to handle the received newborn control message;
RX_CLN	to handle the received change logical number control message;
STREQ_RECEIVE	to handle the received stream request;

DGREQ_RECEIVE	to handle the received datagram request.

RXQUEUE.C — it contains the routines for manipulating the datagram request queue (DRQ), the stream request queue (SRQ), the new born queue (NBQ) and the down station queue (DSQ):

LINK_REQUEST	to enqueue a request in the DRQ/SRQ queue;
DELINK_REQUEST	to dequeue a request from the DRQ/SRQ queue;
DOWN_ENQUEUE	to enqueue a station element in the DSQ;
DEQUEUE	to dequeue elements from the FSQ;
ENQUEUE	to enqueue elements to the FSQ.

RXCONFIG.C — it contains the device configuration table and the user task definition table for the down process.

RXMSGH.C — the "msghandler" message handler task for the down process. Messages sent at interrupt level are displayed on terminal and/or recorded in a special file.

RXASS.S — some assembler routines are here defined.

RXIN68K.S — the assembler file linking the down FODA software to the C-EXEC nucleus.

UBD70.C — files containing the "downring" LAN interface task.

UBD100.C — the "downuser" task, handling the interface between the FODA system and the operator.

COMMON FILES

The following files contain definitions or allocations valid for both the Down and the Up processes and for the software of the local area network linked to the satellite TDMA controller:

UBC*.H — header files; common definitions of variables and structures are here defined.

RXTX.H — header file; it defines variables and structures used by both the Down and the Up processs.

RXTXALLOC.C — it contains the routine ALLOC used to allocate in memory common variables.

RXTXEXT.X	it contains common external references.
UB10/30/60.C	LAN interface common files.
UB200.C	queues and chunks handling routines.
UB210.C	PIA initialization.
XC10A68K.C	routines relative to the LAN handler.
XC11A68K.C	routines relative to the LAN handler.
XC13DUMMY.C	routines relative to the LAN handler.
UBA1.S	PIO handling assembler routines are here defined: pia_bin to read from PIO, pia_bout to write on PIO.

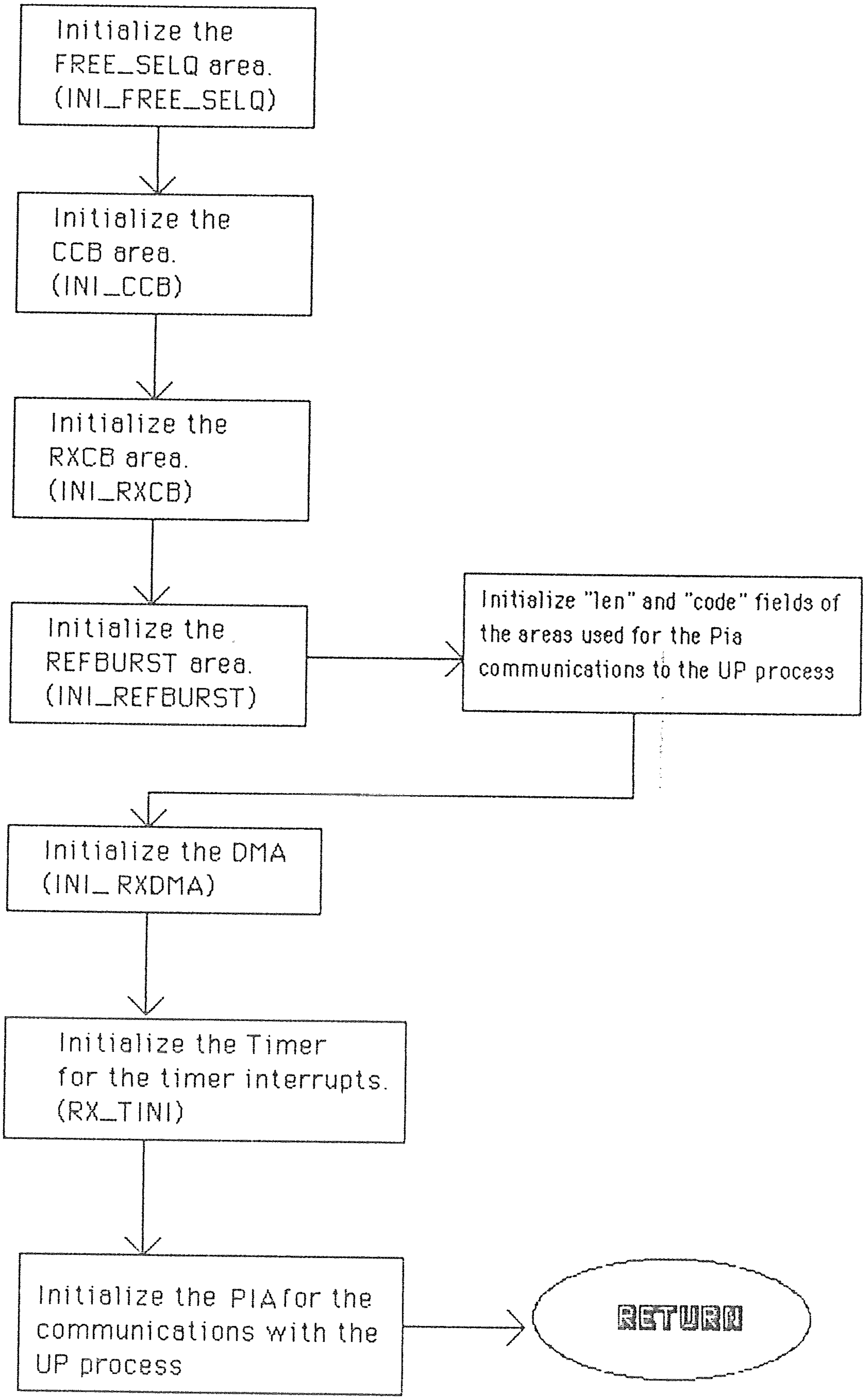
12. The interrupt levels in the DOWN process

```
DMA End Of Burst --------> level 6
TIMER -------------------> level 5
RB UW detection ---------> level 4
PIO read ----------------> level 3
TERMINAL input ----------> level 2
LAN read ----------------> level 1
```

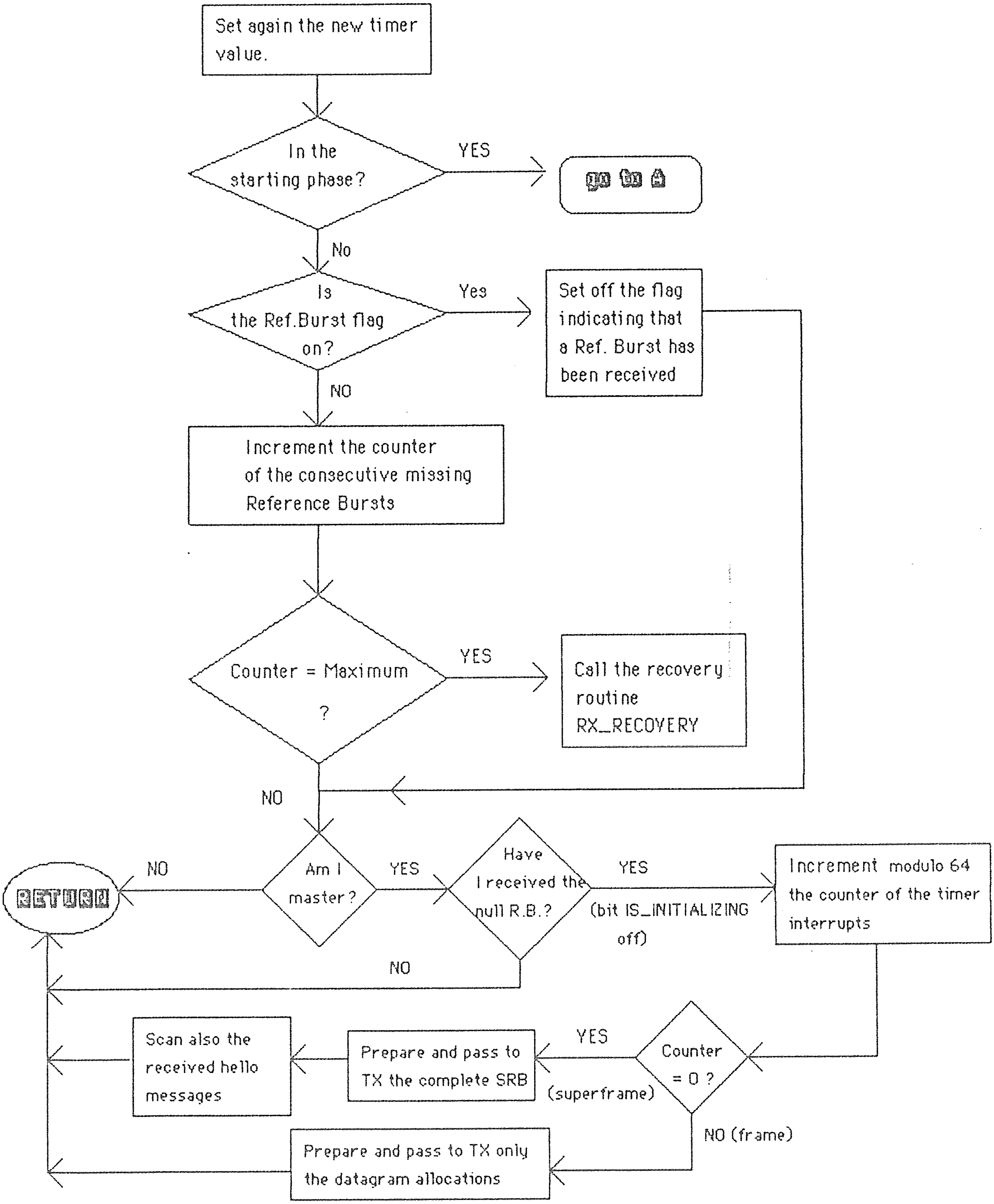
The RX Processor

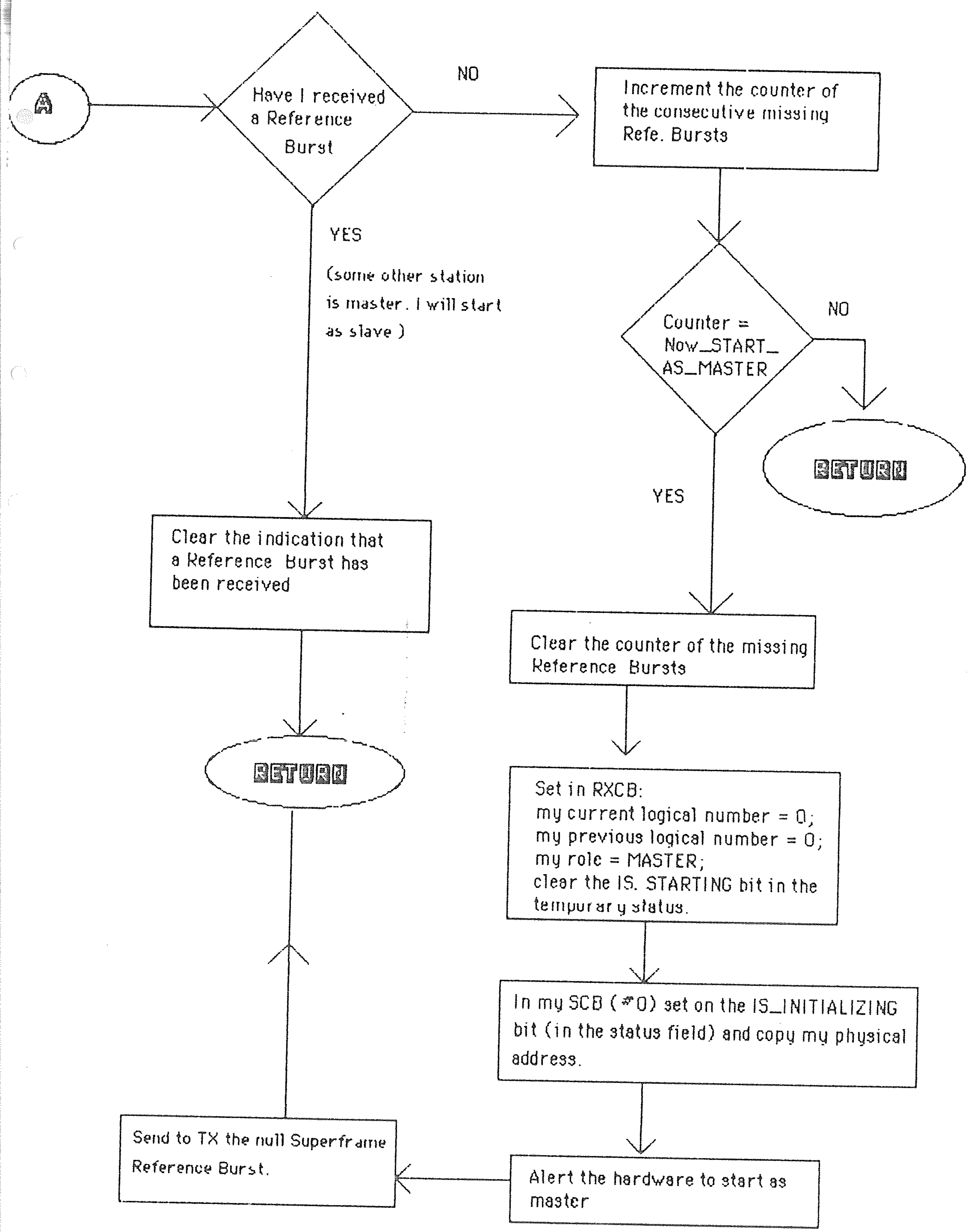
Flow Charts

DOWN Process Initialization: RX_INI routine

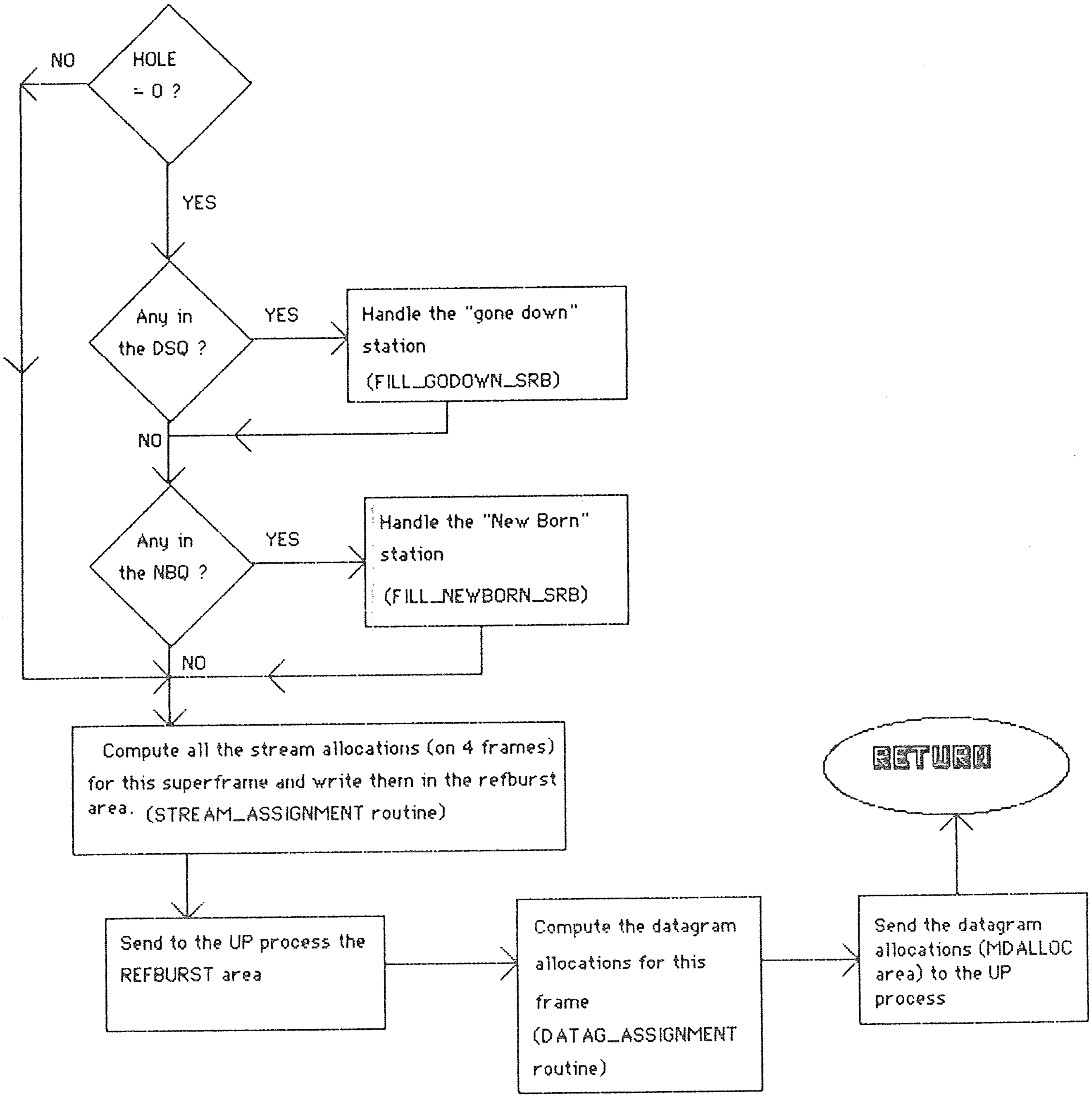


Timer Interrupt: RX_TE routine

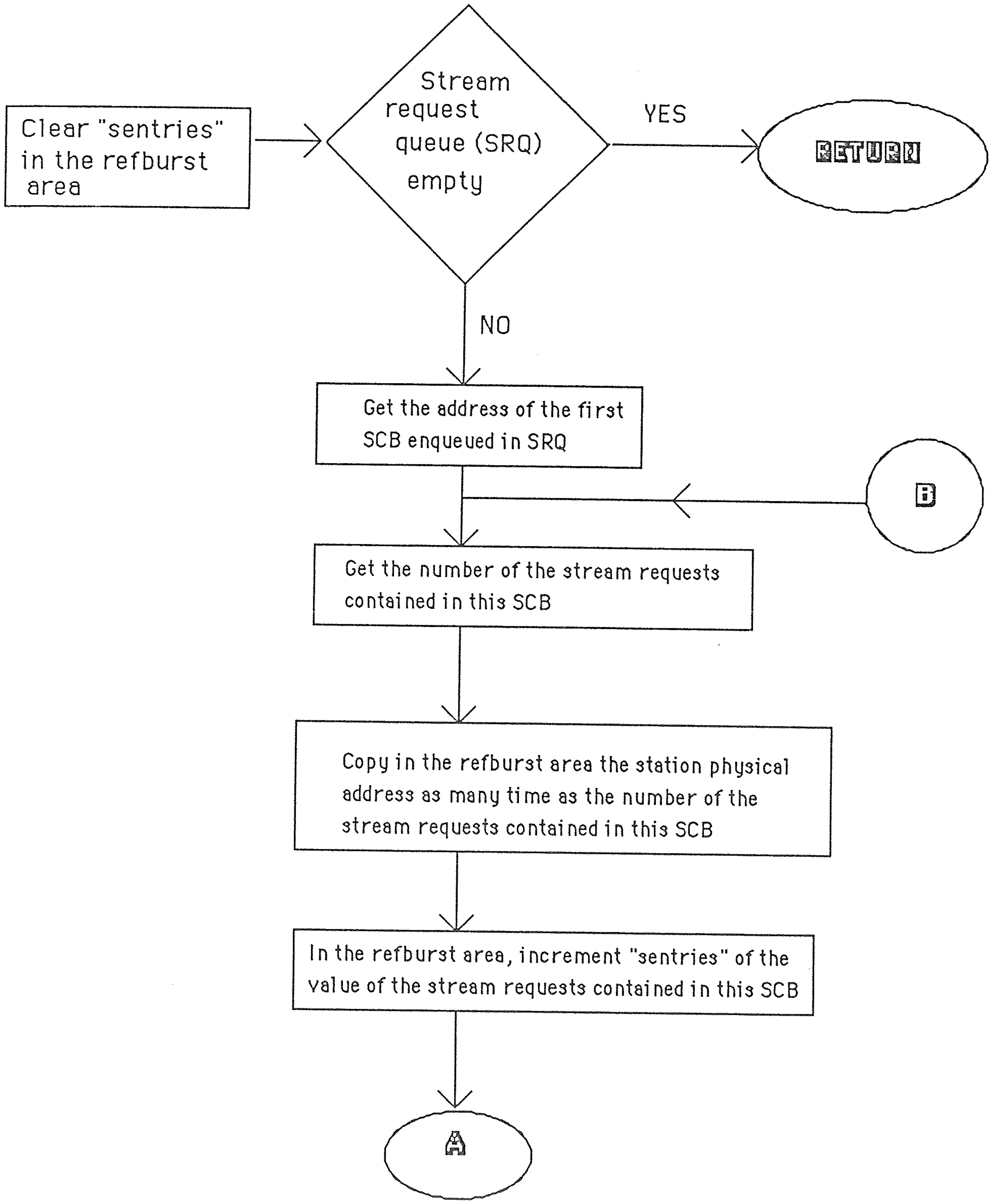


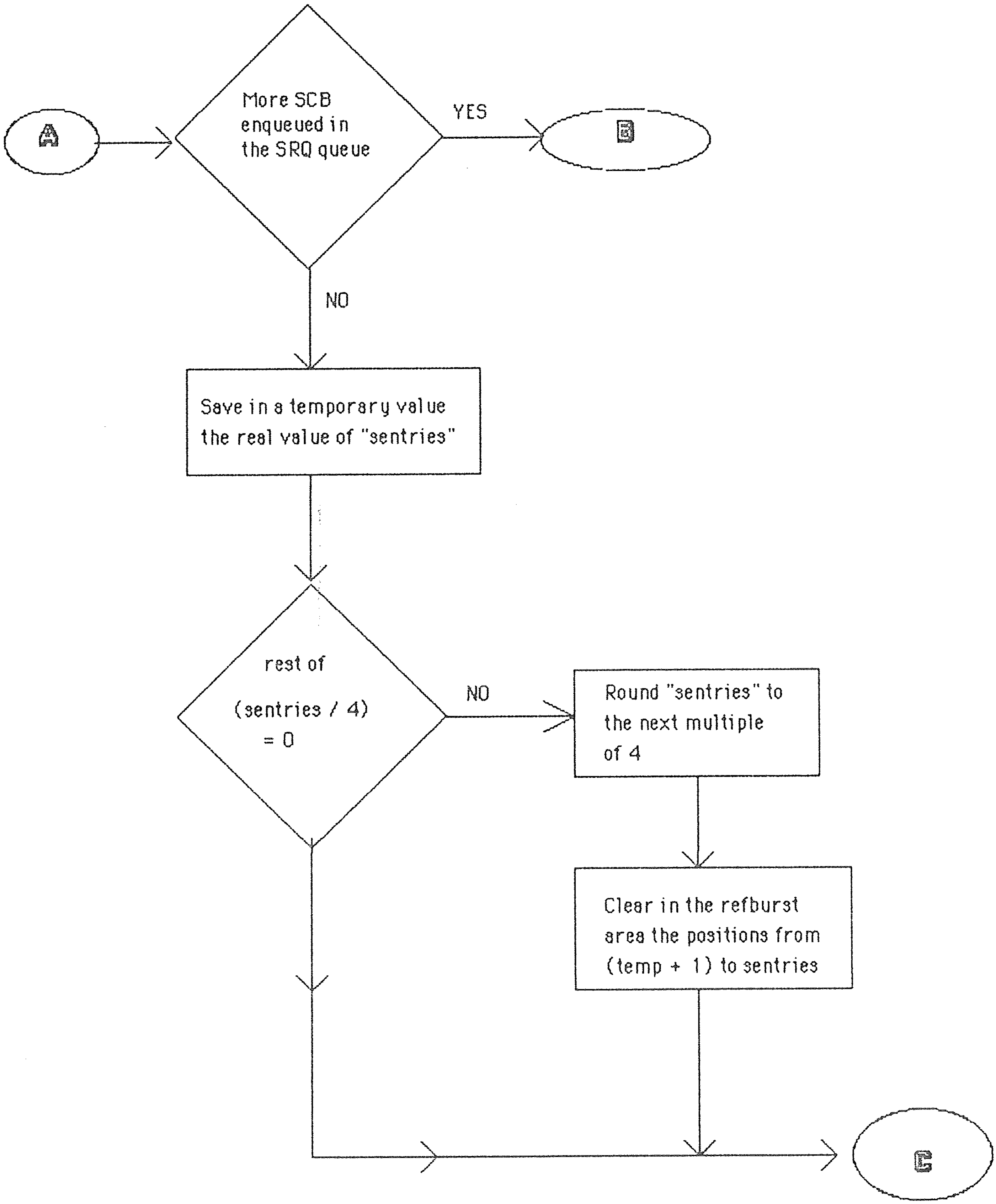

A
Have I received a Reference Burst
NO
Increment the counter of the consecutive missing Refe. Bursts
YES
(some other station is master. I will start as slave)
Counter = Now_START_AS_MASTER
NO
RETURN
YES
Clear the indication that a Reference Burst has been received
Clear the counter of the missing Reference Bursts
RETURN
Set in RXCB:
my current logical number = 0;
my previous logical number = 0;
my role = MASTER;
clear the IS. STARTING bit in the temporary status.
In my SCB (#0) set on the IS_INITIALIZING bit (in the status field) and copy my physical address.
Send to TX the null Superframe Reference Burst.
Alert the hardware to start as master

Superframe Reference Burst preparation: SRB_PREPARE routine

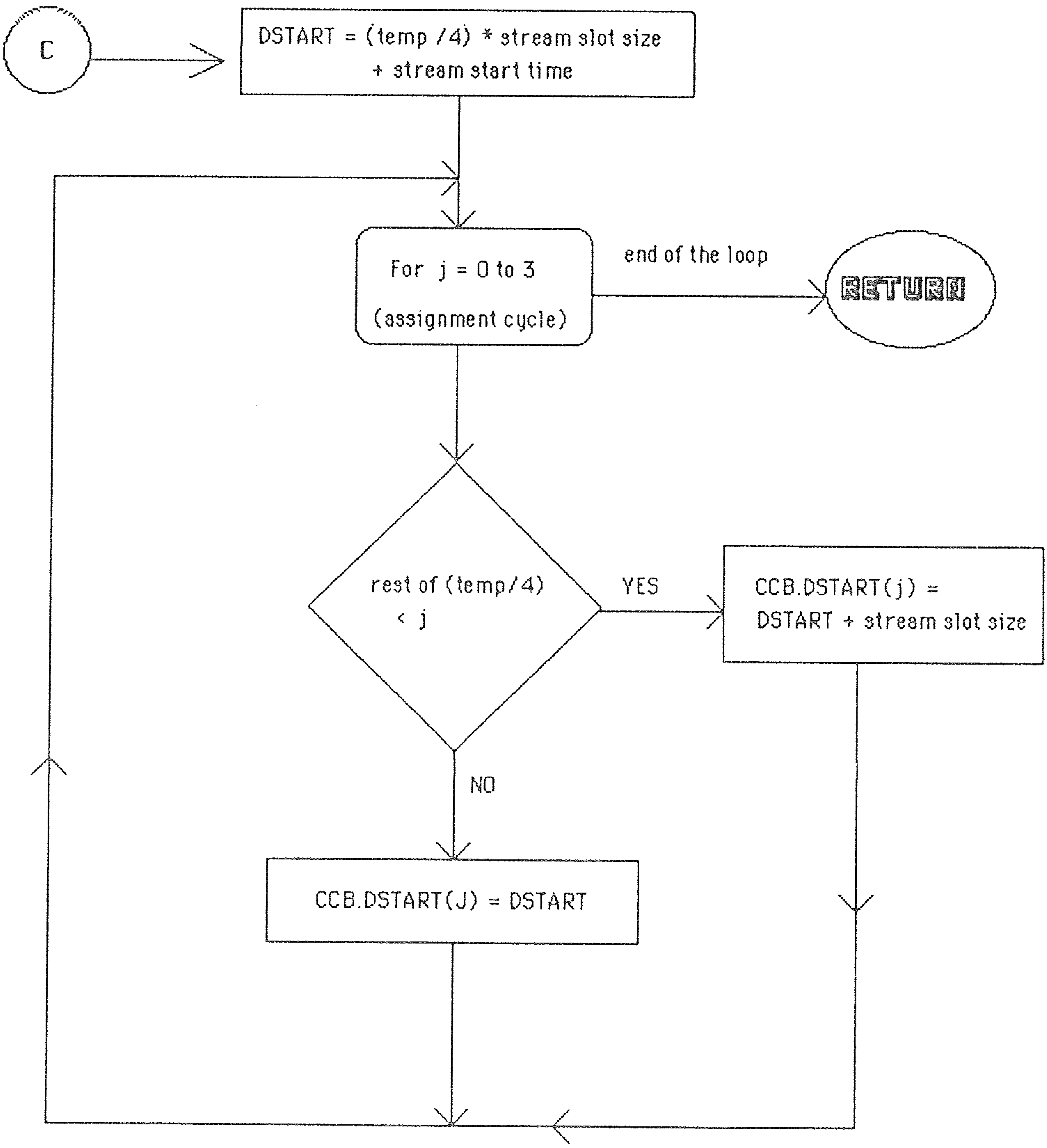


STREAM assignment algorithm: STREAM_ASSIGNMENT routine

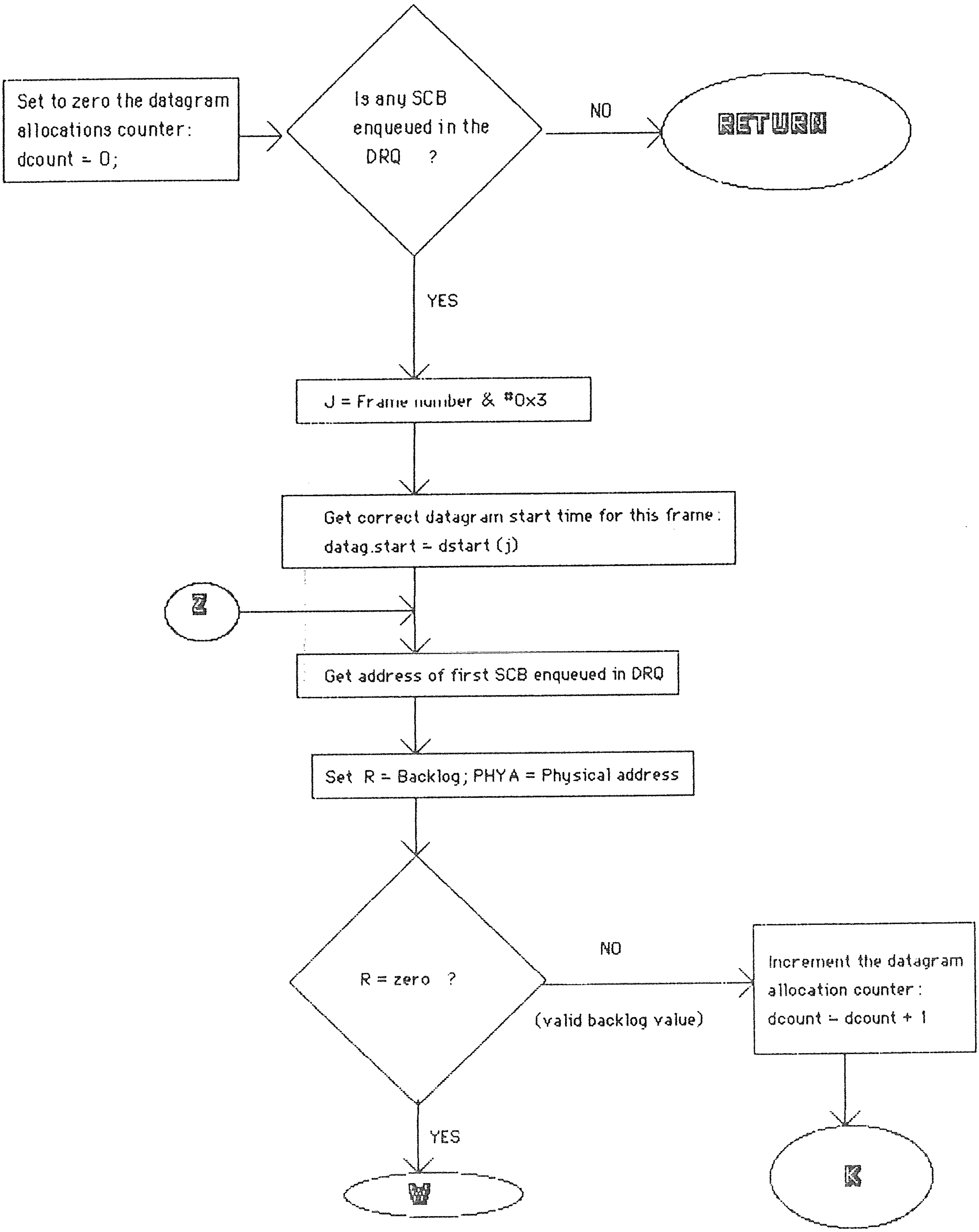


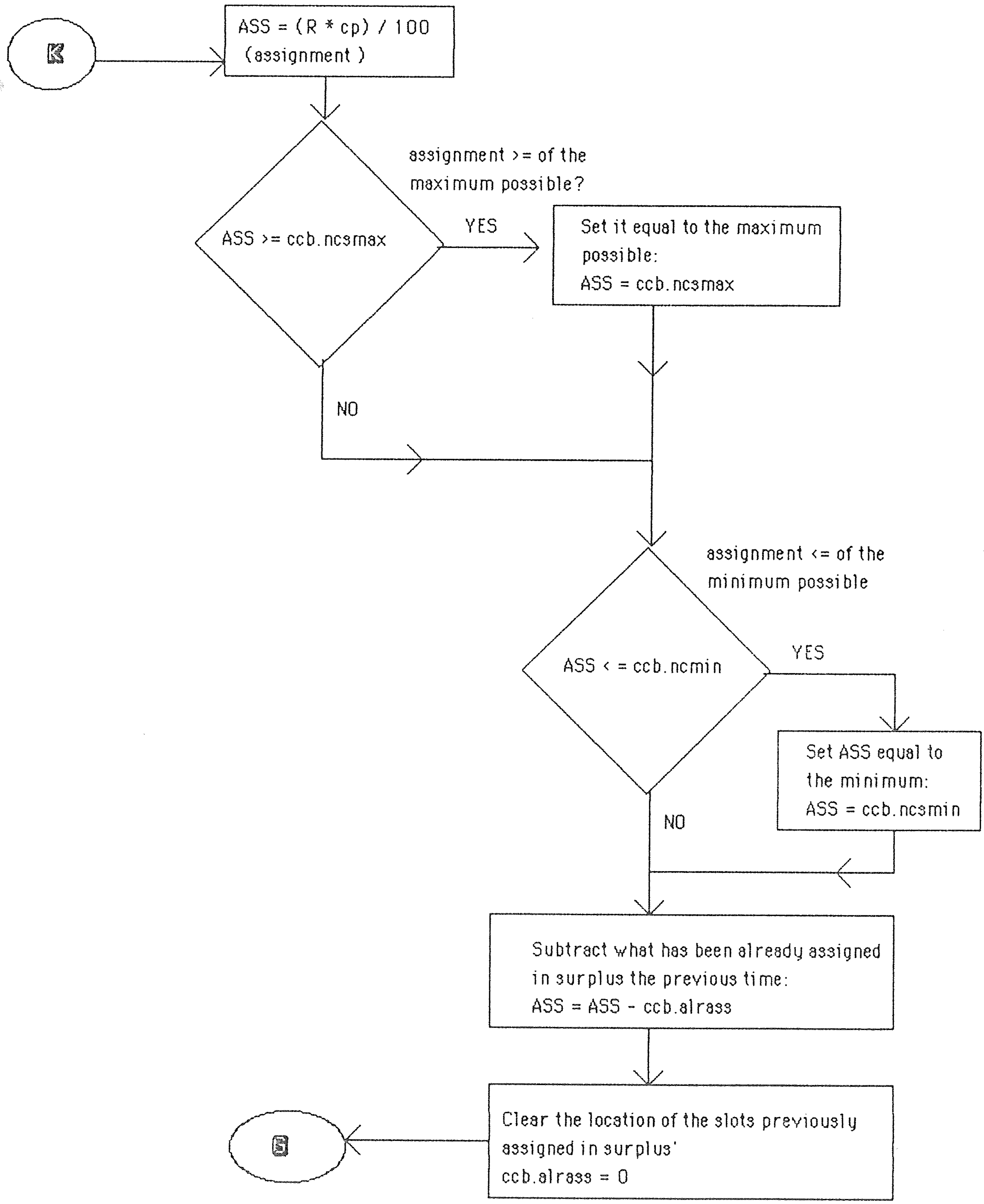


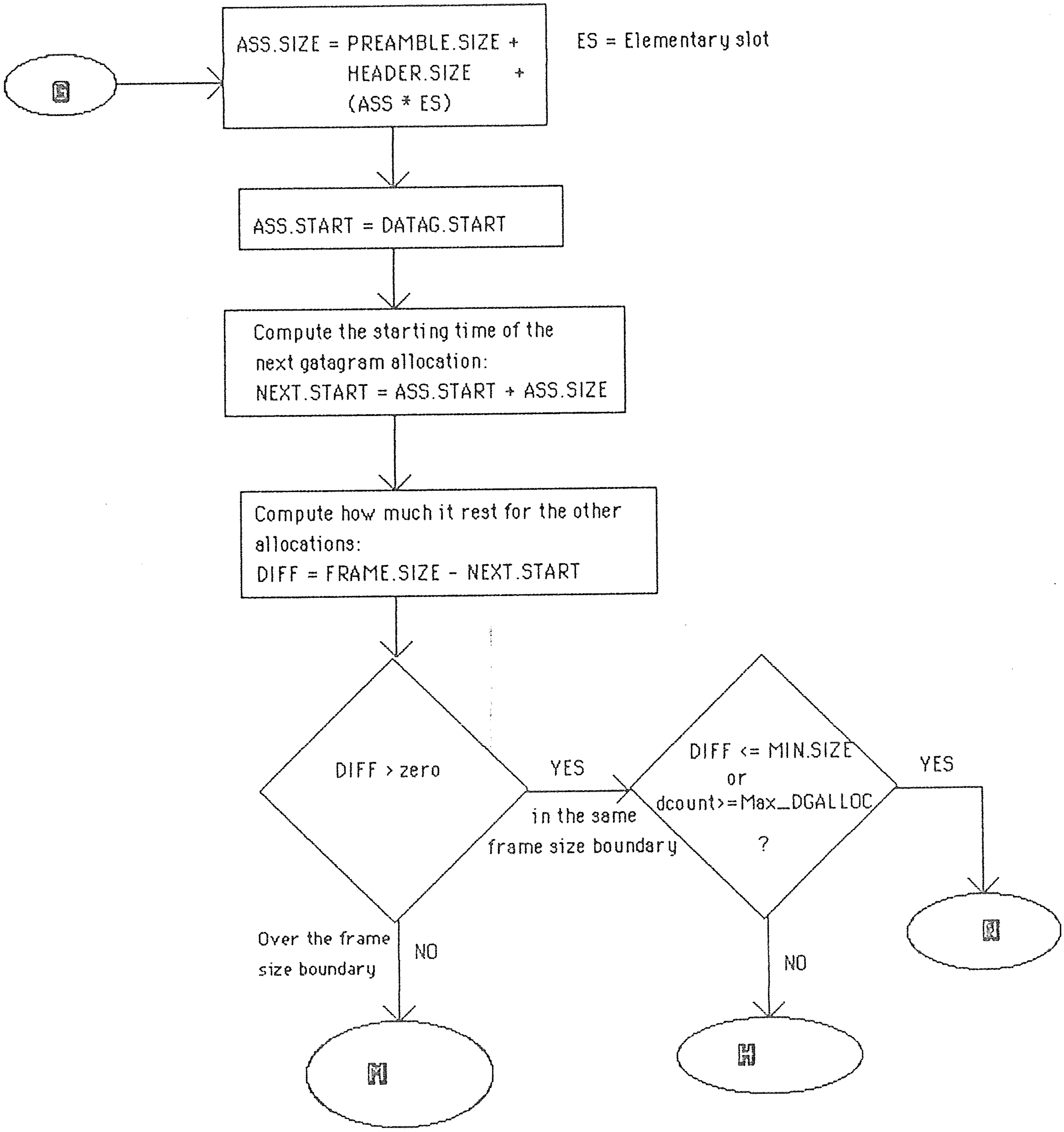
A
More SCB
enqueued in
the SRQ queue
YES
B
NO
Save in a temporary value
the real value of "sentries"
rest of
(sentries / 4)
= 0
NO
Round "sentries" to
the next multiple
of 4
Clear in the refburst
area the positions from
(temp + 1) to sentries
C


C
DSTART = (temp /4) * stream slot size
+ stream start time
For j = 0 to 3
(assignment cycle)
end of the loop
RETURN
rest of (temp/4)
< j
YES
CCB.DSTART(j) =
DSTART + stream slot size
NO
CCB.DSTART(J) = DSTART

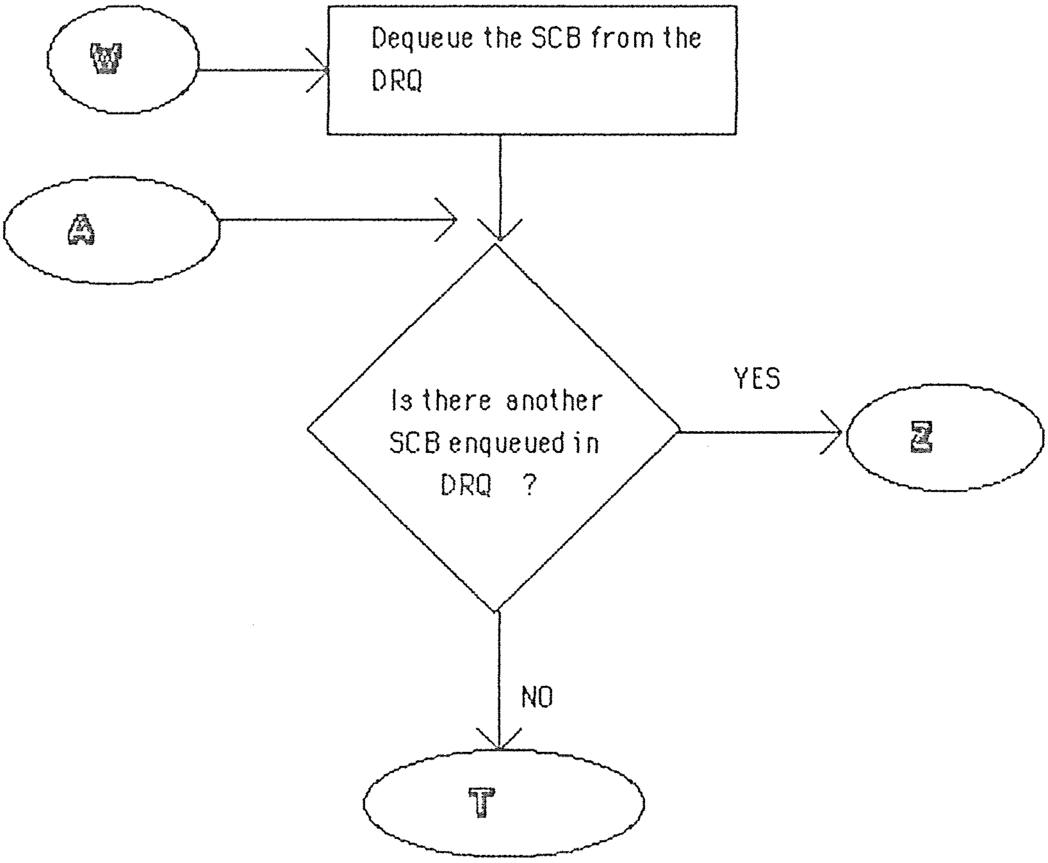
Datagram Assignment Algotithm: **DATAG_ASSIGNMENT routine**

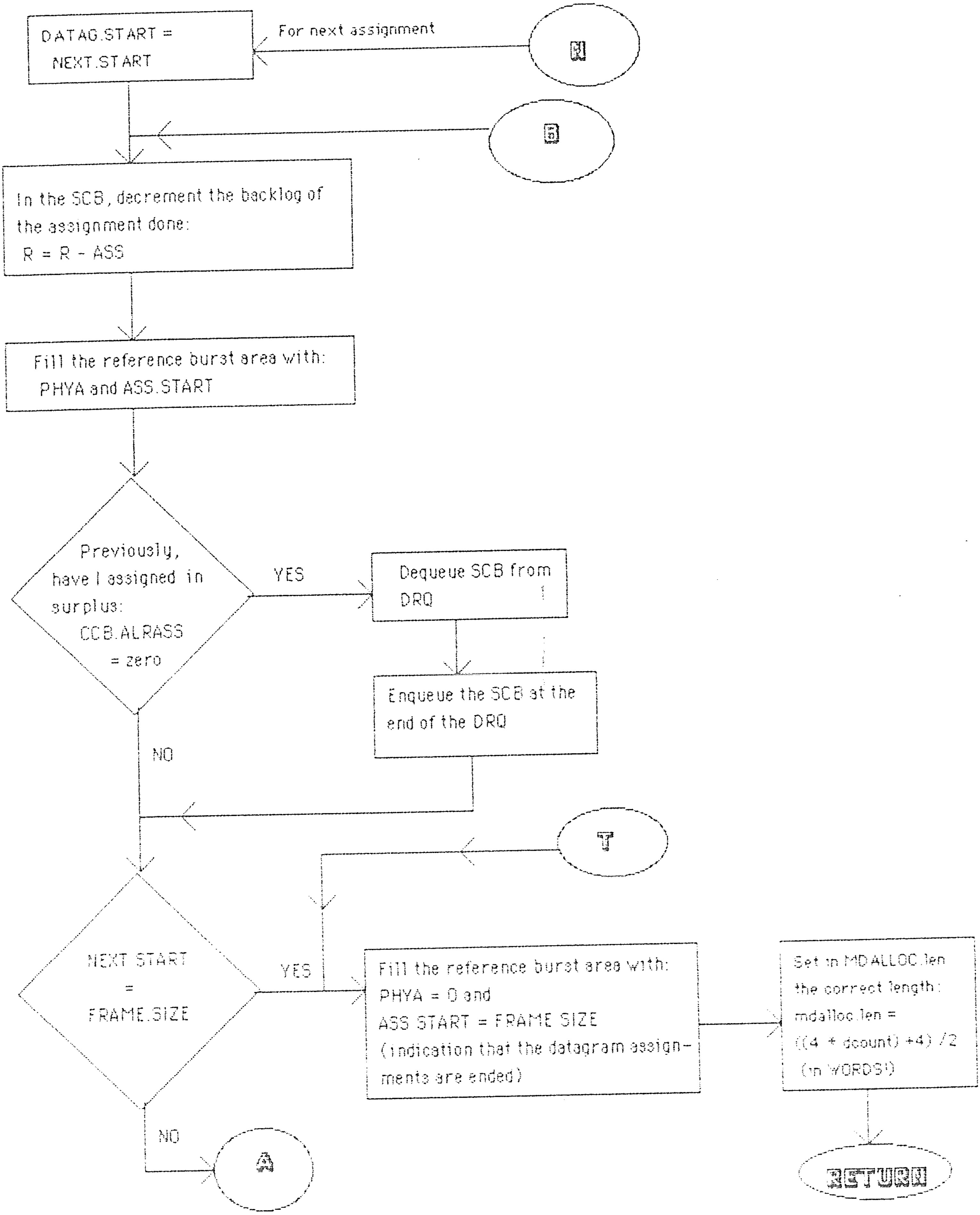


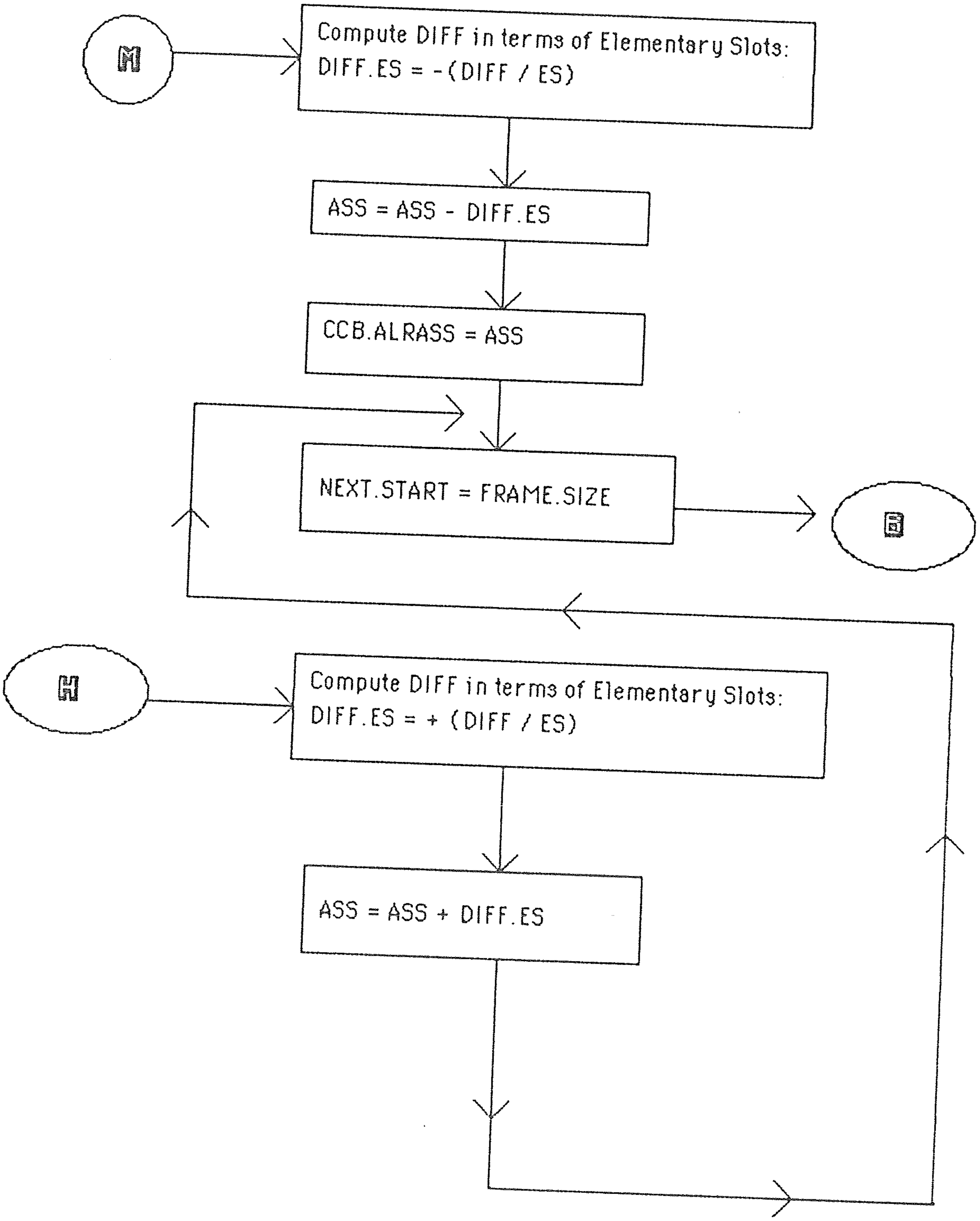

K
ASS = (R * cp) / 100
(assignment)
assignment >= of the
maximum possible?
ASS >= ccb.ncsmax
YES
Set it equal to the maximum
possible:
ASS = ccb.ncsmax
NO
assignment <= of the
minimum possible
ASS < = ccb.ncmin
YES
Set ASS equal to
the minimum:
ASS = ccb.ncsmin
NO
Subtract what has been already assigned
in surplus the previous time:
ASS = ASS - ccb.alrass
Clear the location of the slots previously
assigned in surplus'
ccb.alrass = 0
S



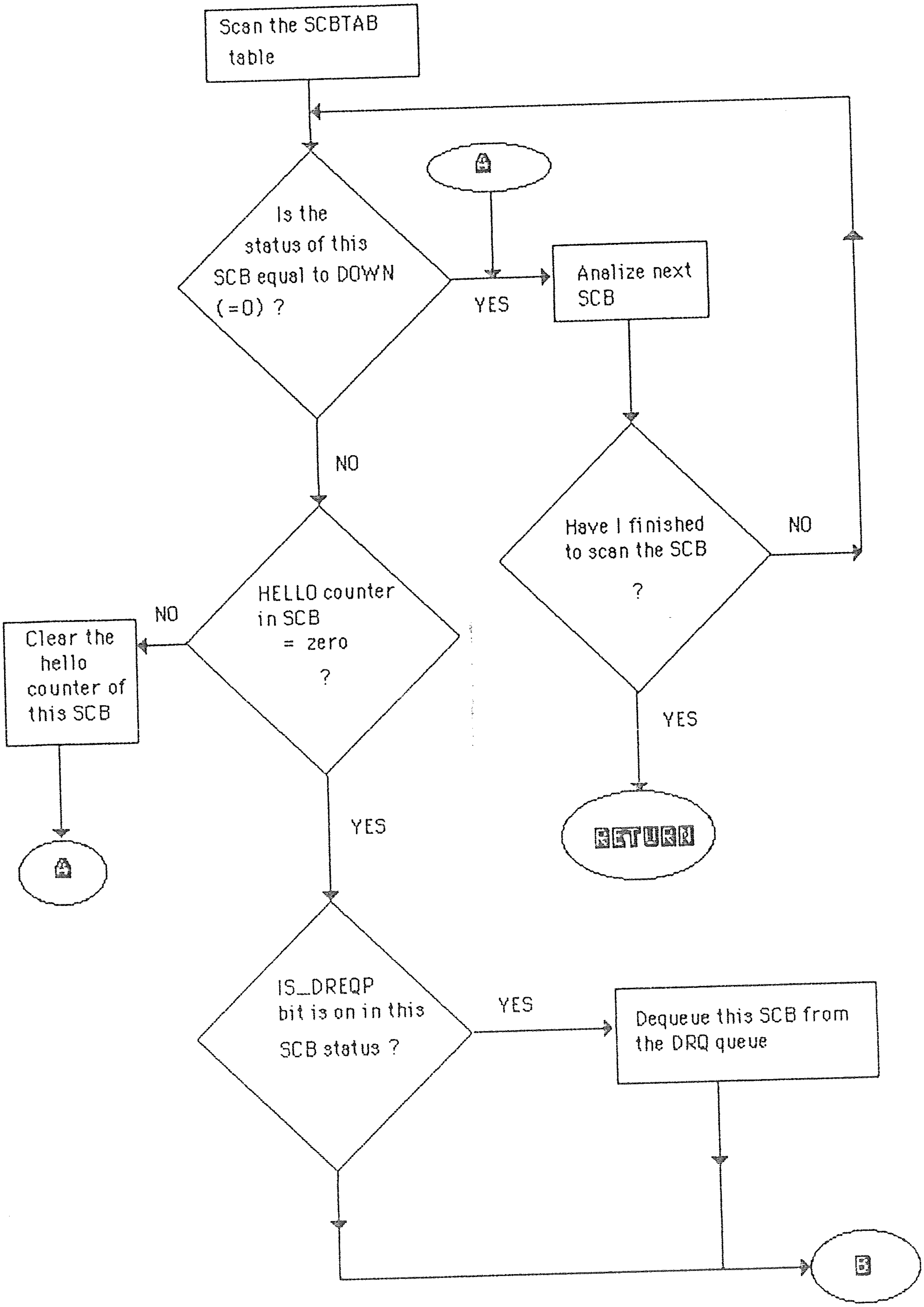
ES = Elementary slot
ASS.SIZE = PREAMBLE.SIZE + HEADER.SIZE + (ASS * ES)
ASS.START = DATAG.START
Compute the starting time of the next gatagram allocation:
NEXT.START = ASS.START + ASS.SIZE
Compute how much it rest for the other allocations:
DIFF = FRAME.SIZE - NEXT.START
DIFF > zero
YES
in the same frame size boundary
DIFF <= MIN.SIZE or dcount>=Max_DGALLOC ?
YES
Over the frame size boundary
NO
NO

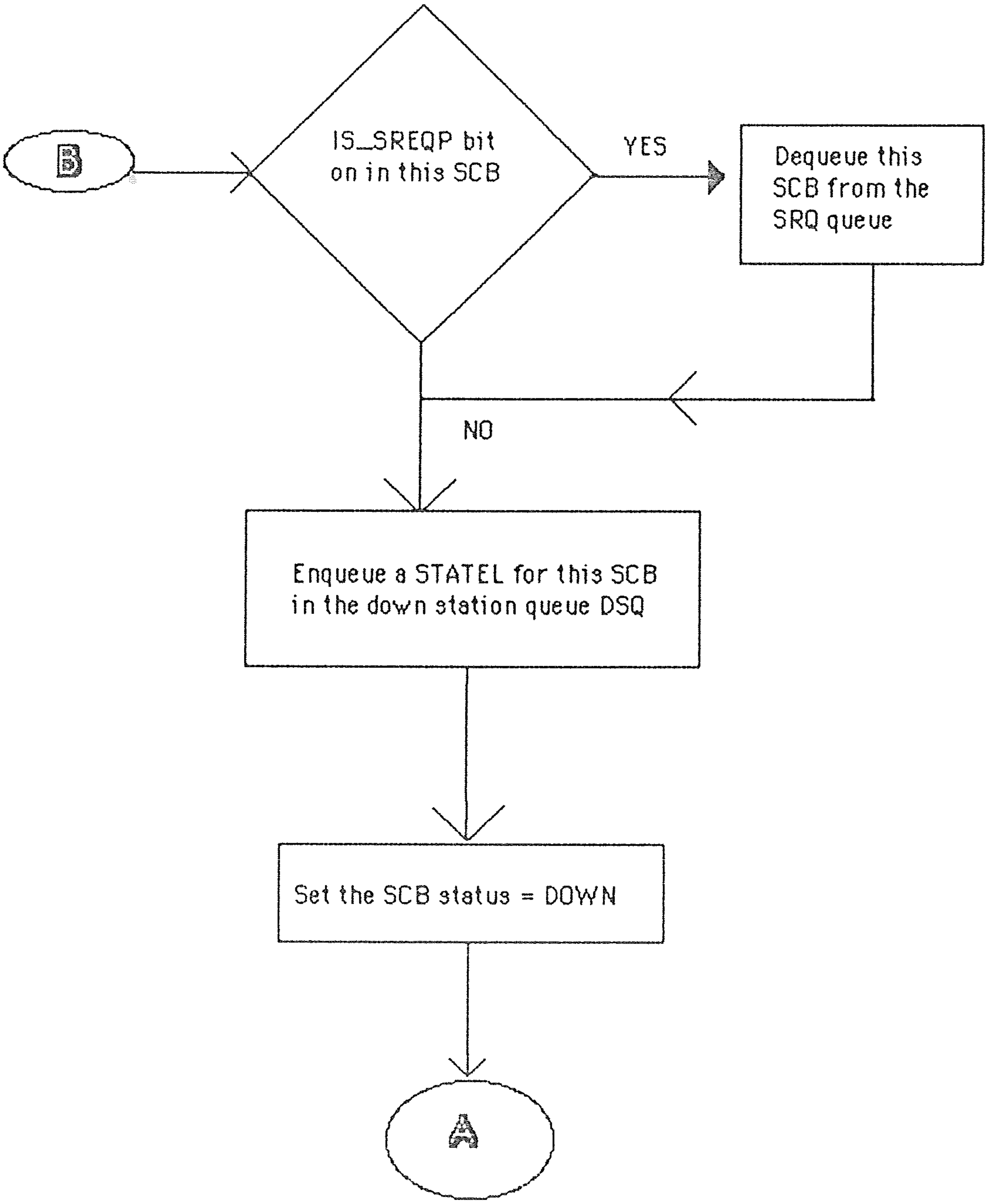

W
Dequeue the SCB from the DRQ
A
Is there another SCB enqueued in DRQ ?
YES
Z
NO
T


DATAG.START =
NEXT.START
For next assignment
R
B
In the SCB, decrement the backlog of
the assignment done:
R = R - ASS
Fill the reference burst area with:
PHYA and ASS.START
Previously,
have I assigned in
surplus:
CCB.ALRASS
= zero
YES
Dequeue SCB from
DRQ
Enqueue the SCB at the
end of the DRQ
NO
T
NEXT START
=
FRAME.SIZE
YES
Fill the reference burst area with:
PHYA = 0 and
ASS START = FRAME SIZE
(indication that the datagram assign-
ments are ended)
Set in MDALLOC.len
the correct length:
mdalloc.len =
((4 + dcount) +4) /2
(in WORDS!)
NO
A
RETURN

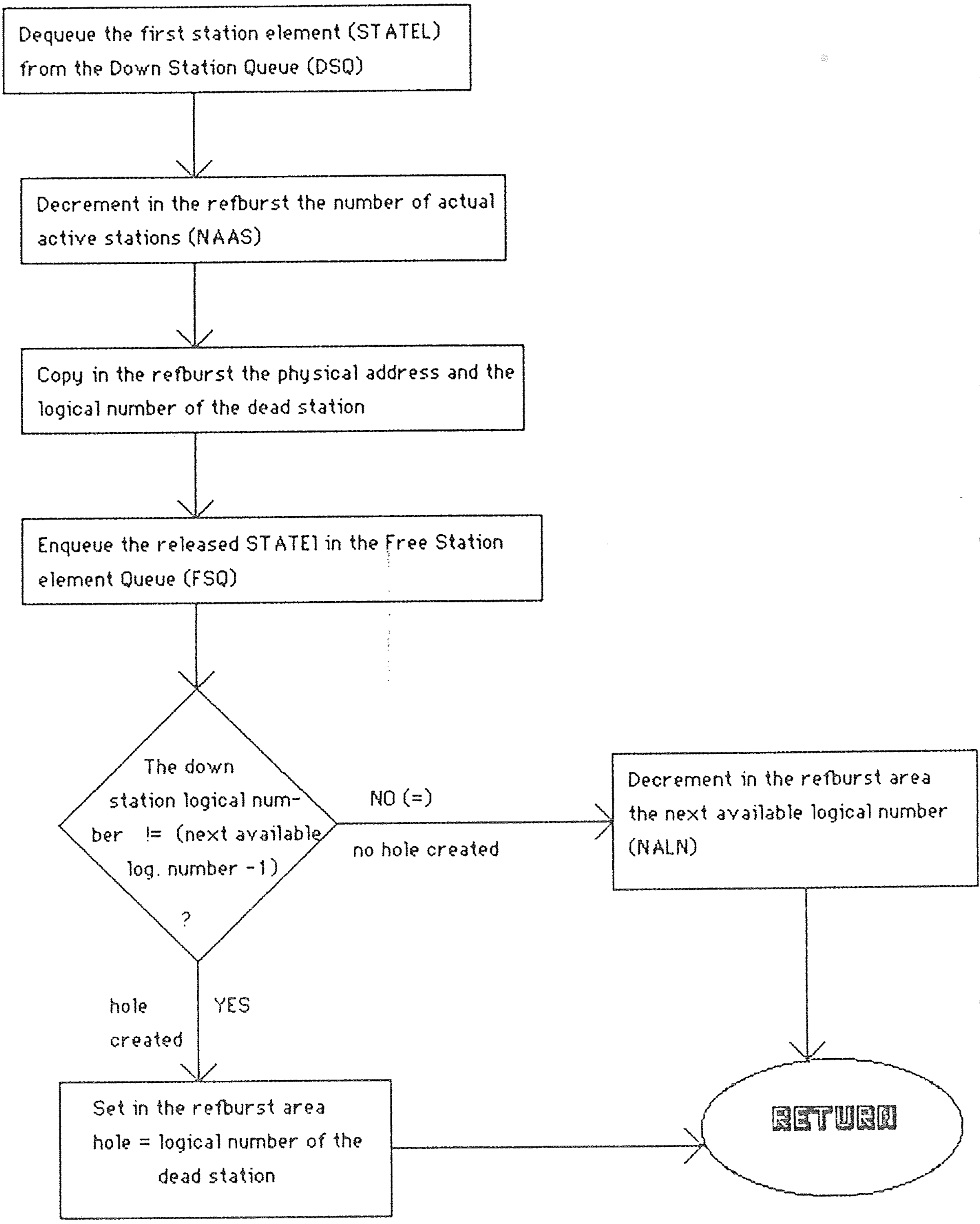

M
Compute DIFF in terms of Elementary Slots:
DIFF.ES = -(DIFF / ES)
ASS = ASS - DIFF.ES
CCB.ALRASS = ASS
NEXT.START = FRAME.SIZE
B
H
Compute DIFF in terms of Elementary Slots:
DIFF.ES = + (DIFF / ES)
ASS = ASS + DIFF.ES

Master scanning of the HELLO msg: SCAN_HELLO_MSG routine

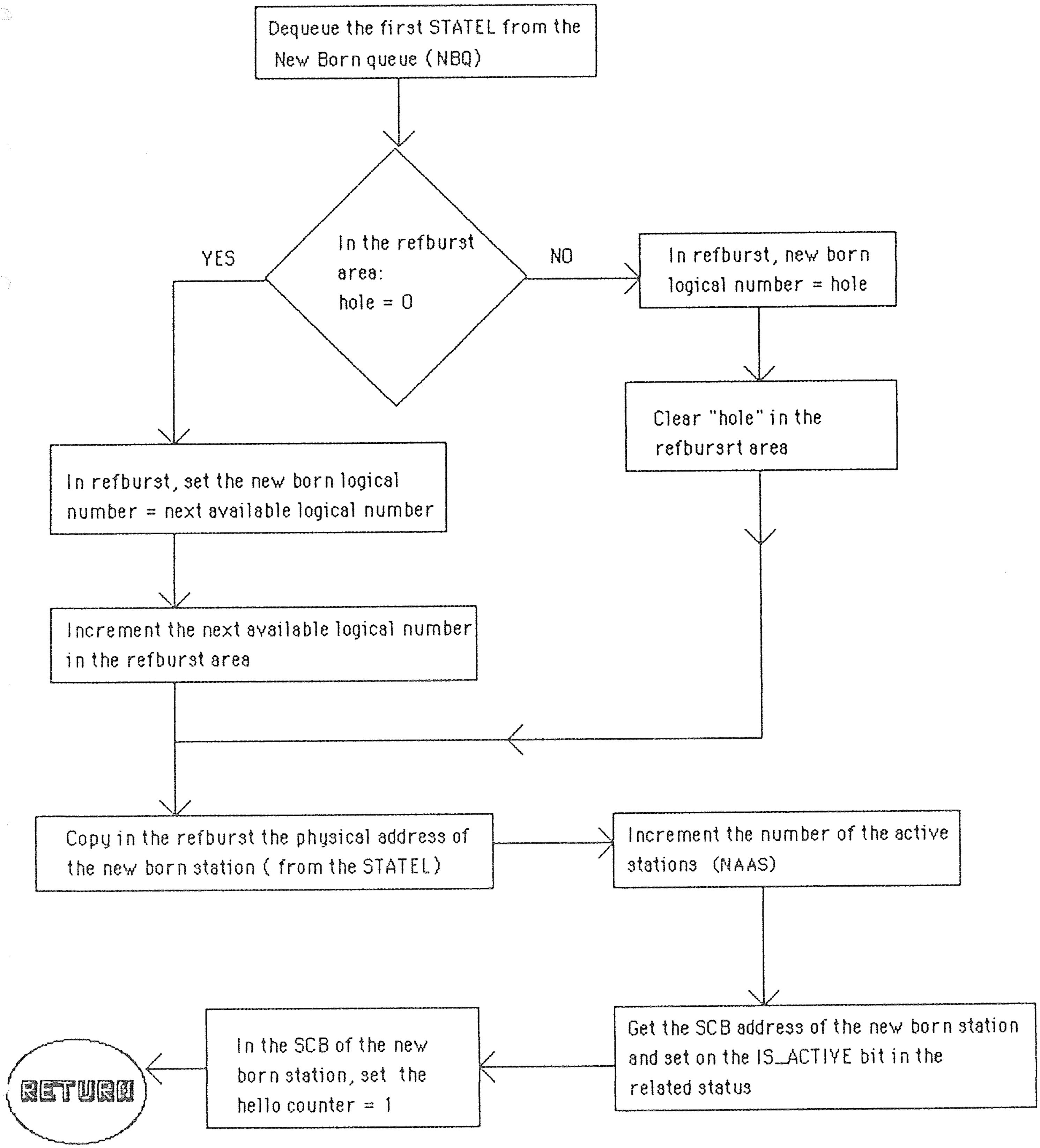



B
IS_SREQP bit
on in this SCB
YES
Dequeue this
SCB from the
SRQ queue
NO
Enqueue a STATEL for this SCB
in the down station queue DSQ
Set the SCB status = DOWN
A

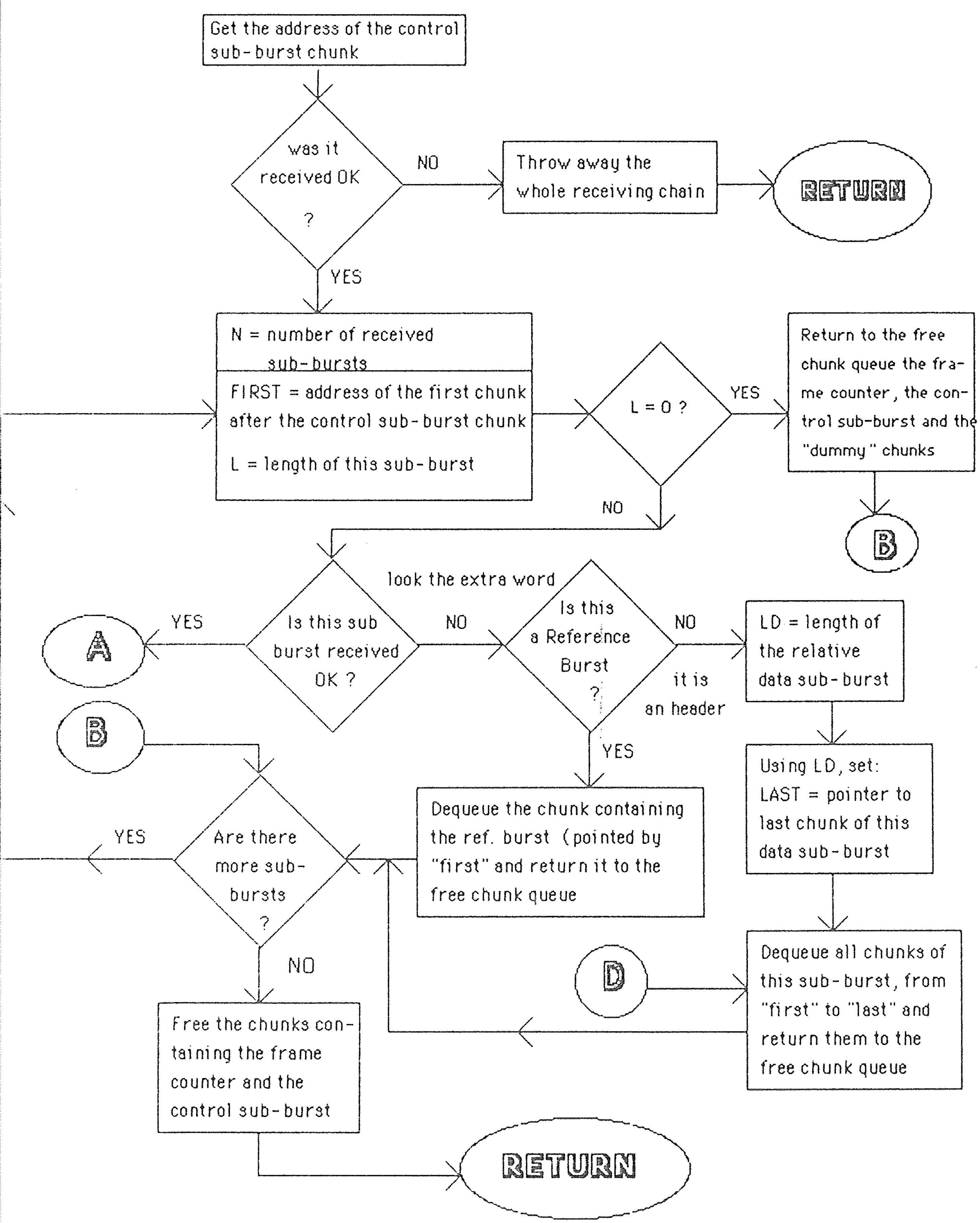
"Gone Down" handling in the Superframe Reference Burst :
FILL_GODOWN_SRB routine

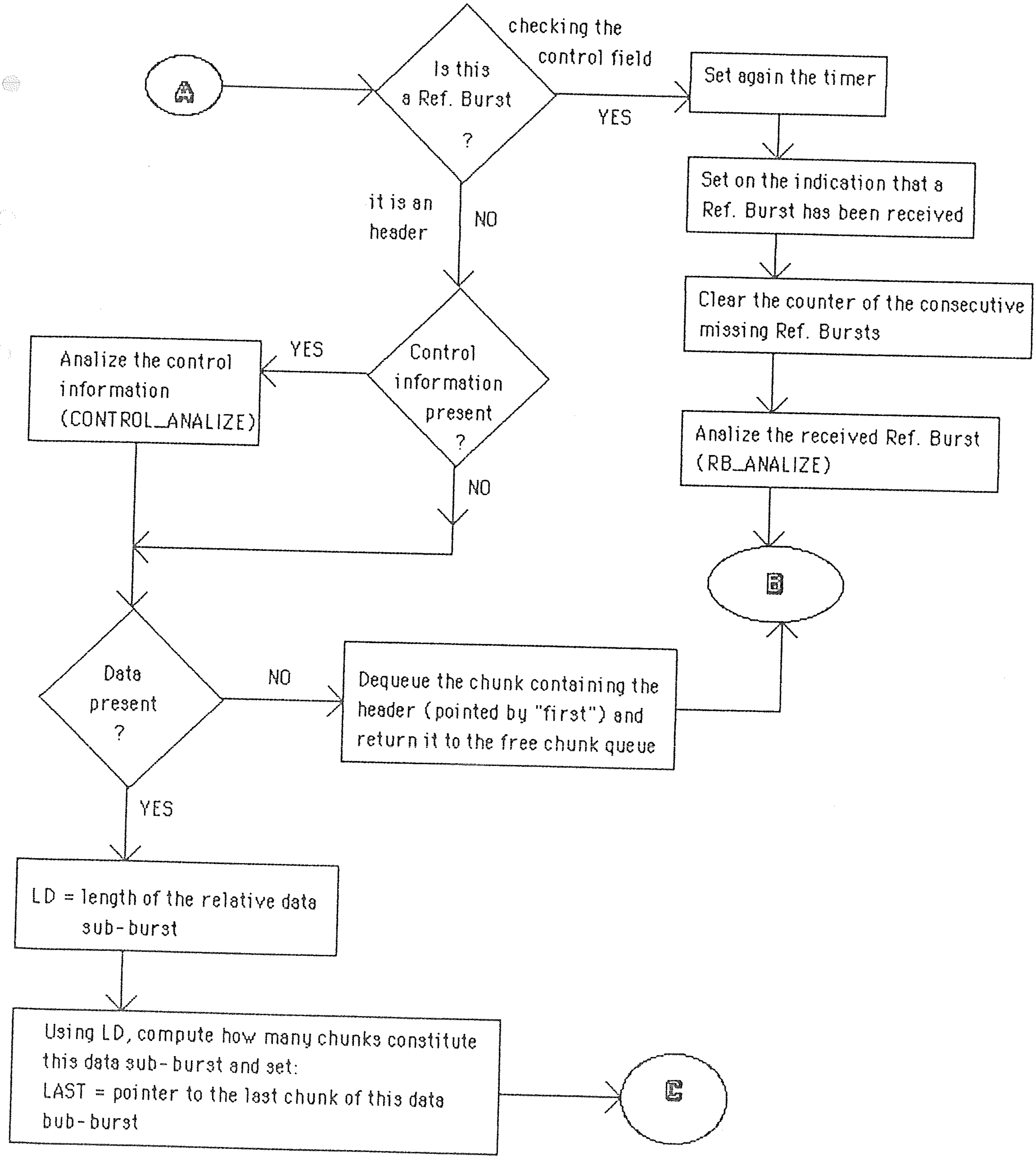


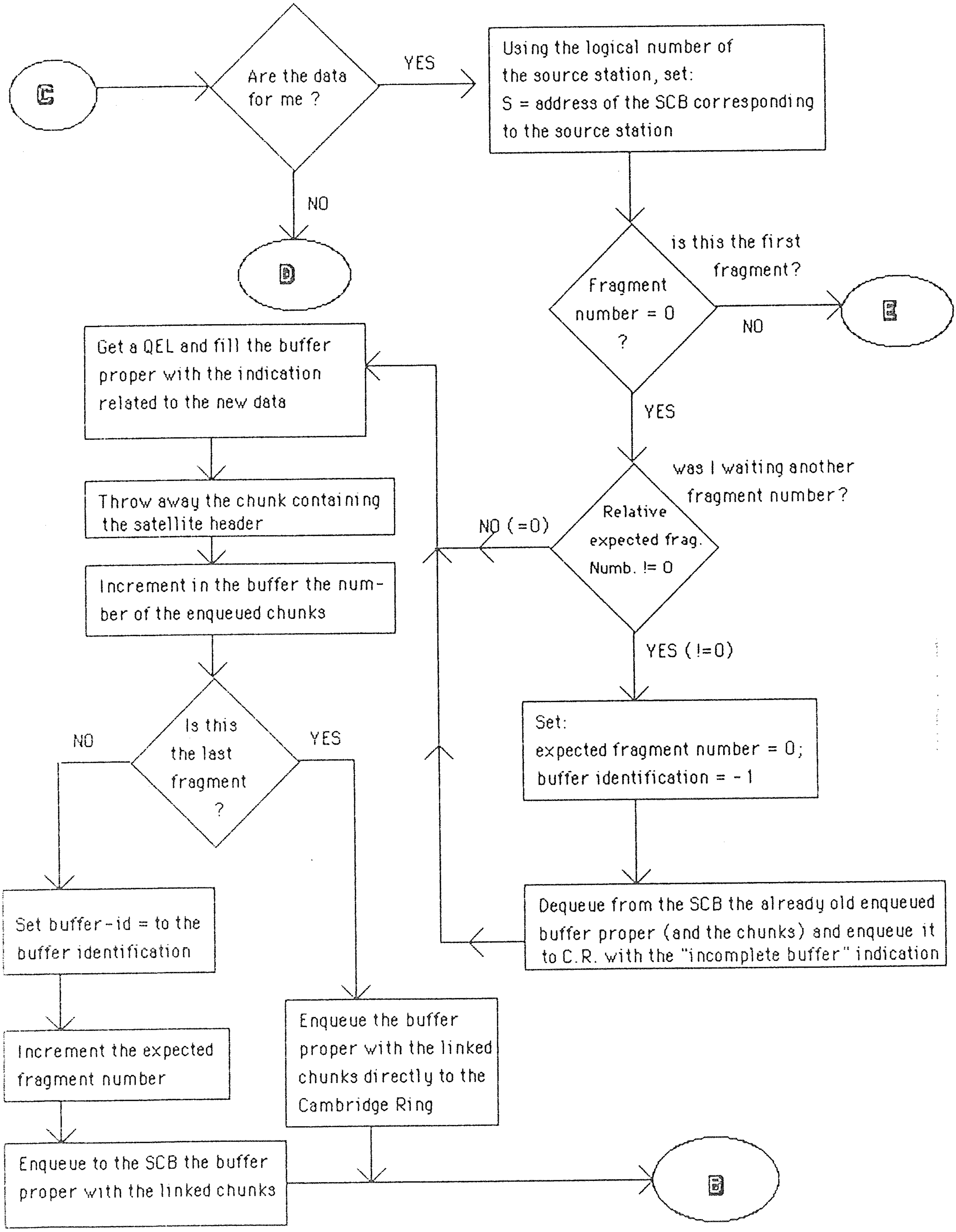
New Born Handling in the SRB: FILL_NEWBORN_SRB routine

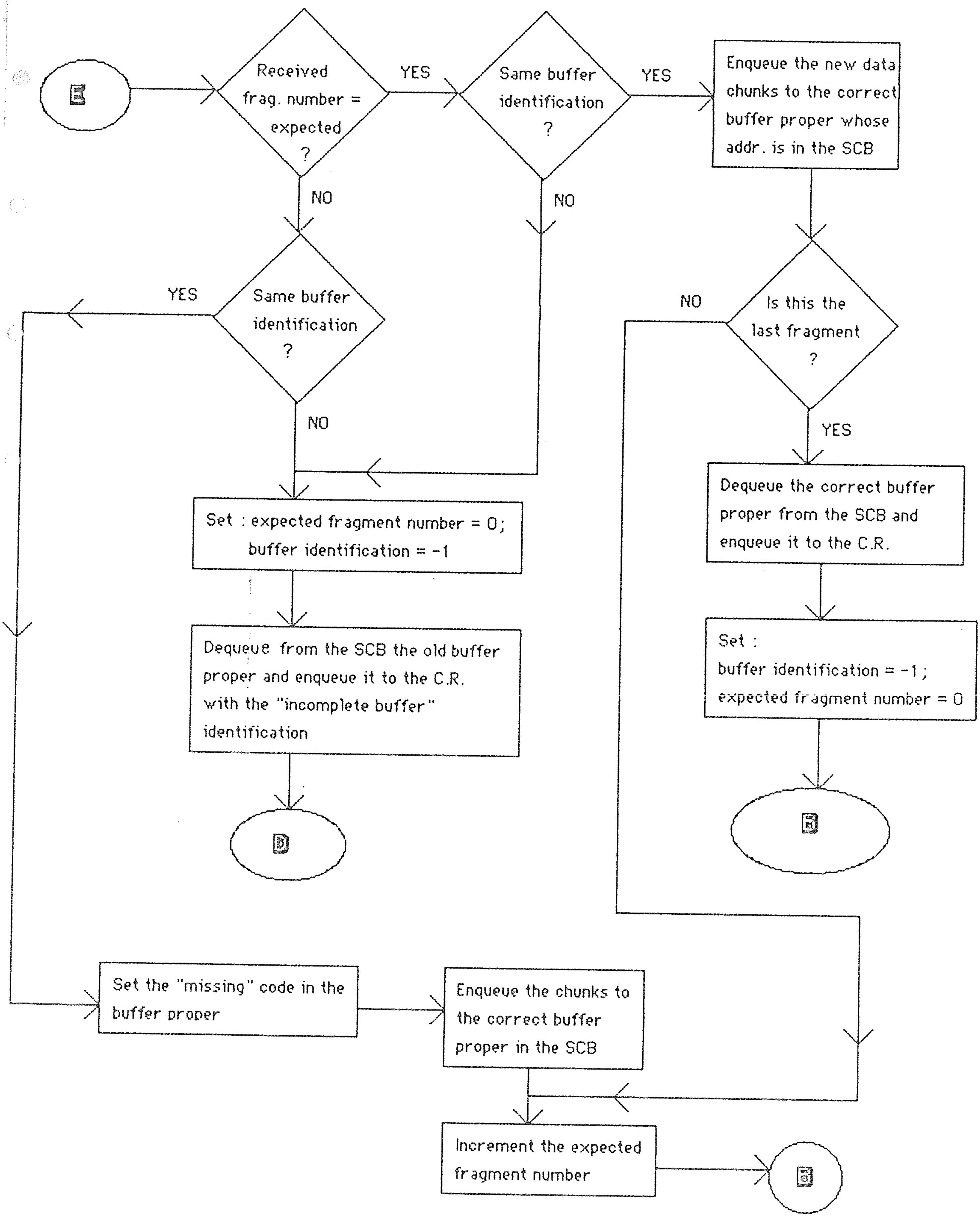


nalisys of the Burst received from satellite: BURST_ANALIZE routine

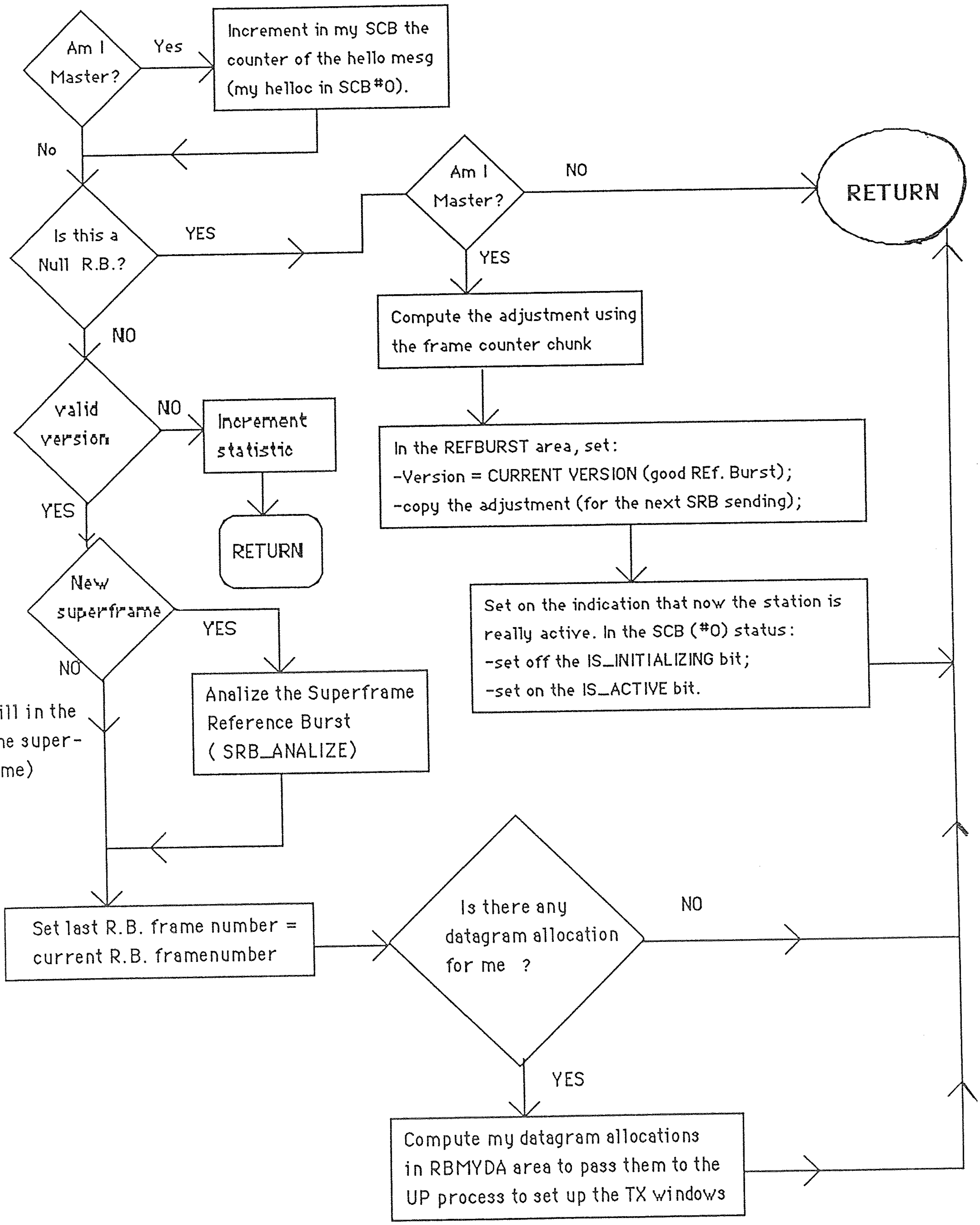



A
Is this
a Ref. Burst
?
checking the
control field
YES
Set again the timer
Set on the indication that a
Ref. Burst has been received
Clear the counter of the consecutive
missing Ref. Bursts
Analize the received Ref. Burst
(RB_ANALIZE)
B
it is an
header
NO
Control
information
present
?
YES
Analize the control
information
(CONTROL_ANALIZE)
NO
Data
present
?
NO
Dequeue the chunk containing the
header (pointed by "first") and
return it to the free chunk queue
YES
LD = length of the relative data
sub-burst
Using LD, compute how many chunks constitute
this data sub-burst and set:
LAST = pointer to the last chunk of this data
bub-burst
C

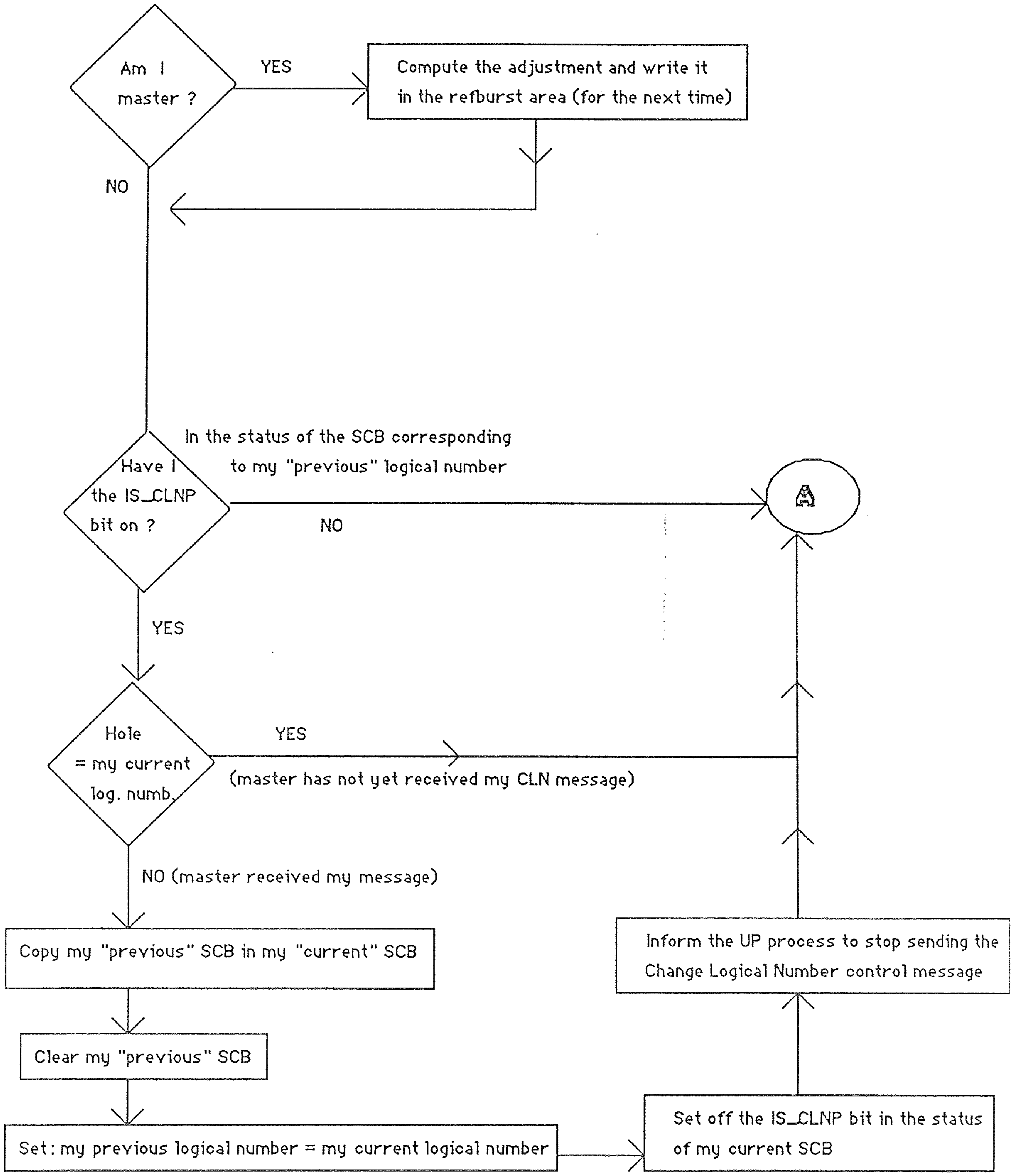

C
Are the data for me ?
YES
NO
D
Using the logical number of the source station, set:
S = address of the SCB corresponding to the source station
is this the first fragment?
Fragment number = 0 ?
NO
E
YES
was I waiting another fragment number?
Relative expected frag. Numb. != 0
NO (=0)
YES (!=0)
Set:
expected fragment number = 0;
buffer identification = - 1
Dequeue from the SCB the already old enqueued buffer proper (and the chunks) and enqueue it to C.R. with the "incomplete buffer" indication
Get a QEL and fill the buffer proper with the indication related to the new data
Throw away the chunk containing the satellite header
Increment in the buffer the number of the enqueued chunks
Is this the last fragment ?
NO
YES
Set buffer-id = to the buffer identification
Increment the expected fragment number
Enqueue to the SCB the buffer proper with the linked chunks
Enqueue the buffer proper with the linked chunks directly to the Cambridge Ring
B

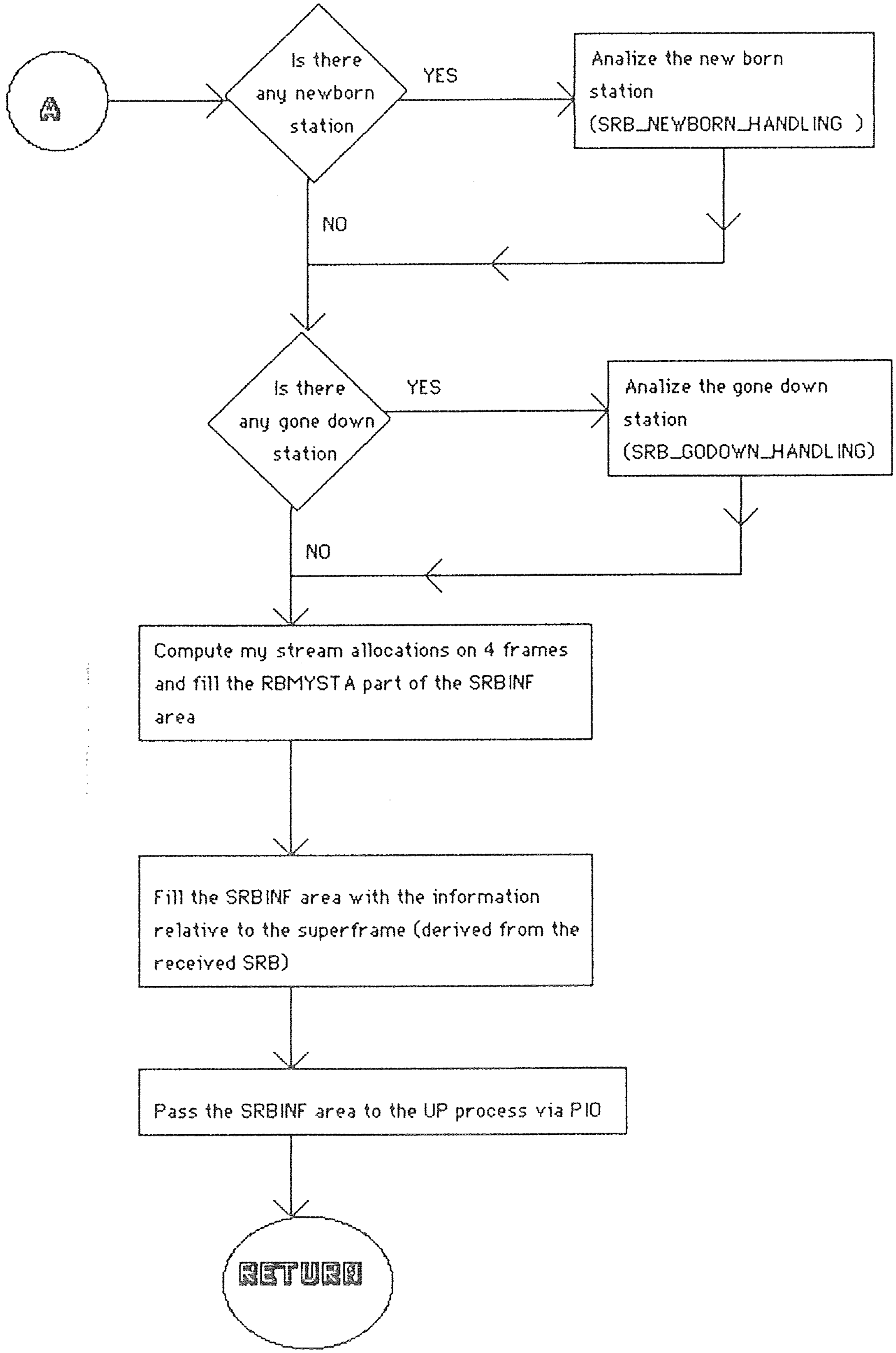

Received frag. number = expected ?
YES
NO
Same buffer identification ?
YES
NO
Enqueue the new data chunks to the correct buffer proper whose addr. is in the SCB
Same buffer identification ?
YES
NO
Is this the last fragment ?
NO
YES
Set : expected fragment number = 0; buffer identification = -1
Dequeue the correct buffer proper from the SCB and enqueue it to the C.R.
Dequeue from the SCB the old buffer proper and enqueue it to the C.R. with the "incomplete buffer" identification
Set : buffer identification = -1 ; expected fragment number = 0
Set the "missing" code in the buffer proper
Enqueue the chunks to the correct buffer proper in the SCB
Increment the expected fragment number

Analisys of the received Reference Burst: RB_ANALIZE routine

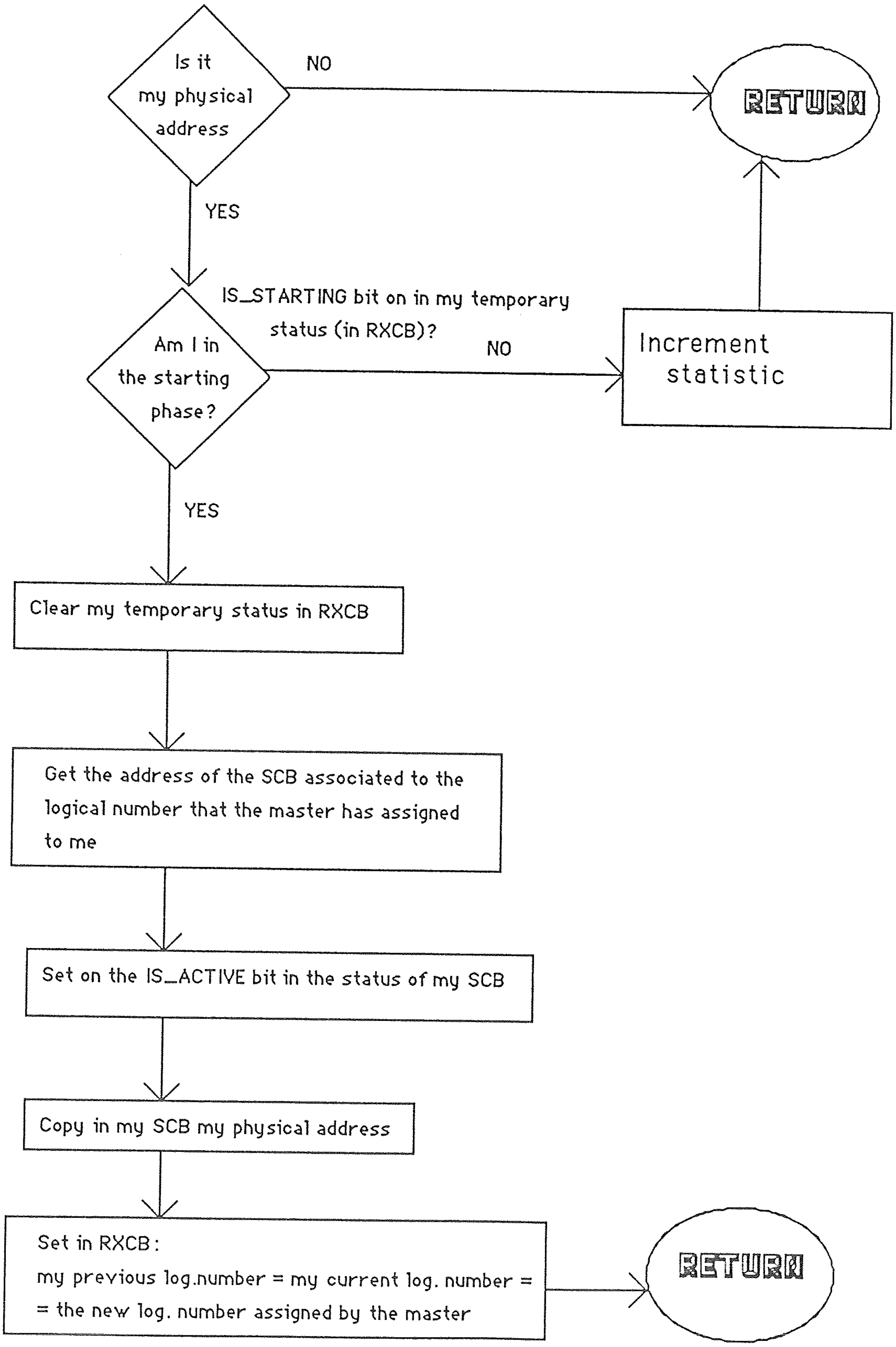


Analisys of the received SRB: **SRB_ANALIZE routine**

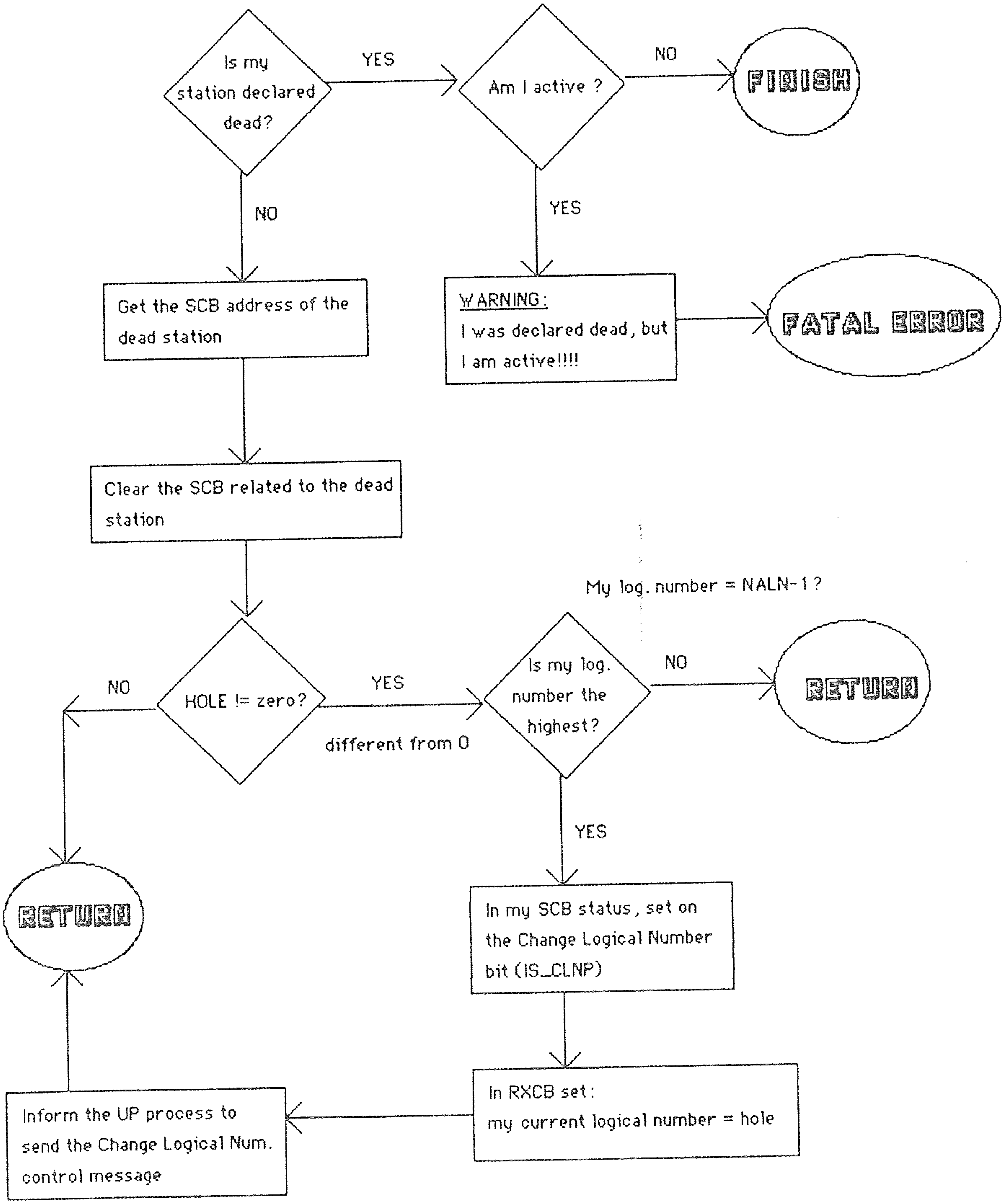



A
Is there any newborn station
YES
Analize the new born station (SRB_NEWBORN_HANDLING)
NO
Is there any gone down station
YES
Analize the gone down station (SRB_GODOWN_HANDLING)
NO
Compute my stream allocations on 4 frames and fill the RBMYSTA part of the SRBINF area
Fill the SRBINF area with the information relative to the superframe (derived from the received SRB)
Pass the SRBINF area to the UP process via PIO
RETURN

Analisys of the "Newborn Station" in the SRB: **SRB_NEWBORN_HANDLING** *routine.*

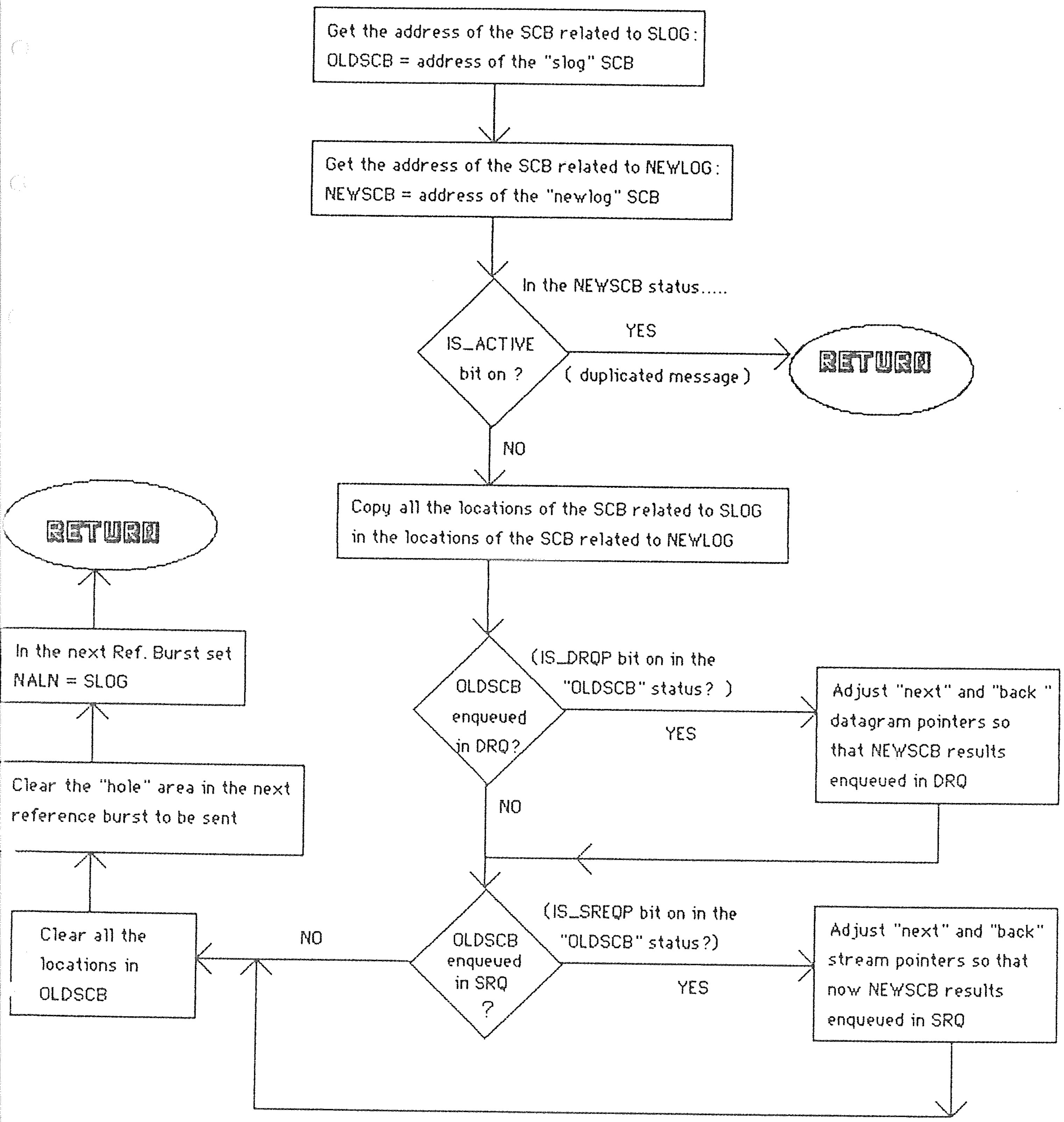


Analisys of the "Gone Down" station: SRB_GODOWN_HANDLING routine

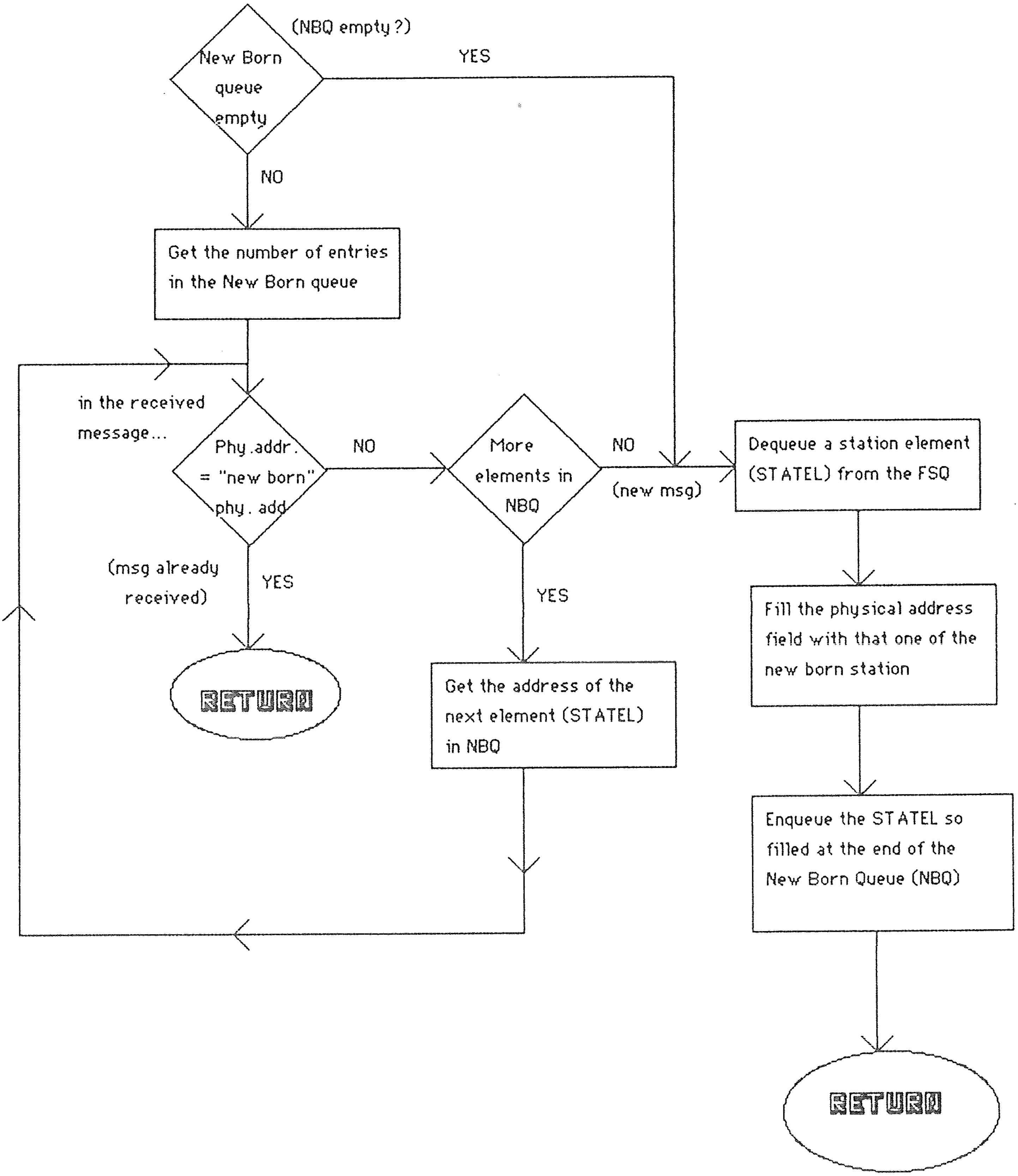


Receiving of a "Change Logical Number" control message (master): RX_CLN routine

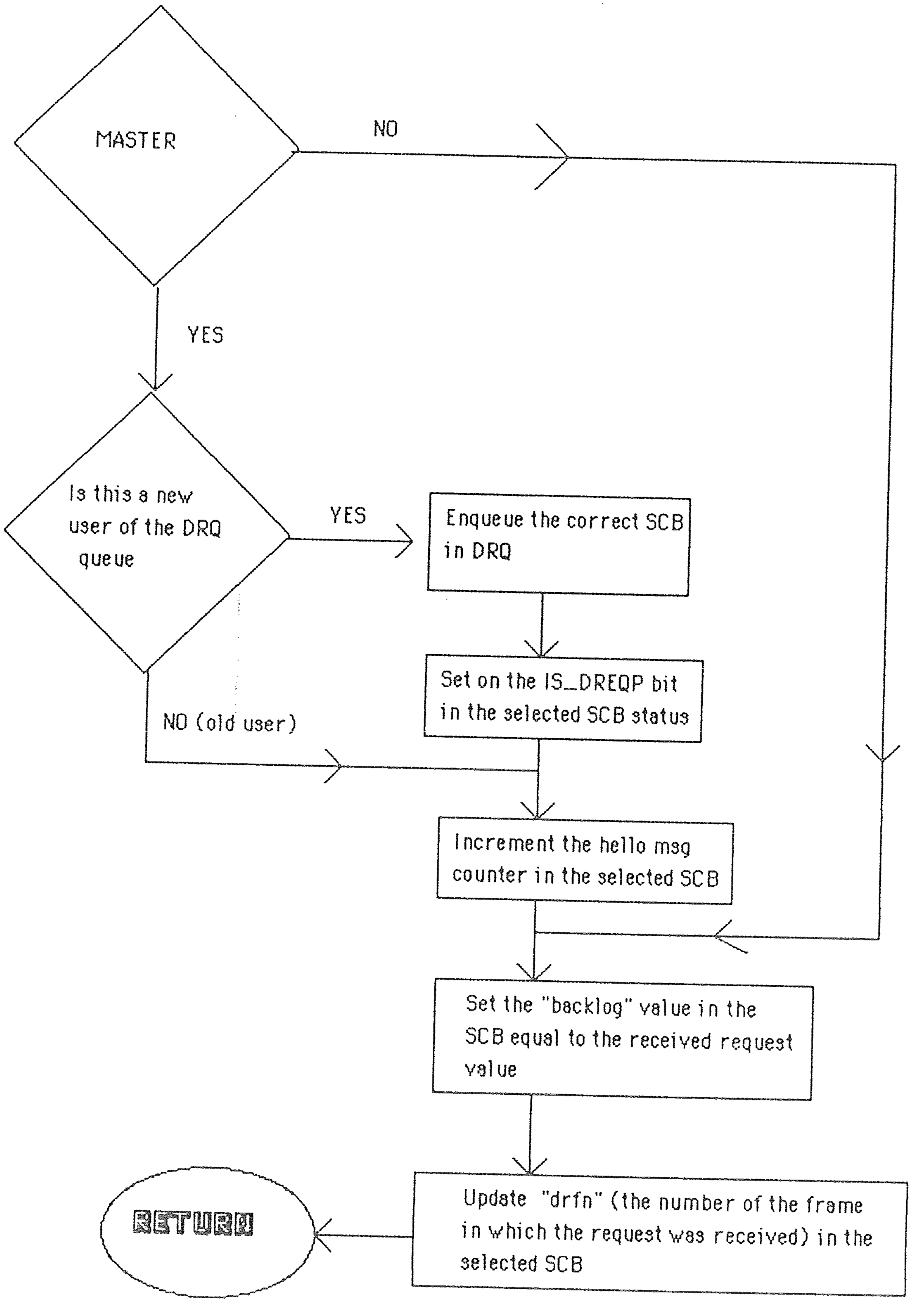
Note: SLOG = current logical number
NEWLOG = new logical number



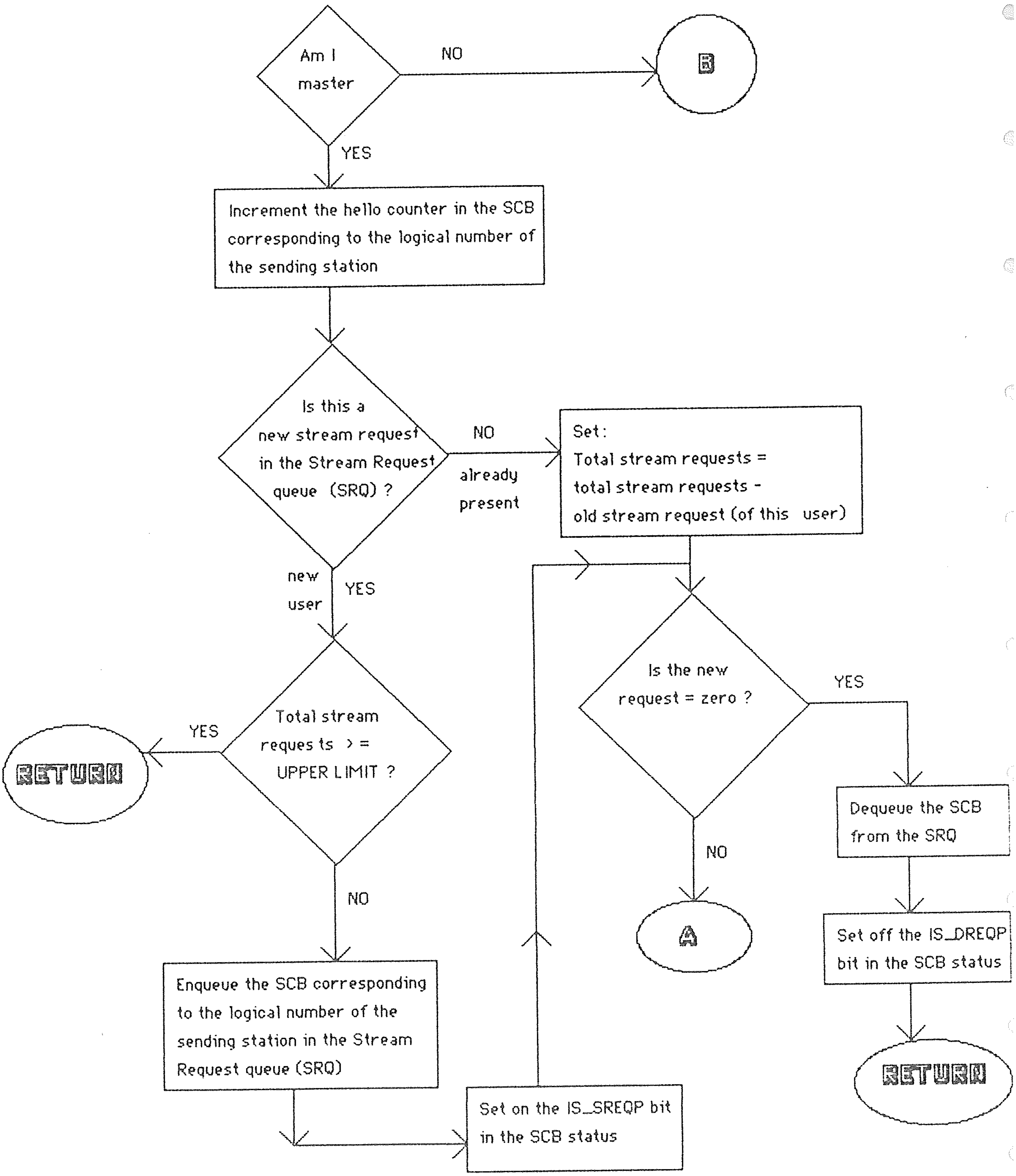
Receiving of a New Born request: RX_NEWBORN routine

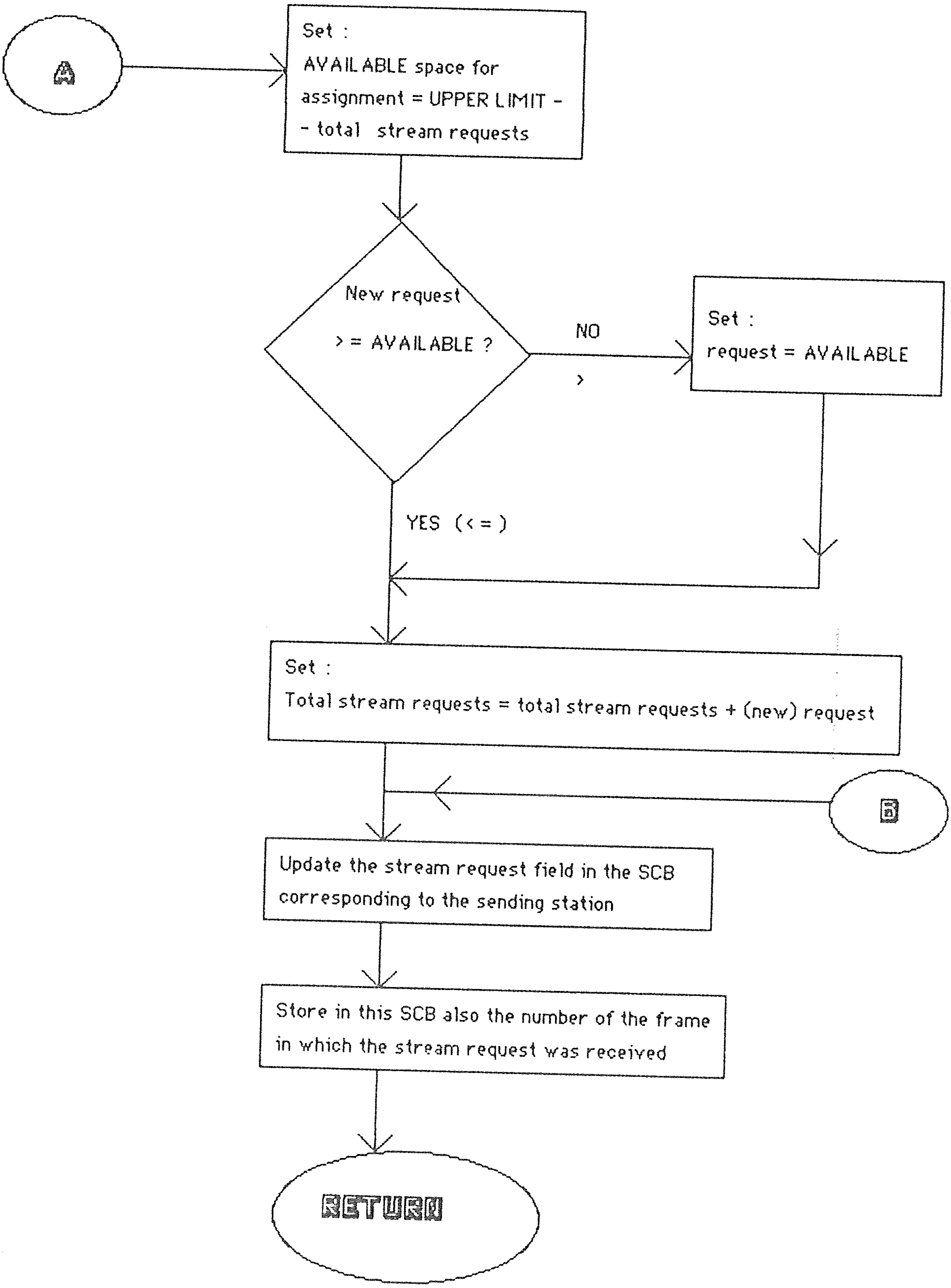


Receiving of a datagram request: DGREQ_RECEIVE



Behaviour on receiving of a stream request: STREQ_RECEIVE routine




A
Set :
AVAILABLE space for
assignment = UPPER LIMIT -
- total stream requests
New request
> = AVAILABLE ?
NO
>
Set :
request = AVAILABLE
YES (< =)
Set :
Total stream requests = total stream requests + (new) request
B
Update the stream request field in the SCB
corresponding to the sending station
Store in this SCB also the number of the frame
in which the stream request was received
RETURN

Second section: the UP process implementation

The behaviour of the transmitting side (UP process) of the new satellite bridge, running the FODA-TDMA satellite access scheme, is documented in this section.

The UP process working areas and some particular behaviours are described.

The flow-charts of the most significative routines are presented in order to make easier the understanding of the process and the reading of the software.

1. The UP process areas

1.1 The Window Descriptor Block (WDB)

```
bit-wide
               +-----------------------+
   32          | pointer to previous   |    last
               |      WDB              |
               +-----------------------+
   32          | pointer to next WDB   |    next
               |                       |
               +-----------------------+
   32          |   window type         |    wdbtype
               |                       |
               +-----------------------+
   8           |  frame number         |    frame_nr
               |                       |
               +-----------------------+
   8           |  counter of the       |    chunks_count
               | enqueued chunks       |
               +-----------------------+
   16          |  window start time    |    start_time
               |    (in bytes)         |
   ------------+-----------------------+------------16 bytes boundary
   16          |  available length     |    av_len
               |    (in bytes)         |
               +-----------------------+
   16          |  total length         |    tot_len
               |    (in bytes)         |
               +-----------------------+
   32          |  pointer to the last  |    lastchunk
               | enqueued chunk        |
               +-----------------------+
               |                       |
               |    PREAMBLE           |
               +-----------------------+
               |                       |
               | CONTROL-SUB-BURST     |
               +-----------------------+
```

WDBTAB is the table of the WDBs. They are built in a circular way (pointer to the next and backpointer to the previous). In the TXCB area (see following), the pointer to the first used WDB and the pointer to the first free WDB are maintained. This structure is used in order to save time.
The number of active WDBs is always less than the maximum number of WDBs in WDBTAB in order to avoid the overlapping between the head and the tail of the circular queue.

1.1.1 The PREAMBLE field

```
 bit-wide
              +-----------------------+
    16        |  preamble len         |  p_len
              |                       |
              +-----------------------+
    16        |  preamble code        |  p_code
              |                       |
              +-----------------------+
    32        | pointer to the        |  csb_ptr
              | control sub-burst     |
              +-----------------------+
    32        |      CBTRS field      |  cbtrs(0)
              |                       |
              +-----------------------+
    32        |      CBTRS field      |  cbtrs(1)
              |                       |
              +-----------------------+
                          .
                          .
              +-----------------------+
     32       |      CBTRS field      |  cbtrs (CBTRS_COUNT=6)
              |                       |
              +-----------------------+
     32       |    unique word (UW)   |  uw
              |                       |
              +-----------------------+
```

1.1.2 The Control sub-burst field

```
bit-wide
              +-----------------------+
    16        |  control sub-burst    |  csb_len
              |       length          |
              +-----------------------+
    16        |  control sub-burst    |  csb_code
              |     code              |
              +-----------------------+
    32        | pointer to the first  |  dsb_ptr
              |    data chunk         |
              +-----------------------+
    16        | counter of the number|  sb_count
              | of sub-bursts         |
              +-----------------------+
    16        |     control word      |  cw(0)
              |                       |
              +-----------------------+
    16        |     control word      |  cw(1)
              |                       |
              +-----------------------+
                         .
                         .
              +-----------------------+
    16        |     control word      |  cw(MAX_SB_COUNT=32)
              |                       |
              +-----------------------+
```

The control word format is the following:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C|C|C|C|L|L|L|L|L|L|L|L|L|L|L|L|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
15                             0    bits
```

C = top 4 bits: sub-burst encoding;

L = bottom 12 bits: sub-surst word length.

1.1.3 The window type sub-field

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|T|x|x|x|W|D|S|C|R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
31                                            9 8 7 6 5 4 3 2 1 0  bit
```

where:

R = is_RB bit: Reference Burst window (dummy or real);

C = is_CS bit: Control window;

S = is_ST bit: Stream window;

D = is_GD bit: Datagram window;

W = waiting_for_stream bit;

T = is_STARTED bit: window already started;

X = spare bits.

1.2 The transmission chunk

```
bit-wide
               +---------------------+
    16         |  data word length   |  tc_len
               |                     |
               +---------------------+
    16         |  data coding        |  tc_code
               |                     |
               +---------------------+
    32         | pointer to next     |  tc_ptr
               |      chunk          |
               +---------------------+
               |                     |  tc_data
               |                     |  (CHUNK_SIZE = 128 words)
               |   data part of the  |  Really used only 126 words
               |     chunk           |   (HARDWARE_DATACKSIZE)
               |                     |
               |                     |
               |                     |
               +---------------------+
```

The format of the transmission chunk is different from that one of the receiving chunk and the total length could be longer than 130 16-bit words because different are the hardware requirements. However, for semplicity, the CHUNK_SIZE value has been fixed equal to the one used in the Down process, and data are really filled in for HARDWARE_DATCKSIZE words.

The chunk data part can be:

a frame Counter
  or
a reference Burst
  or
an header
  or
a control sub-burst
  or
 user data

1.3 The chunk containing the fixed part of the Reference Burst (the RBFIX area)

```
bit-wide
           +-------------------------+
    16     |    word length of the   |   len
           |         data            |
           +-------------------------+
    16     |   data coding           |   code
           |                         |
           +-------------------------+
    32     |pointer to the chunk of  |   rbv_ptr
           |the R.B. variable part   |
           +-------------------------+
           |                         |
   192     |  Fixed part of the      |
(22 bytes) |   Reference Burst       |   fixed
           |  (see 1.7.5 of the      |
           |   Down description)     |
           +-------------------------+
           |                         |
   512     |  Max number of stream   |
(64 bytes) |  allocations in the     |   st_alloc
           |  superframe             |
           |                         |
           +-------------------------+
    32     |   back pointer to the   |   wdb_ptr
           |     window              |
           +-------------------------+
```

1.4 The chunk containing the variable part of the Reference Burst (the RBVAR area)

```
 bit-wide
                 +-------------------------+
     16          |    word length of the   |   len
                 |        data             |
                 +-------------------------+
     16          |   data coding           |   code
                 |                         |
                 +-------------------------+
     32          |   pointer to the null   |   rbvc_ptr
                 |       chunk             |
                 +-------------------------+
                 |                         |
    352          |  Max number of datagram|
 (44 bytes)|    allocations         |   dg_alloc
                 |  (plus extra 32 bits)   |
                 |   in a frame            |
                 +-------------------------+
```

Each datagram allocation has the following format:

```
 bit-wide
                 +-------------------------+
     16          |station physical address|
                 |   (rigth  justified)    |
                 +-------------------------+
     16          |  starting time          |
                 |  (in bytes)             |
                 +-------------------------+
```

1.5 The TX Control Block (TXCB)

```
  bit-wide
               +---------------------------+----------------------
     8         |   next frame number       | nextframe           |
               |                           |                     |
               +---------------------------+                     |
     8         |  active station counter   | ac_count          filled
               |rounded to next mult. of 4 |                     by
               +---------------------------+                    PIO
     8         | next available logical    | naln                |
               |                           |                     |
               +---------------------------+                     |
     8         | my logical number         | asln                |
               |                           |                     |
               +---------------------------+                     |
     8         |  current speed            | speed               |
               |                           |                     |
               +---------------------------+                     |
     8         *       alignment           *                     |
               *                           *                     |
               +---------------------------+                     |
     16        | bit adjustment for master | bitadj              |
               |                           |                     |
               +---------------------------+                     |
     64        | my stream assignment      | satab               |
  (8 bytes)    |    table                  |                     |
         ------+---------------------------+------------16 bytes boundary
     16        |  R.B. start time          | RB_start
               |   (in bytes)              |
               +---------------------------+
     16        |  R.B. window length       | RB_len
               |                           |
               +---------------------------+
     16        |  control Slot Window      | CS.w_start
               |    start time             |
               +---------------------------+
     16        |  control Slot Window      | CS.w_len
               |   byte length             |
               +---------------------------+
     8         |  control Slot step        | CS_step
               |                           |
               +---------------------------+
     8         | next frame in which a     | CS_nextframe
               | control slot is assigned  |
               +---------------------------+
     8         |  available stream channels| av_STch_count
               |                           |
               +---------------------------+
     8         *       alignment           *
               *                           *
               +---------------------------+
```

```
              +-----------------------------+
    128       |    Stream Window Table      |   swtab
(16 bytes)    |                             |
              +-----------------------------+
     32       |        status field         |   status
              |                             |
              +-----------------------------+
     32       |        action field         |   action
              |                             |
              +-----------------------------+
     16       |     buffer identifier       |   bufid
              |        counter              |
              +-----------------------------+
      8       |    time to perform the      |   tu_nextframe
              |    traffic update           |
              +-----------------------------+
      8       |    time to update the       |   DG_nextframe
              |    datagram request         |
              +-----------------------------+
     32       | pointer to first wdb        |   wdbhead
              |                             |
              +-----------------------------+
     32       | pointer to first free       |   wdbtail
              |       wdb                   |
              +-----------------------------+
      8       | counter of the outstanding  |   wdbcount
              |        wdb                  |
              +-----------------------------+
      8       *     alignment               *
              *                             *
              +-----------------------------+
```

1.5.1 My stream assignment table (SATAB sub-field)

bit-wide		
8	start slot number (for frame #0s)	start_slot
8	number of slots	slots_count
8	start slot number (for frame #1s)	start_slot
8	number of slots	slots_count
8	start slot number (for frame #2s)	start_slot
8	number of slots	slots_count
8	start slot number (for frame #3s)	start_slot
8	number of slots	slots_count

1.5.2 The Stream Window Table (SWTAB sub-field)

```
bit-wide
          +--------------------------------+
   16     |     window start time          |     w_start
          |     (window of frames #0s)     |
          +--------------------------------+
   16     |     window byte length         |     w_len
          |                                |
          +--------------------------------+
   16     |     window start time          |     w_start
          |     (window of frames #1s)     |
          +--------------------------------+
   16     |     window byte length         |     w_len
          |                                |
          +--------------------------------+
   16     |     window start time          |     w_start
          |     (window of frames #2s)     |
          +--------------------------------+
   16     |     window byte length         |     w_len
          |                                |
          +--------------------------------+
   16     |     window start time          |     w_start
          |      (window of frames #3s)    |
          +--------------------------------+
   16     |     window byte length         |     w_len
          |                                |
          +--------------------------------+
```

1.5.3 The status sub-field

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|K|N|x|x|x|T|R|F|M|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
31                                              8 7       3 2 1 0   bit
```

where:

M = is_master bit: the station is acting as master;

F = is_SFenable bit: enable superframe;

R = is_RBenable bit: enable reference burst;

T = is_txpending bit: transmission pending;

N = is_NFupdating bit: when off, the next frame number ("nextframe" field of the TXCB) has already been updated.

K = is_rbsent bit: when on, a reference burst EOB has been received. It is set off in the tx_te() routine if found on when a timer interrupt occurs. If found off, the "skipped_rb" field in the tx statistic area is increased.

X = spare bit.

1.5.4 The action sub-field

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|x|R|C|S|D|H|N|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
31+                                                   5 4 3 2 1 0- bit
```

where:

N = is_sendnewborn bit: the new born control message must be sent by the slave station;

H = is_sendhello bit: the hello control message must be sent;

D = is_sendDGreq bit: the datagram request must be sent;

S = is_sendSTreq bit: the stream request must be sent;

C = is_sendcln bit: the change logical number control message must be sent;

R = is_phyarrived bit: the station physical address and the tx (slave) station offset have been received via PIO message. The traffic generator can be started, if selected.

X = spare bit.

1.6 DAB, SITEINF and FRAME_AREA

These three areas are filled by PIO on the base of the control codes MY_DATAG_ALLOC, PHY_SITE and DOWN_FRAME respectively.
They are filled with the datagram allocations for my station, with the station physical address and relative tx slave station offset value and with the current frame number used by the down process, respectively.

```
   DAB
  -----

              +--------------------------------+
     8        |  number of the reference burst|   dw_frame
              | containing my allocations      |
              +--------------------------------+
     8        |  number of my datagram entries|   dw_count
              |                                |
              +--------------------------------+---------------
    16        | 1st datagram starting time     |              |
              |                                |              |
              +--------------------------------+              |
    16        |  relative length (multiple of  |              |
              |  8 bits)                       |              V
              +--------------------------------+  datab(MAX_DW_COUNT)
    16        | 2nd datagram starting time     |
              |                                |
              +--------------------------------+
    16        |  relative length (multiple of  |
              |  8 bits)                       |
              +--------------------------------+
    16        |               .                |
              |               .                |
              |               .                |
    16        |               .                |
              |               .                |
              +--------------------------------+
```

SITEINF

```
---------

            +---------------------------------+
     8      |  my physical address            |   phya
            |                                 |
            +---------------------------------+
     8      |         spare                   |   site_spare
            |                                 |
            +---------------------------------+
    16      | my tx (slave) station offset    |   tx_stoff
            |                                 |
            +---------------------------------+
```

FRAME_AREA

```
------------

            +---------------------------------+
     8      |  current frame number used      |   rx_frame
            |  in the down process            |
            +---------------------------------+
     8      |         spare                   |   rx_spare
            |                                 |
            +---------------------------------+
```

1.7 TXSTAT (TX STATistic block)

The TX statistic block is defined 24 words long. Each word is a counter, a part the very last one.

```
          +--------------------------------+
  16      |   spurious PIO messages        |  spurious_pio (0)
          |                                |
          +--------------------------------+
  16      |   duplicated windows           |  duplicated_
          |                                |  window (1)
          +--------------------------------+
  16      |times in which the transmission|  skipped_rb (2)
          |of a R.B. has been skipped      |
          +--------------------------------+
  16      |                                |
          |                                |
          +--------------------------------+
  16      .                                .
          .                                .
          +--------------------------------+
  16      |                                |
          |                                |
          +--------------------------------+
  16      |  delay (in msec) of the        |  delay (23)
          |  delay measurement packet      |
          +--------------------------------+
```

2. Master/slave behaviour

The Up process always acts as master, i.e. it always prepares the window for sending the reference burst at the beginning of each frame. If it is really master, a real reference burst will be transmitted in that window (data length different from zero); if slave, a dummy reference burst (data length = 0) is sent.

At the starting time, the Up process starts sending dummy reference bursts, till when PIO messages from the Down process change the situation (see chapter 6. of section I).

In fact, if a MASTER_SRB PIO message is received, the station is automatically declared master, and a real superframe reference burst (received by the Down process) is transmitted in the reference burst window. In this case the logical number assigned to the station is zero.

If a SLAVE_SRB message is received, the assigned logical number is checked. If it is negative, the Up process assumes that the station is slave and its logical number has not yet been assigned by the master. So the "newborn" control message is sent in the control window computed on the base of the next available logical number (which is preempted by the station).

The confirmation that the master has received the newborn control message and that a valid logical number has been assigned is detected by the Up process as soon as a positive "my" logical number is passed from the Down process with another SLAVE_SRB control message.

3. Interfacing between the Up process and the LAN handling task

Three queues

streamq	for stream connections
interactq	for high priority datagram
bulkq	for low priority datagram

are used in the FODA system. They are filled by the LAN handler task on the base of the type of traffic. The UP process gets data from the queues for transmission on the base of the data priority.

In the queues the buffer elements (el) are enqueued. They, together with the buffer proper, are described in 8.1 and 8.2 of the first section.

The following fields in the buffer proper (buf) and in the data header (hdr) must be filled in by the LAN:

buf -->pkt	pointer to the first byte of packet data;
buf -->checksum	checksum value;
buf -->bytecount	number of bytes in data packet;
buf -->b_dat	pointer to first tx chunk in the chain;
buf -->chunk_count	number of the chunks enqueued to the buffer;
buf -->b_code	coding level of the buffer;
buf -->b_head	pointer to the header chunk (hdr);
hdr -->tc_code	header level coding;
hdr -->tc_len	header length;
hdr -->dest	destination (physical) address;
hdr -->h_port	port label in bridge;
hdr -->h_length	length in bytes of the basic block;
hdr -->h_check	basic block checksum;
hdr -->h_head	header type of basic block.

In addition, each chunk in the chunk chain must have the correct size filled in. The sum of these chunks sizes must agree with both "bytecount" and "h_length".

The various routines on the Up side are activated by calling them directly:

```
tx_s_in    for stream traffic
tx_i_in    for interactive traffic
tx_b_in    for bulk traffic.
```

4. TX window, burst building and DMA I/O handling time

```
      RBn                RBn+1                RBn+2                RBn+3
       |                   |                    |                    |
       |                   |                    |                    |
       V                   V                    V                    V
-----------------------------------------------------------------------------RX

                                                                        frames

-----------------------------------------------------------------------------TX
 |     PIO EOB       | |____|      |__| |                  |
 |                   |  R.B.       dtg  |                  |
 |                   |+control          |                  |
                      +stream
      n-1                   n                n+1                  n+2
```

At the beginning of the transmission of frame n-1 (dummy or real DMA EOB), the following windows for frame n are built by the UP process:

--Reference — in which a dummy (if slave) or a real (if master) transmission of a reference burst will be done;

and, if n less than 64

--Control — if a control window is allowed in this frame;

--Stream — if a stream window is allowed in this frame.

At the time PIO EOB, the time allocations for this station (which have been received in the reference burst "n") are received from the Down process. The following windows for the frame n are built:

--Control — if RBn is the Superframe Ref. Burst;

--Stream — if RBn is the Superframe Ref. Burst;

--Datagram — if the station has any datagram allocation for that frame.

At a suitable time, a little before the beginning of the transmitting frame n, the eventual window in the frame n reserved for the stream traffic is made available also for the datagram traffic. The chosen time to do that is at the Tx timer interrupt.

The building of the bursts are attempted at the following times:

- after any window building;
- at the moment (described above), in which stream windows are made available for datagram;
- on reception of a buffer from the local area network;
- at the timer interrupt for building the reference burst (master only).

The starting of the I/O operation on the Tx DMA interface is performed at the DMA EOB of the previous operation. At this time the window relative to the already transmitted burst is also dequeued.

It is supposed that at least one window is always present and one DMA operation is enqueued to the interface (dummy reference burst, if no real data are available).

The only exception is the starting of the reference burst transmission in case of master. In this case the operation may be performed only if the is_RBenable bit is set on in the TXCB status field. Such a bit is set on only after that the datagram allocations for that frame are received via PIO message from the Down process. The PIO message (see chapter 6. of the Down desription) is any way received, even if no allocation is present. This procedure is necessary because, at the starting of the DMA process, the FIFO (64 16-bit words) is filled immediately with the preamble, the control sub-burst and part of the data su-burst, making impossible any addition of further information to the burst.

5. PIO messages incoming from the Down process

As already described in the chapter 6. of the first section of this manual, the PIO communications are always done by the Down process toward the Up process.
On the base of the PIO code, data are received directly in the following areas (the PIO code and length not being passed):

MASTER_SRB	the RBFIX area is filled with the fixed part of the received superframe reference burst (stream allocations included). The TXM_SRB routine is entered.
MASTER_DALLOC	the RBVAR area is filled with the global datagram allocations. The TXM_DA routine is entered.
MY_DATAG_ALLOC	the DAB area is filled with the datagram allocations relative to the station and valid for next frame. The TXS_DA routine is entered.
SLAVE_SRB	the first 16 8-bit bytes of the TXCB area are used to receive the information regarding the received superframe reference burst. The TXS_SRB routine is entered.
SEND_CLN	the CLN area is filled with the new logical number the station is going to assume. The TX_CLN routine is entered.
STOP_CLN	no area is filled. The TX_SLN routine is entered just to set off the "is_sendcln" bit in the TXCB action field.
PHY_SITE	the SITEINF area is filled with the station physical address and the tx (slave) station offset. This two values are also copied in the "thissite" and in the "STATION_OFFSET" global areas respectively. The TX_SITE routine is entered.
DELAY_MEASUREMENT	no area is filled. The TXD_MSR routine is entered. This is a very special routine written to display the delay of the FODA system after the reception of the very special delay measurement packet (see Appendix A).
DOWN_FRAME	the FRAME_AREA area is filled with the current frame number used in the down process (synchronization of the frame numbers). The TXS_DA routine is entered.

6. The transmitting chunks chain

The Up process has three queues (interactive, bulk and stream) to receive data from the local area network.
When a buffer is enqueued in one of the three queues, the data structure is the following:

```
+---------+   +-----------+
| Buffer  |-->|           |    +--------+
| element |   |   Buffer  |-->| Header |
+---------+   |   Proper  |   | chunk  |
              |           |   +--------+
              |           |
              |           |   +--------+   +--------+     +--------+
              |           |-->|  data  |-->| data   |..-->|  data  |
              |           |   |  chunk |   | chunk  |     |  chunk |
              +-----------+   +--------+   +--------+     +--------+
```

and the buffer element is enqueued at the end of the selected queue.

One chunk at a time, starting from the header chunk, is delinked from the buffer and enqueued to the WDB relative to the transmitting window in which the data will be transferred.
After each enqueuing, the field "tc_ptr" of the chunk enqueued to the WDB is loaded in such a way to point to the null chunk (a field containing zero) just in case that there is no more time to enqueue more chunks.

Before the transmission, the word length field of each chunk must be increased by 3 because of the TX hardware requirements.

If a chunk is too big for the current window (i.e. not all the data can be transferred), an empty chunk is get and a copy of the header is prepared for the next data chunks which have to be transferred. Moreover, another chunk is get and the remaining data part of the chunk which cannot be transferred is copied. Both the new chunks are enqueued to the buffer proper.

When all the chunks of a buffer are transferred, the buffer proper and the buffer element are released. When a buffer proper is released, a chunk is always assumed to be linked in the "b_head" position.

7. The transmitting coding values

The coding values required by the Tx hardware and their memory allocations are the following:

Area containing the code value	Coded value	Unencoded value	Memory allocations c=coded; u=unencoded
Chunk containing the control sub-burst	0x0004	---	c = CSBCHUNK_CODING u = ---
Data coding in the control sub-burst	0x0	0x2000	c = CSBCW_CODING u = CSBCW_UNCODING
Data coding in the chunk	0x0004	0x0	c = DCHUNK_CODING u = DCHUNK_UNCODING
Preamble	0x0	---	c = PREAMBLE_CODING u = ---

8. The Tx file organization in the C language implementation

The routines implementing the FODA-TDMA satellite access scheme on the Up process are organized in files in the following way:

TX.H	header file; it defines variables used by the Up process only.
TXHARDW.H	header file; it contains some definitions relative to the hardware.
TXTEMPL.H	header file; it contains the structures of the areas defined by the Up process.
TX.X	header file; it contains external references.
TXALLOC.C	it contains the real allocations of the Tx areas.
TXINI.C	it contains the routines used at the initialization time:

TX_INI : main initialization program;
TXI_WDBQ_INI : to initialize the WDBs circular queue.

TXPIO.C	it contains the routines entered when a PIO read operation is completed:

TXS_DA	on reception of the datagram allocation of this station;
TXS_SRB	on reception of the information relative to the new superframe;
TXM_SRB	on reception of the superframe reference burst to be transmitted;
TXM_DA	on reception of the global datagram allocations to be transmitted in the reference burst;
TX_CLN	on reception of the command to send the change logical number control message;
TX_SLN	on reception of the command to stop sending the change logical number control message.
TXD_MSR	on reception of the DMP on the down side;
TX_SITE	on reception of the information regarding the station physical address and the tx slave station offset.
TX_FRAME	on reception of the current frame number used on the down process.

TXTIM.C	it contains all the routines invoked when a timer interrupt occurs:

TX_TE	entered when a timer interrupt occurs;
TX_CLEAR_WFS	to clear the "waiting for stream" condition in the stream window of the current frame.

TXRING.C — it contains the routines invoked when data are put in the appropriate queue from the LAN (in this implementation a Cambridge Ring):

TX_I_IN	entered by the LAN software when data are put in the interactive data queue;
TX_B_IN	entered by the LAN software when data are put in the bulk data queue;
TX_S_IN	entered by the LAN software when data are put in the stream data queue.

TXDMA.C — it contains the routines related to the DMA operations:

TX_EOB	entered on a DMA EOB operation;
TX_START_DMA	entered to start a DMA operation.

TXWINDOW.C — it contains the window building routines:

TX_RBWB	to build a reference burst window;
TX_CSWB	to build a control slot window;
TX_STWB	to build a stream window;
TX_WINDOW_SETUP	to build a datagram window of the specified type.

TXBURST.C — it contains the burst building routines:

TX_RBB	to build the reference burst;
TX_AC	to add control information to a burst;
TX_DGBB	to build a datagram burst;
TX_STBB	to built a stream burst;
TX_ADD_FRAGMENT	to add a fragment to a burst;
TX_HEADER_CHAIN	to add an header chunk to the burst chunk chain;
TX_CHAIN_CHUNK	to add a data chunk to a burst chunk chain;
TX_PREPARE_HEADER	to prepare a new header for a subsequent fragment of the same buffer;
FILL_HDR_FIELDS	to fill the fields of a new header.

TXCONFIG.C	it contains the device configuration table and the user task definition table for the Up process.
TXMSGH.C	the "msghandler" message handler task for the Up process. Messages sent at interrupt level are displayed on terminal and/or recorded in a special file.
TXFRTG.C	fixed rate traffic generator for the Up process. Mixed data are generated, on the base of some operator specifications, simulating their incoming from the attached LAN.
TXASS.S	some assembler routines are here defined.
TXIN68K.S	the assembler file linking the Up FODA software to the C-EXEC nucleus.
UBU70.C	files containing the "upring" LAN interface task.
UBU100.C	the "upuser" task, handling the interface between the FODA system and the operator.

COMMON FILES

The following files contain definitions or allocations valid for both the Down and the Up processes and for the software of the local area network linked to the satellite TDMA controller:

UBC*.H	header files; common definitions of variables and structures are here defined.
RXTX.H	header file; it defines variables and structures used by both the Down and the Up processs.
RXTXALLOC.C	it contains the routine ALLOC used to allocate in memory common variables.
RXTXEXT.X	it contains common external references.
UB10/30/60.C	LAN interface common files.
UB200.C	queues and chunks handling routines.
UB210.C	PIA initialization.
XC10A68K.C	routines relative to the LAN handler.
XC11A68K.C	routines relative to the LAN handler.
XC13DUMMY.C	routines relative to the LAN handler.

UBA1.S — PIO handling assembler routines are here defined:

pia_bin — to read from PIO,

pia_bout — to write on PIO.

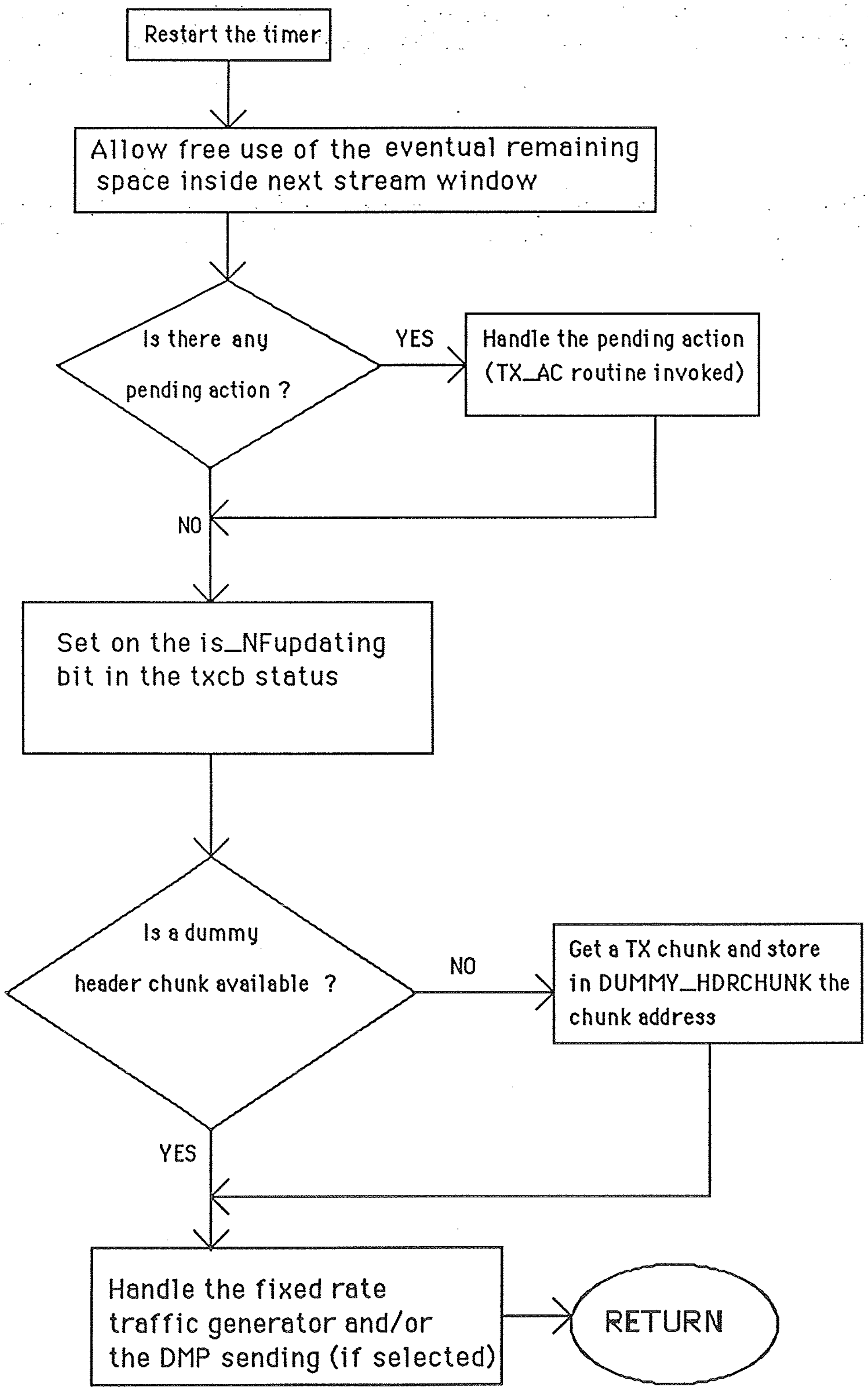
9. The interrupt levels in the TX process

```
read DMA EOB  -------> level 6
PIO read  -----------> level 5
dummy DMA EOB -------> level 4
TIMER  --------------> level 3
TERMINAL ------------> level 2
LAN read ------------> level 1
```

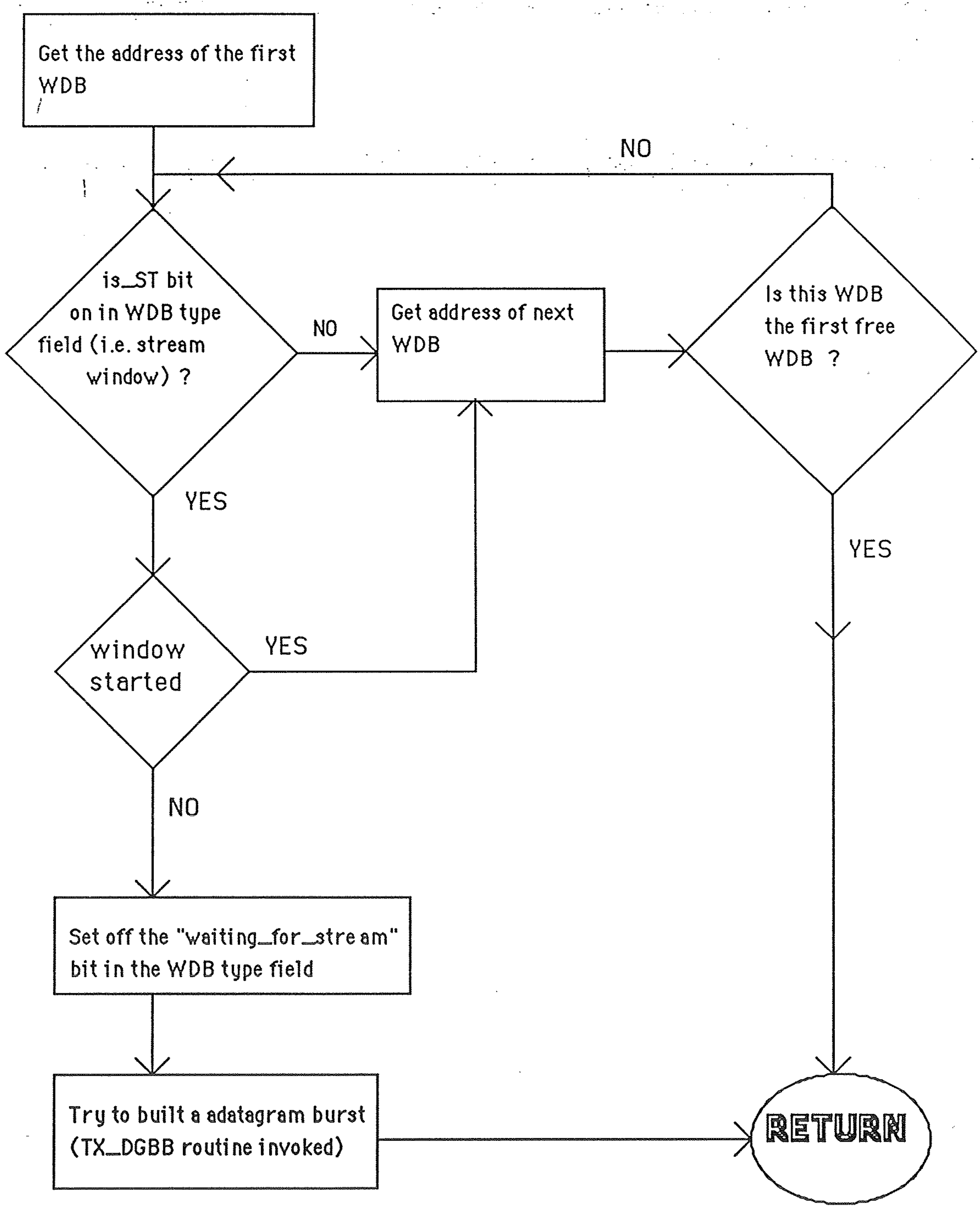
The TX Processor

Flow Charts

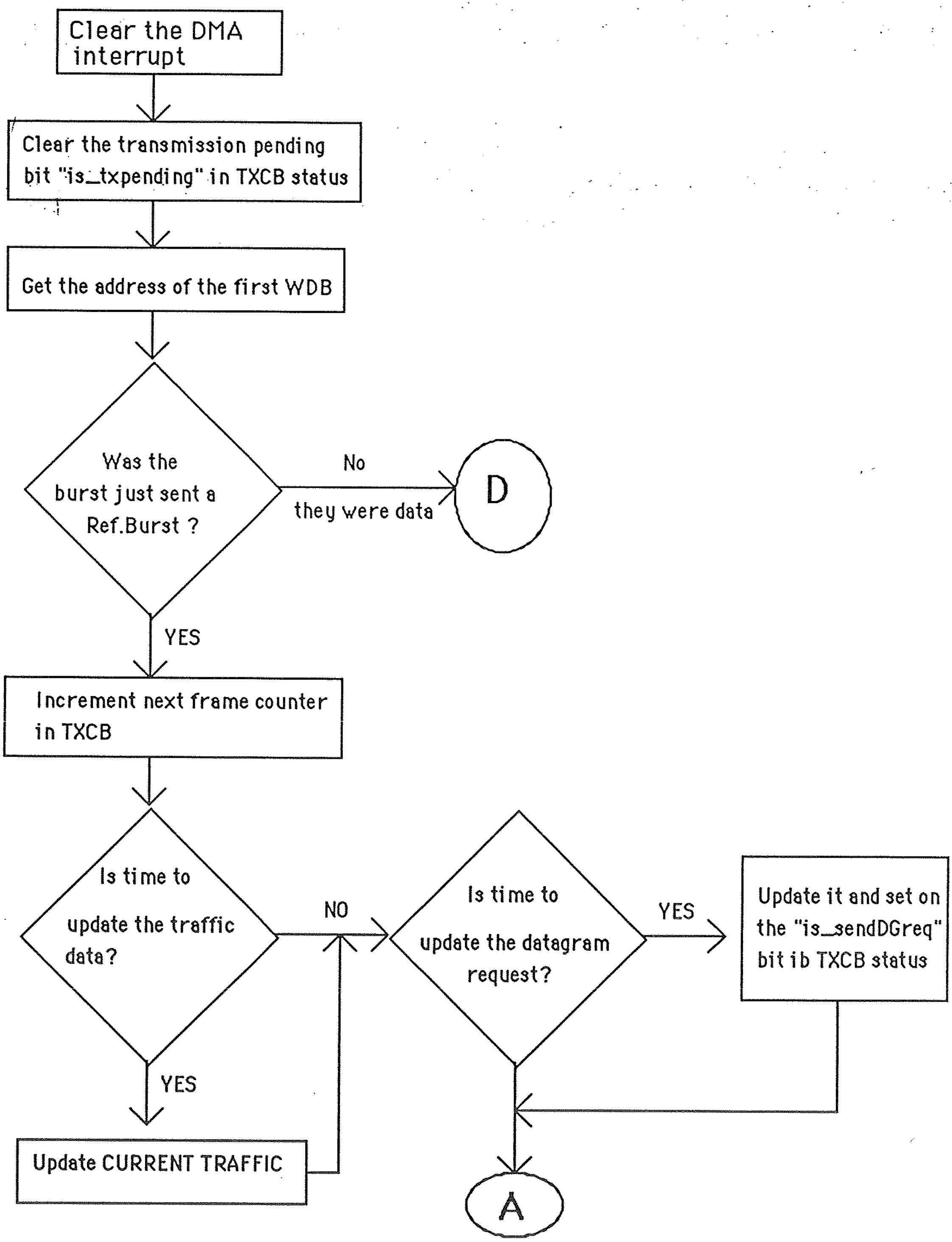
Timer Interrupt: TX_TE routine

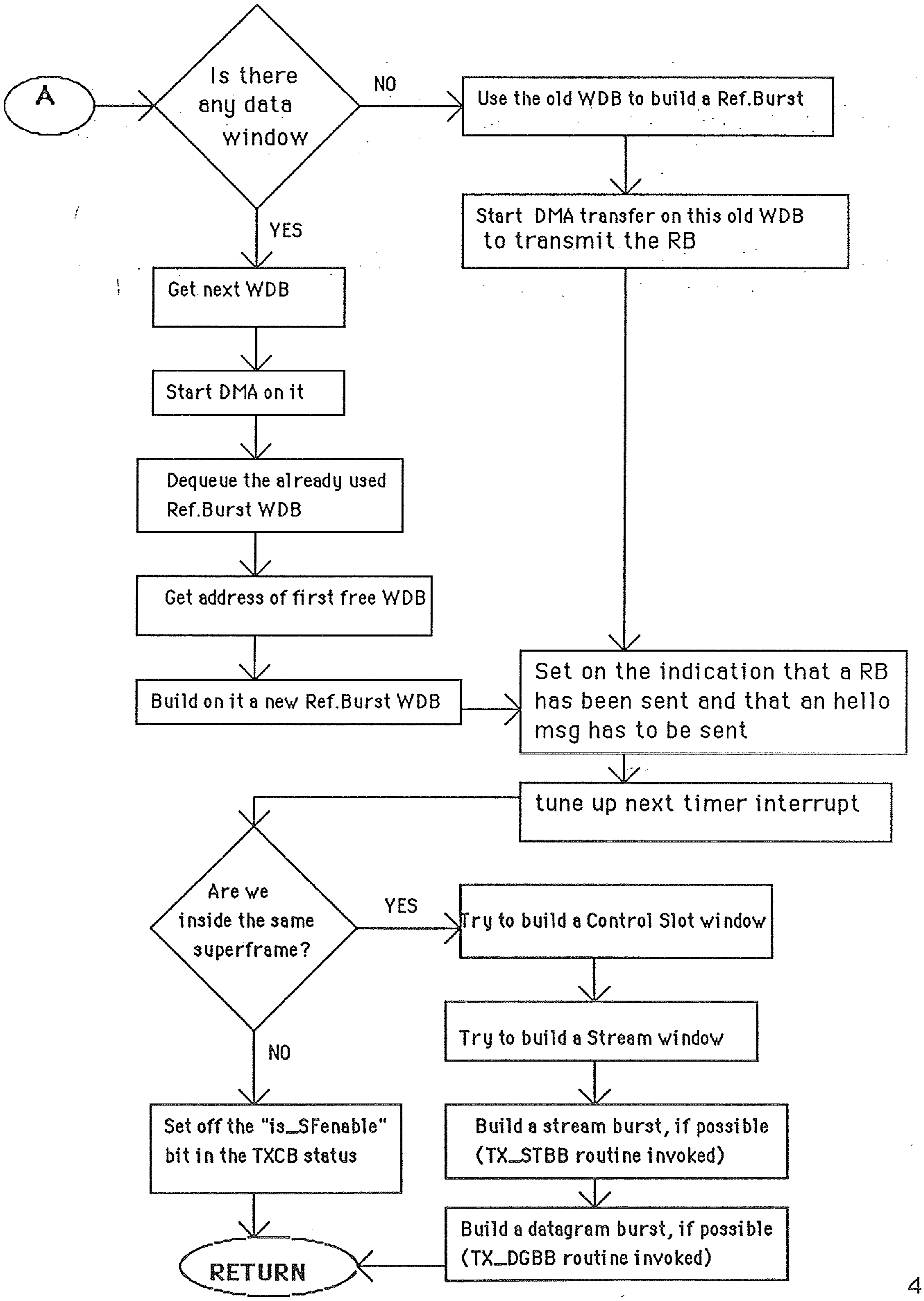


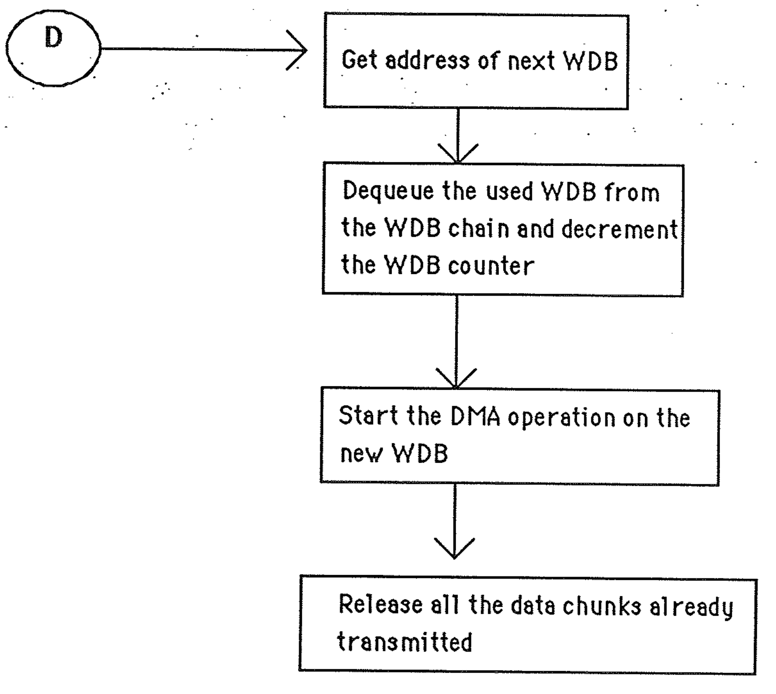
Timer interrupt: TX_CLEAR_WFS routine



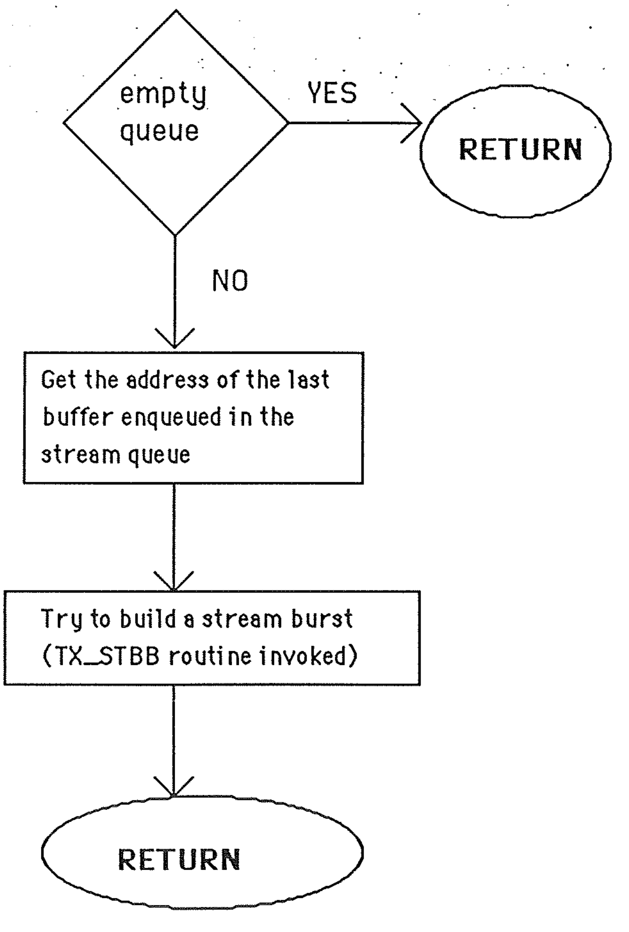
The DMA EOB handling: TX_EOB routine



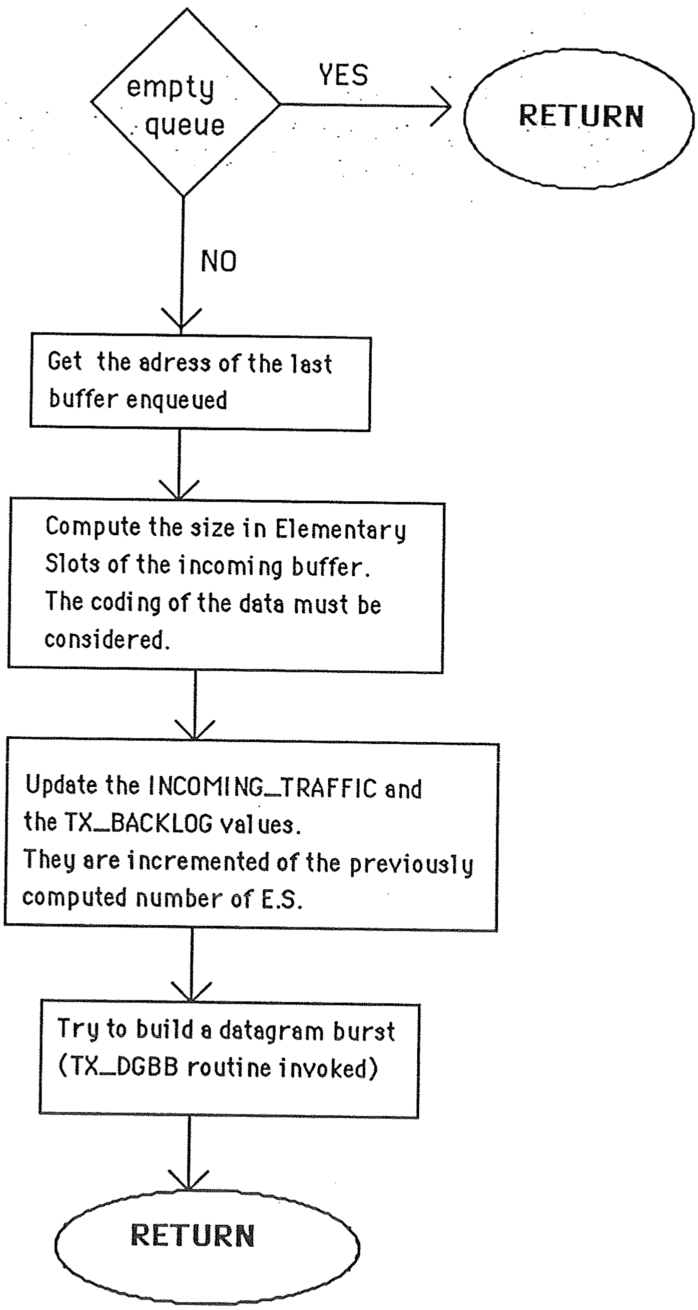

A
Is there any data window
NO
Use the old WDB to build a Ref.Burst
Start DMA transfer on this old WDB to transmit the RB
YES
Get next WDB
Start DMA on it
Dequeue the already used Ref.Burst WDB
Get address of first free WDB
Build on it a new Ref.Burst WDB
Set on the indication that a RB has been sent and that an hello msg has to be sent
tune up next timer interrupt
Are we inside the same superframe?
YES
Try to build a Control Slot window
Try to build a Stream window
NO
Set off the "is_SFenable" bit in the TXCB status
Build a stream burst, if possible (TX_STBB routine invoked)
Build a datagram burst, if possible (TX_DGBB routine invoked)
RETURN


D
Get address of next WDB
Dequeue the used WDB from the WDB chain and decrement the WDB counter
Start the DMA operation on the new WDB
Release all the data chunks already transmitted

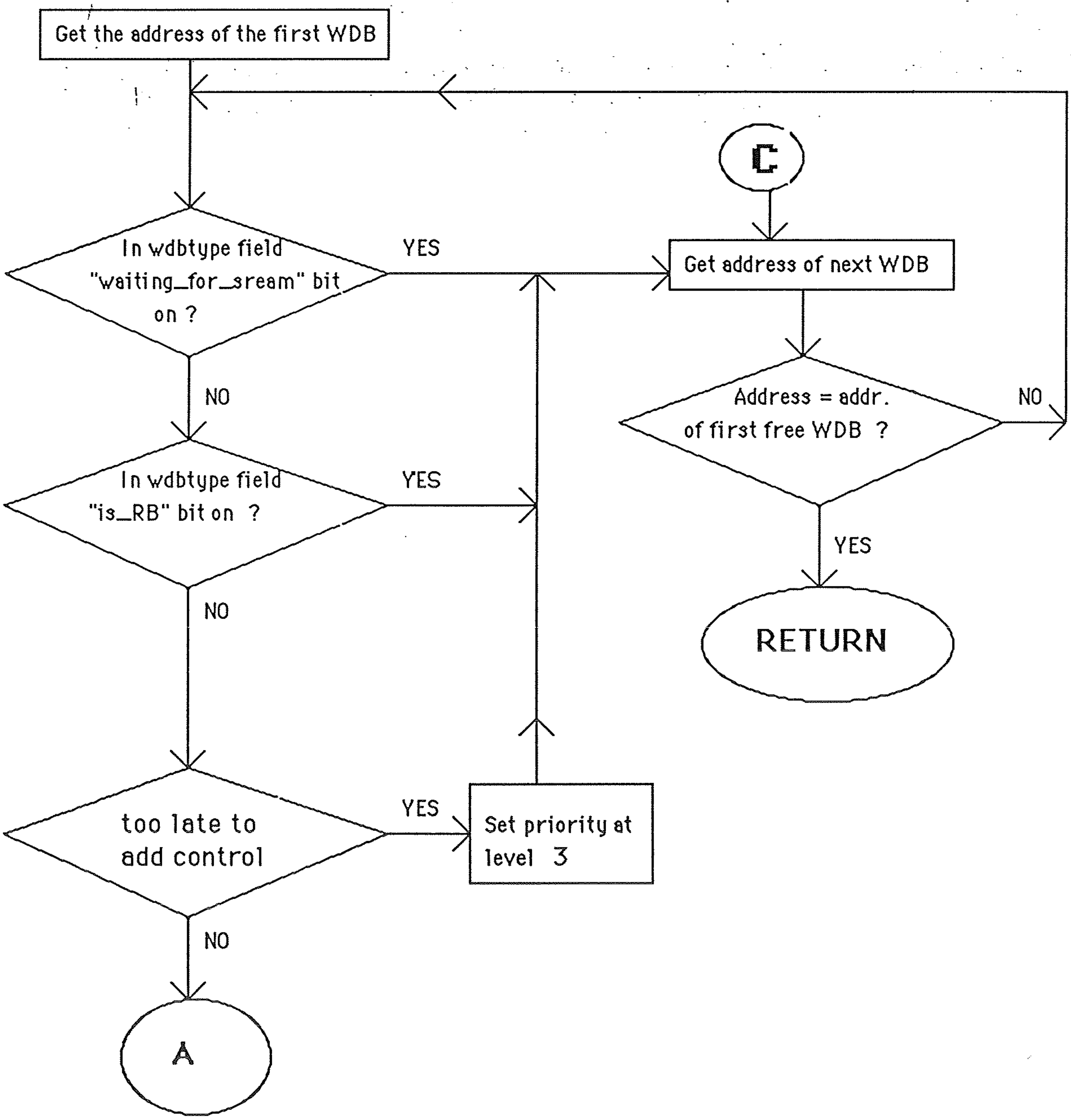
Ring interrupt: TX_S_IN routine

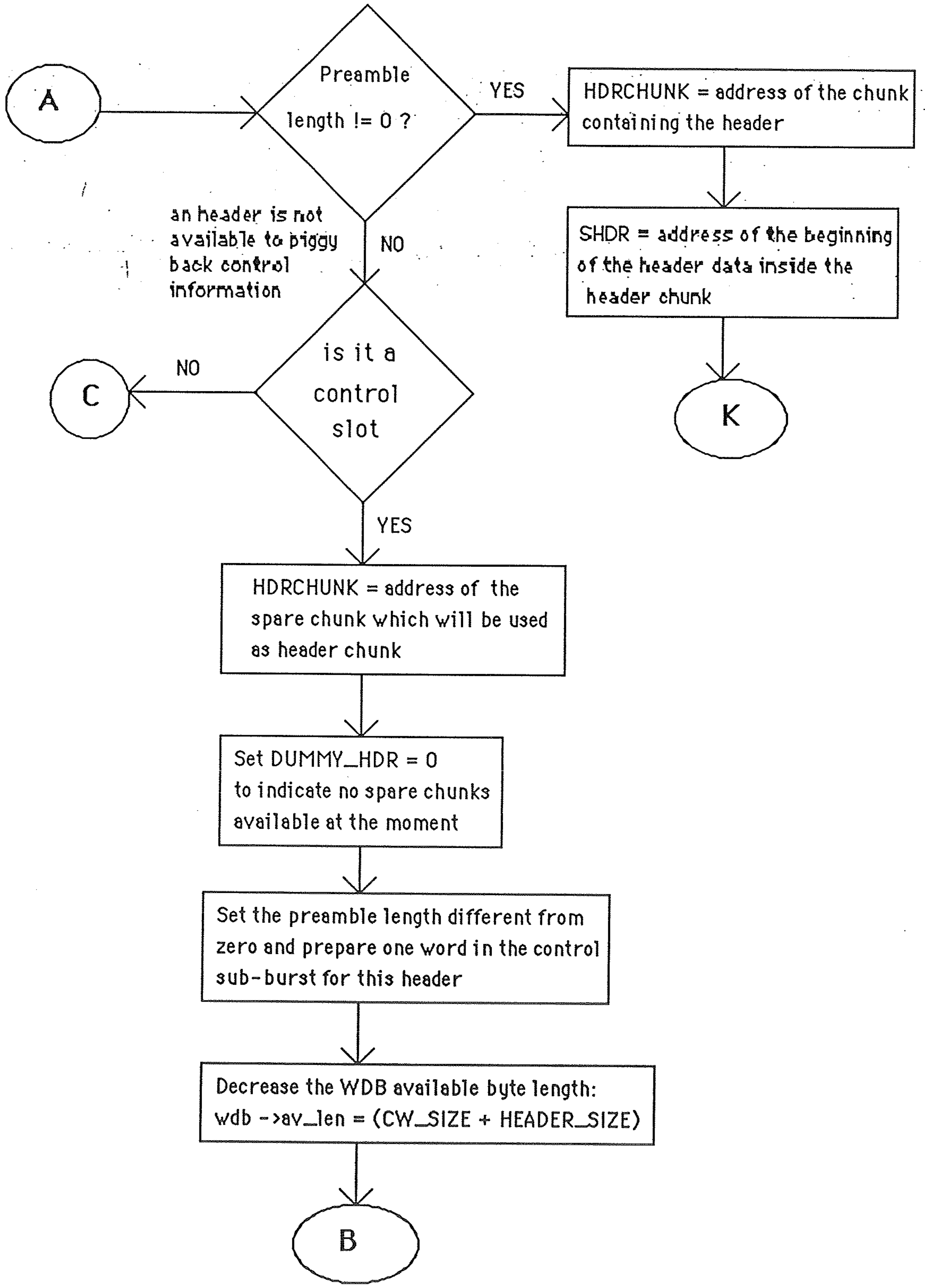


Ring interrupt: TX_I_IN & TX_B_IN routines

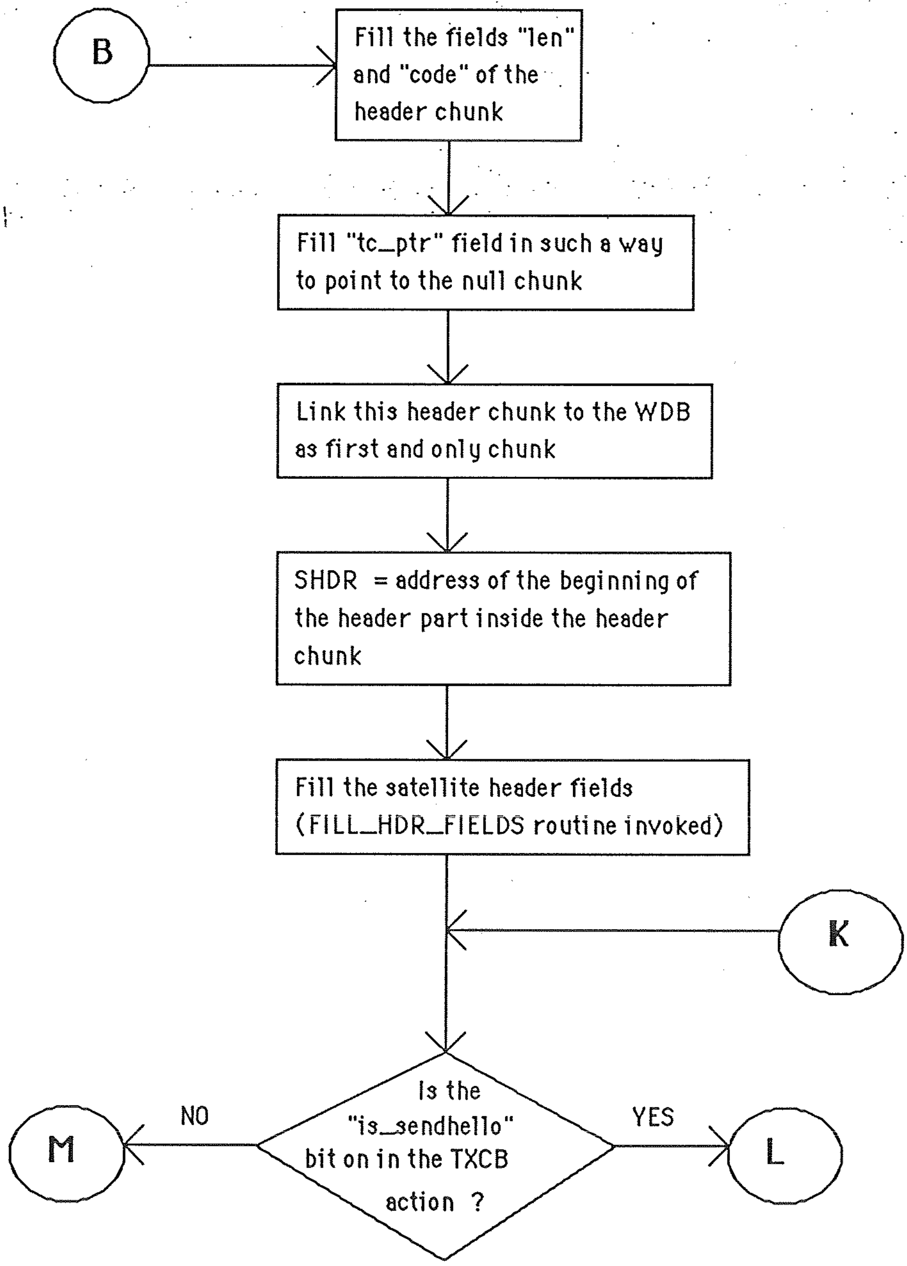


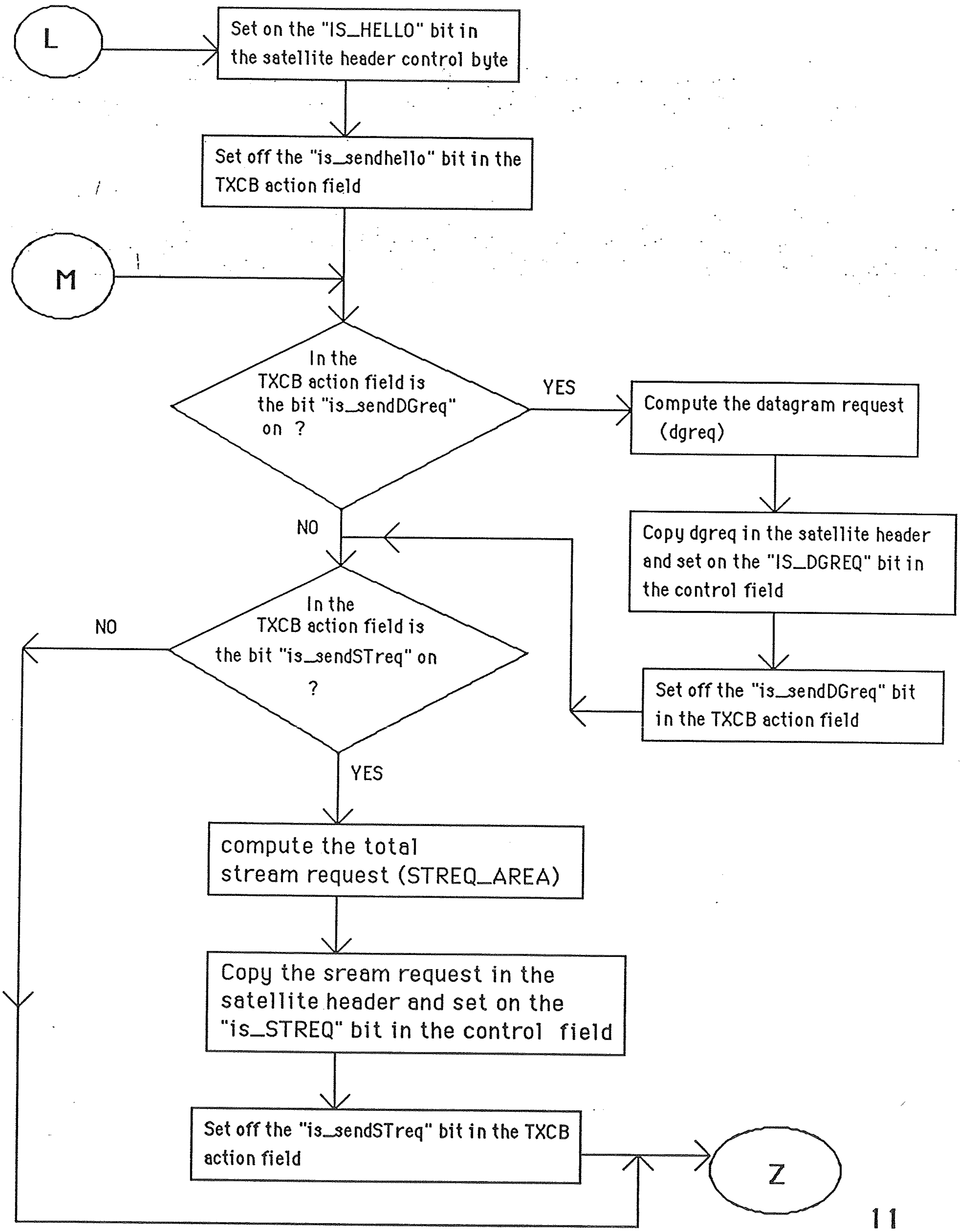
To add control to a burst: TX_AC routine

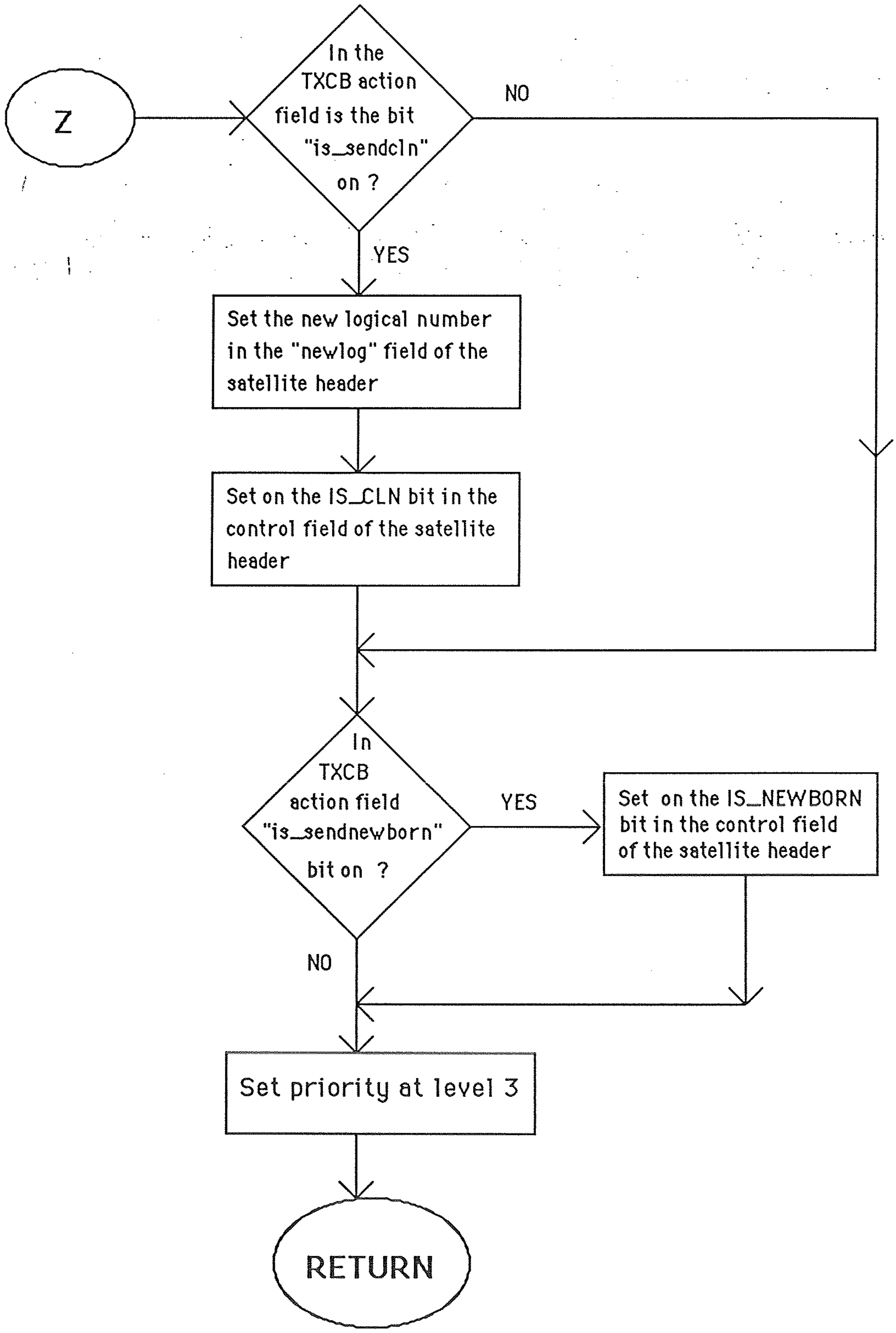




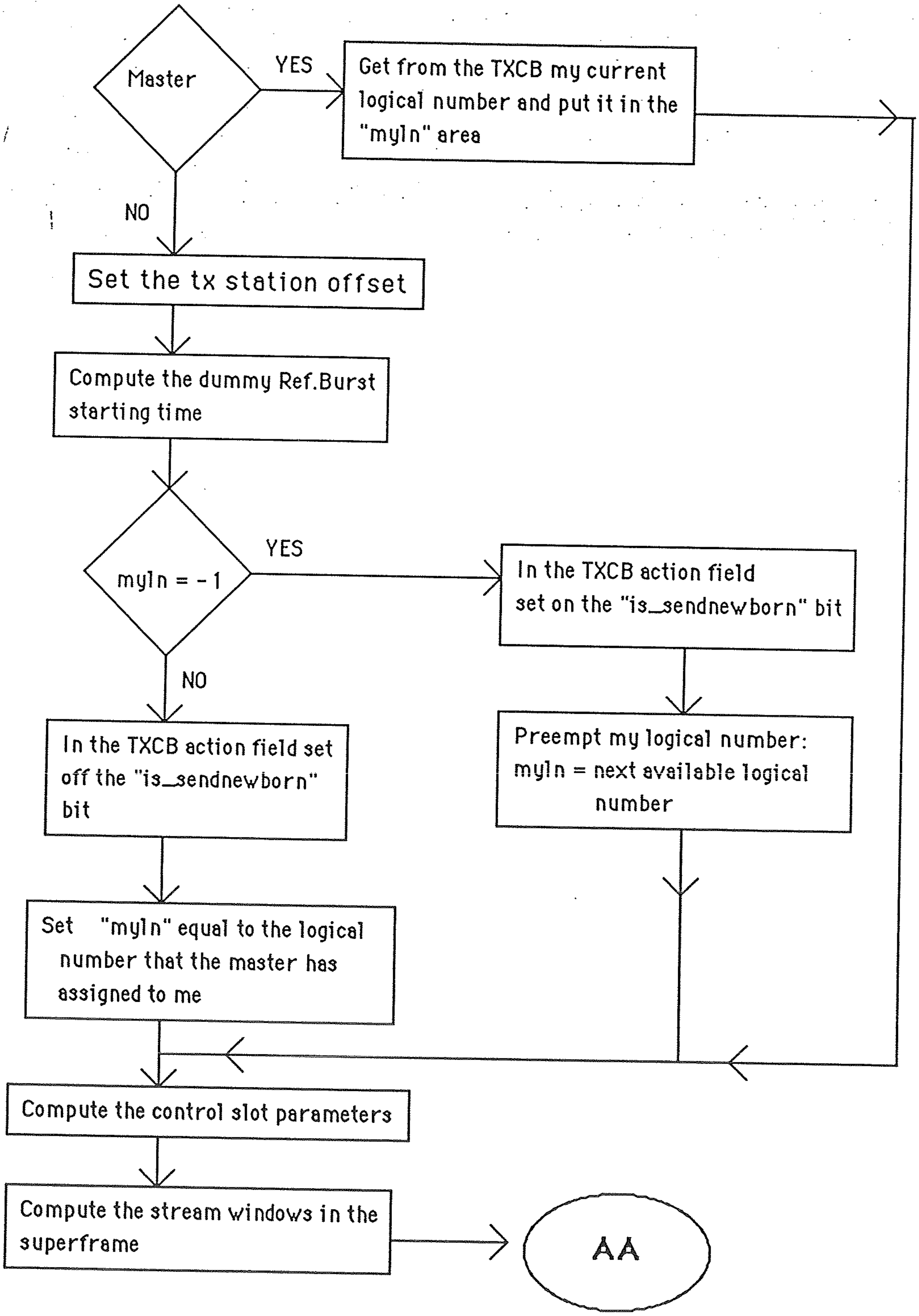
A
Preamble
length != 0 ?
YES
HDRCHUNK = address of the chunk containing the header
SHDR = address of the beginning of the header data inside the header chunk
K
an header is not available to piggy back control information
NO
is it a control slot
NO
C
YES
HDRCHUNK = address of the spare chunk which will be used as header chunk
Set DUMMY_HDR = 0 to indicate no spare chunks available at the moment
Set the preamble length different from zero and prepare one word in the control sub-burst for this header
Decrease the WDB available byte length: wdb ->av_len = (CW_SIZE + HEADER_SIZE)
B

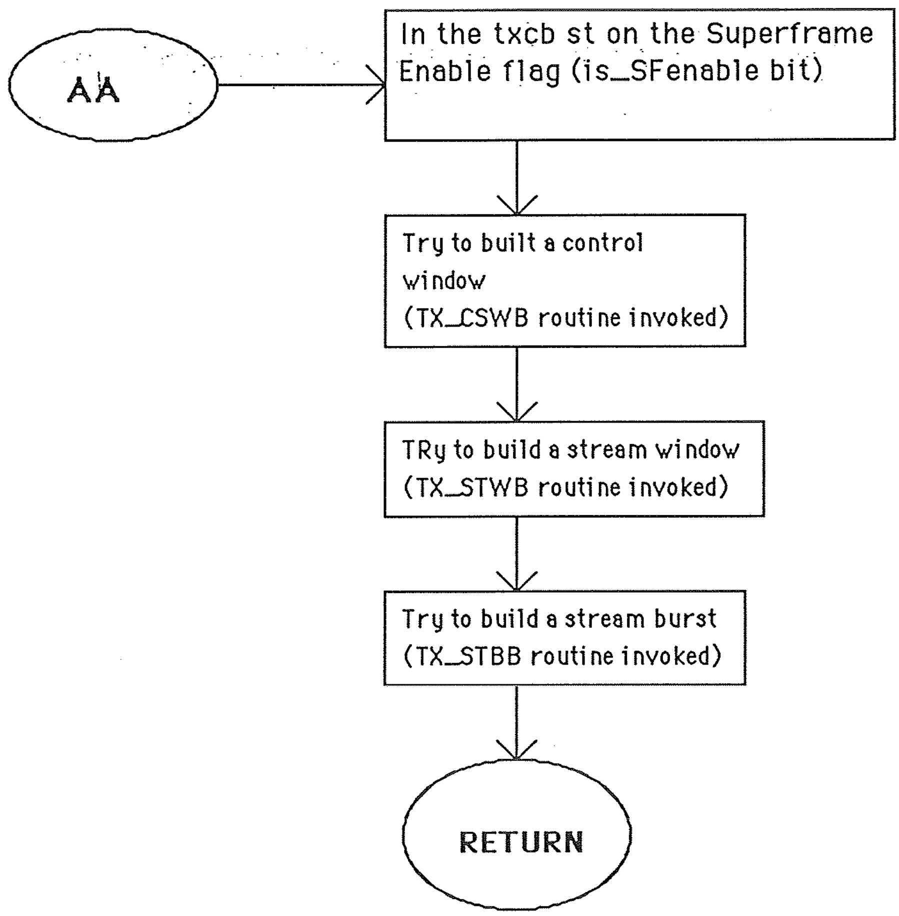

B
Fill the fields "len" and "code" of the header chunk
Fill "tc_ptr" field in such a way to point to the null chunk
Link this header chunk to the WDB as first and only chunk
SHDR = address of the beginning of the header part inside the header chunk
Fill the satellite header fields (FILL_HDR_FIELDS routine invoked)
K
Is the "is_sendhello" bit on in the TXCB action ?
NO
M
YES
L


L
Set on the "IS_HELLO" bit in the satellite header control byte
Set off the "is_sendhello" bit in the TXCB action field
M
In the TXCB action field is the bit "is_sendDGreq" on ?
YES
Compute the datagram request (dgreq)
Copy dgreq in the satellite header and set on the "IS_DGREQ" bit in the control field
Set off the "is_sendDGreq" bit in the TXCB action field
NO
In the TXCB action field is the bit "is_sendSTreq" on ?
NO
YES
compute the total stream request (STREQ_AREA)
Copy the sream request in the satellite header and set on the "is_STREQ" bit in the control field
Set off the "is_sendSTreq" bit in the TXCB action field
Z

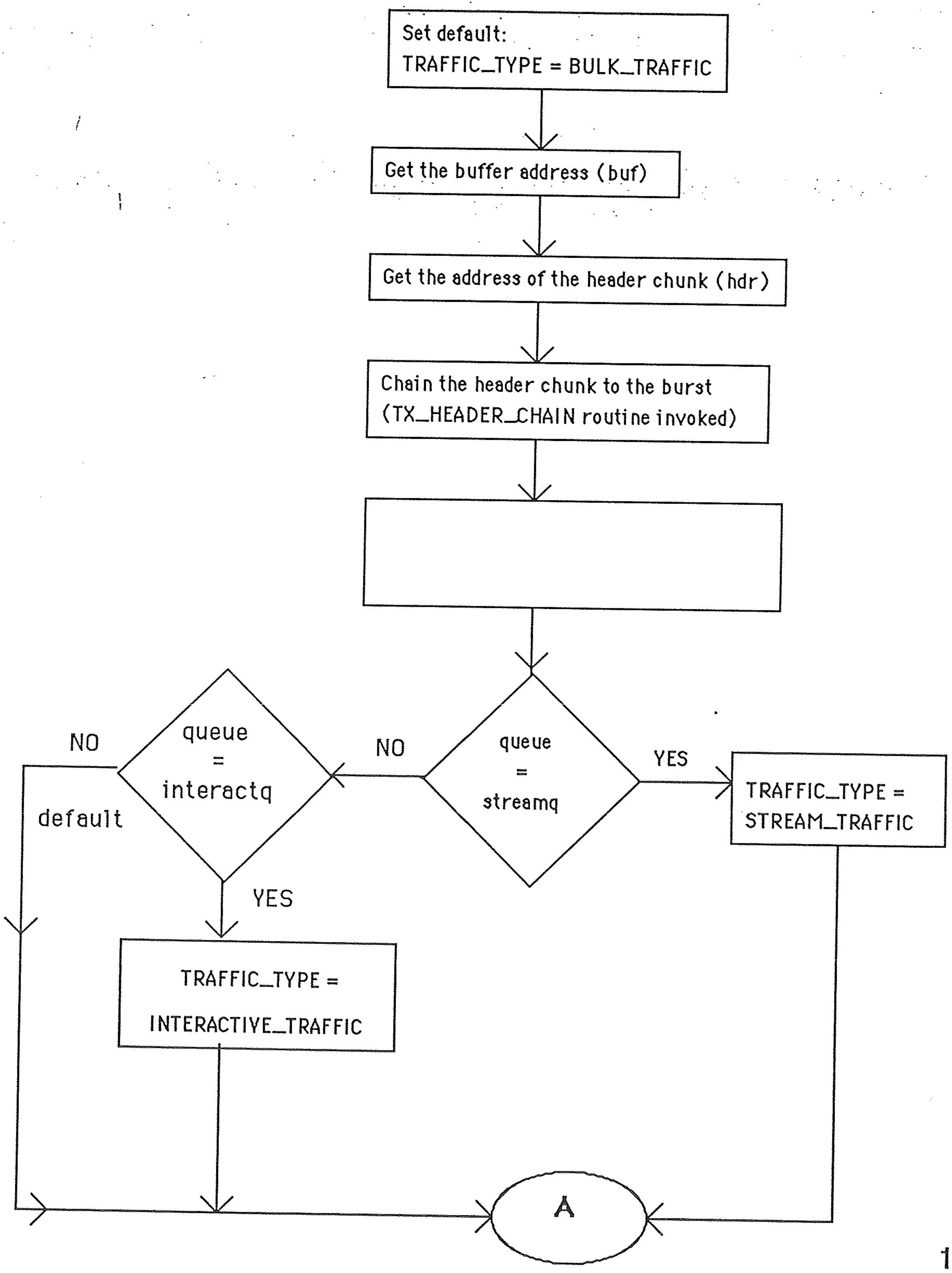

Z
In the TXCB action field is the bit "is_sendcln" on ?
NO
YES
Set the new logical number in the "newlog" field of the satellite header
Set on the IS_CLN bit in the control field of the satellite header
In TXCB action field "is_sendnewborn" bit on ?
YES
Set on the IS_NEWBORN bit in the control field of the satellite header
NO
Set priority at level 3
RETURN

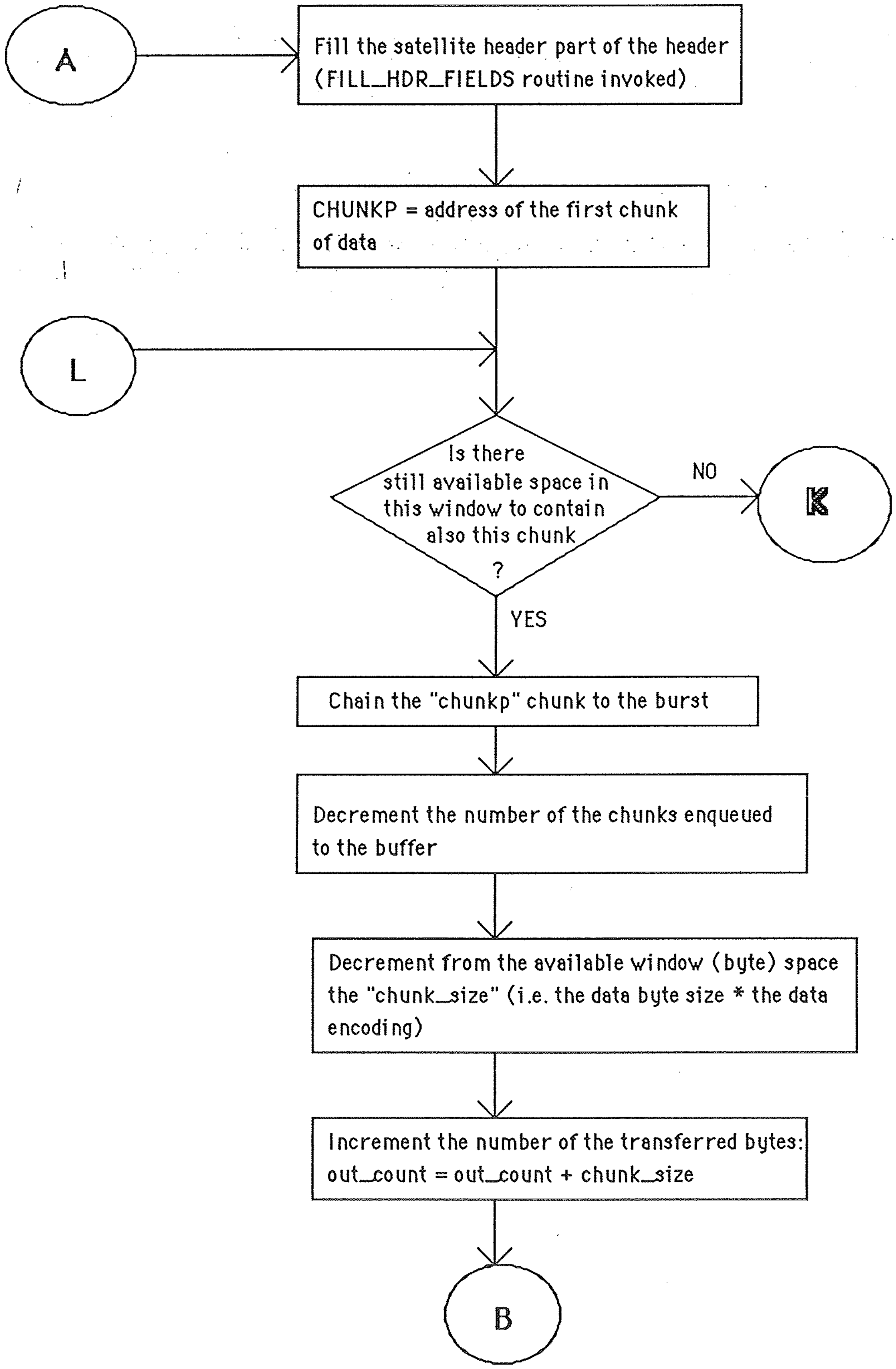
Pio interrupt: TXS_SRB routine

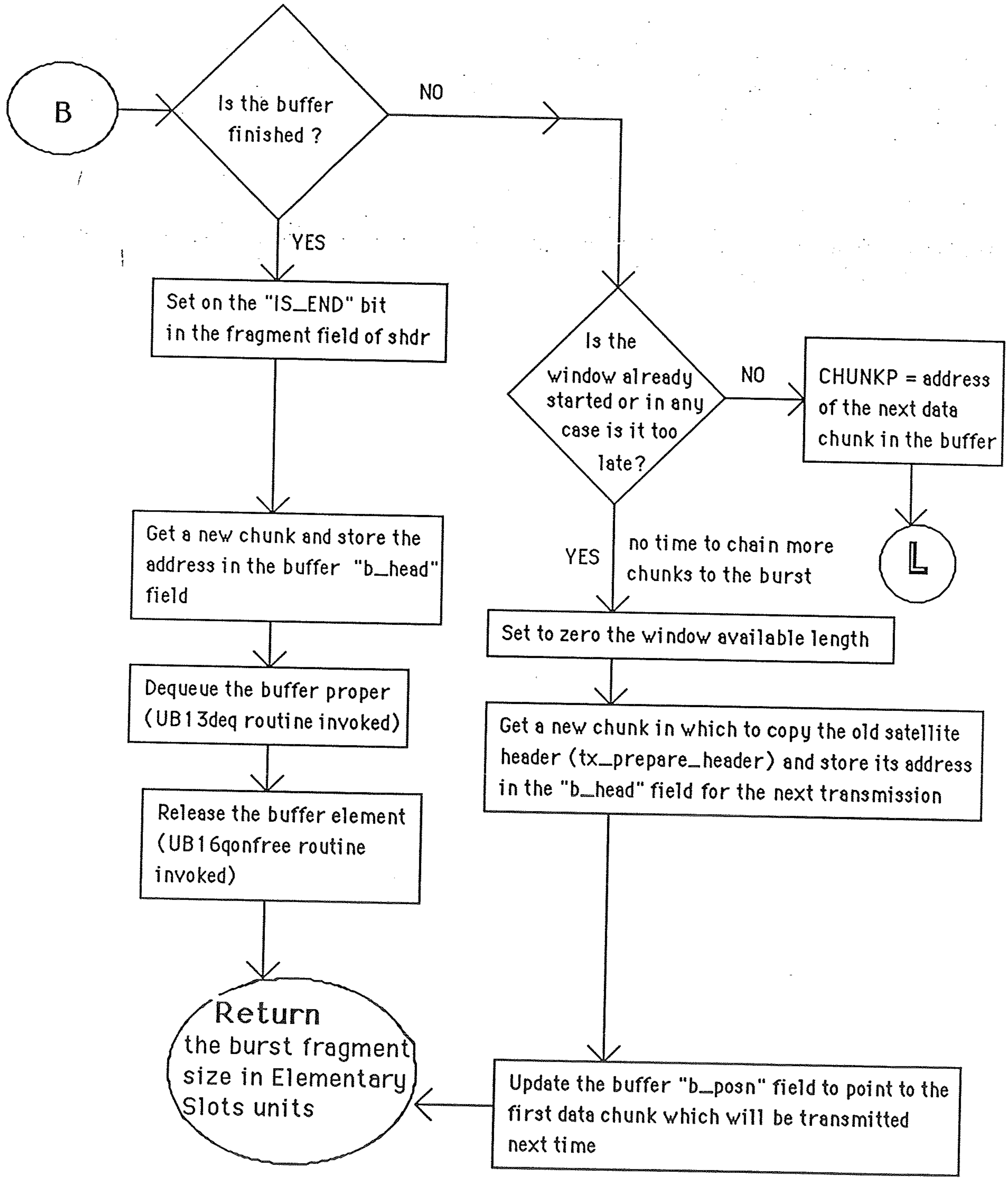


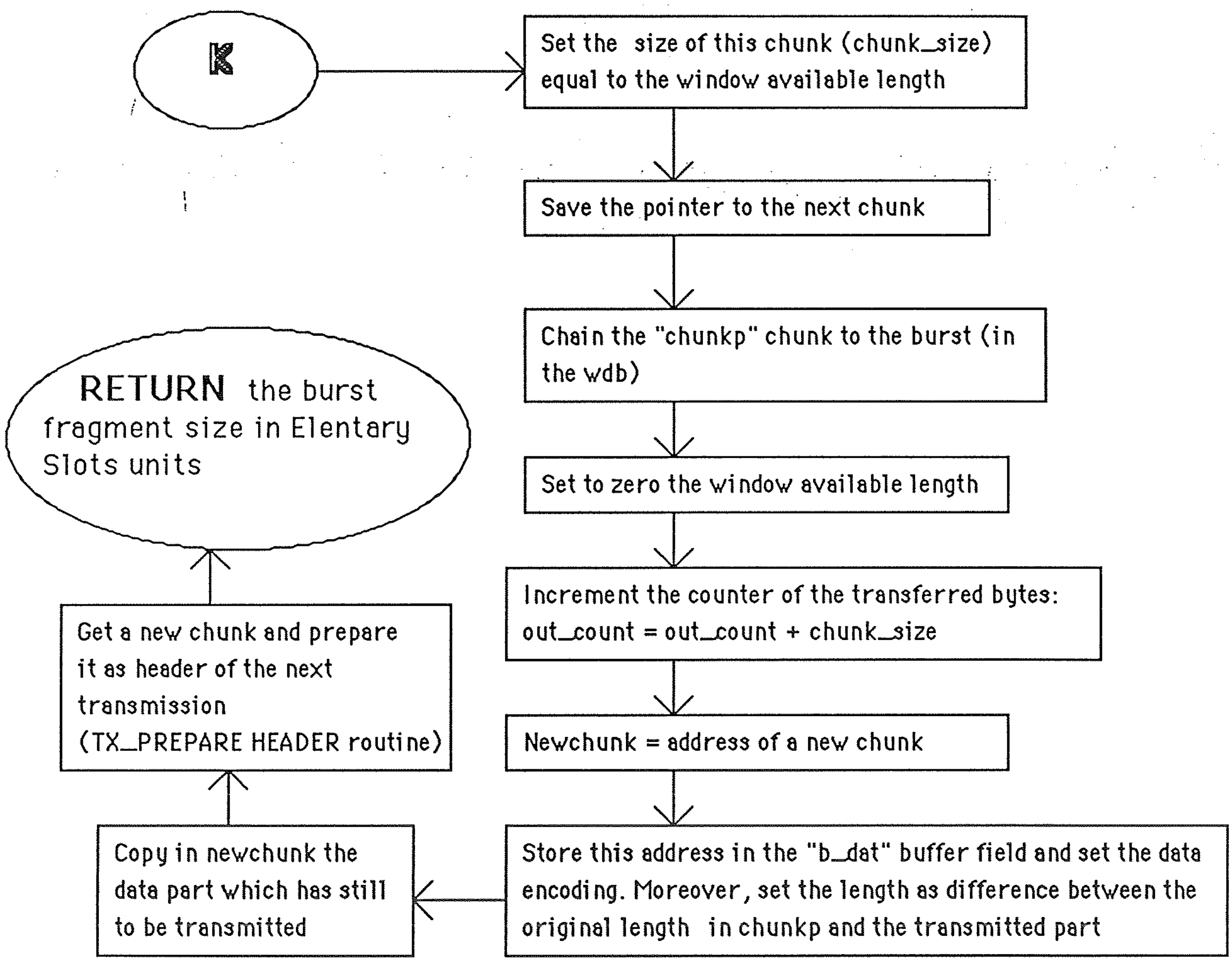

AA
In the txcb st on the Superframe Enable flag (is_SFenable bit)
Try to built a control window (TX_CSWB routine invoked)
TRy to build a stream window (TX_STWB routine invoked)
Try to build a stream burst (TX_STBB routine invoked)
RETURN

To add a fragment: TX_ADD_FRAGMENT routine




A
Fill the satellite header part of the header (FILL_HDR_FIELDS routine invoked)
CHUNKP = address of the first chunk of data
L
Is there still available space in this window to contain also this chunk ?
NO
K
YES
Chain the "chunkp" chunk to the burst
Decrement the number of the chunks enqueued to the buffer
Decrement from the available window (byte) space the "chunk_size" (i.e. the data byte size * the data encoding)
Increment the number of the transferred bytes: out_count = out_count + chunk_size
B


B
Is the buffer finished ?
NO
YES
Set on the "IS_END" bit in the fragment field of shdr
Get a new chunk and store the address in the buffer "b_head" field
Dequeue the buffer proper (UB13deq routine invoked)
Release the buffer element (UB16qonfree routine invoked)
Return the burst fragment size in Elementary Slots units
Is the window already started or in any case is it too late?
NO
CHUNKP = address of the next data chunk in the buffer
L
YES
no time to chain more chunks to the burst
Set to zero the window available length
Get a new chunk in which to copy the old satellite header (tx_prepare_header) and store its address in the "b_head" field for the next transmission
Update the buffer "b_posn" field to point to the first data chunk which will be transmitted next time


K
Set the size of this chunk (chunk_size) equal to the window available length
Save the pointer to the next chunk
Chain the "chunkp" chunk to the burst (in the wdb)
Set to zero the window available length
Increment the counter of the transferred bytes: out_count = out_count + chunk_size
Newchunk = address of a new chunk
Store this address in the "b_dat" buffer field and set the data encoding. Moreover, set the length as difference between the original length in chunkp and the transmitted part
Copy in newchunk the data part which has still to be transmitted
Get a new chunk and prepare it as header of the next transmission (TX_PREPARE HEADER routine)
RETURN the burst fragment size in Elentary Slots units

Section III

Third section: the fixed rate traffic generator

In this section a fixed rate traffic generator is described. It has been written in C language and the software added on the Up process in order to test the access scheme with different traffic conditions.

The fixed rate traffic generator

1. The environment

The term "FODA system" indicates the satellite bridge TDMA controller running the FODA-TDMA satellite access scheme.
The TDMA controller is essentially composed by:

- *a microprocessor MOTOROLA 68000* having the functions of receiver from satellite and running the RX part of the FODA access scheme. It is equiped to send data to the attached local area network (DOWN machine);

- *a microprocessor MOTOROLA 68000* having the functions of sender to the satellite and running the TX part of the FODA access scheme. It is equiped to receive data from the attached local area network (UP machine). Up and Down machines communicate via PIA interface.

- *a variable bit rate modem;*

- *a variable coding rate codec;*

- *the FODA-TDMA* (Fifo ordered demand assignment-TDMA) satellite access scheme.

The system is not restricted to fixed point-to-point links, but supports the special requirements of distributed computing and information dissemination.
A group of users shares the satellite channel in time division multiple access mode on a demand basis: that means that only when packets are to be sent transmission time slots are actually allocated to a station. The access to a satellite channel is given by the satellite bridge which receives packets from individual users by a LAN and transfers them over the satellite channel to the addressed user or host.

The system not only provides computer data transmission facilities but also voice or slow scan or compressed video communications.
These different services have their own "quality of service" parameters:

- required bit rate
- maximum tolerable bit error rate

- priority of the service
- burstiness of the data.

The FODA access scheme has been designed and developed just to support "stream" traffic (voice and immages) and "datagram" traffic (bulk and interactive data), dynamically and according to the traffic requirements.
It is based on reservation of the bandwidth and the time is divided into slots in which the various stations alternate their use of the entire capacity of the channel. The assignment of the time slots for stream and for datagram is made dynamically upon demand of the satellite channel users (earth stations).

The assignment algorithm is different for stream and for datagram slots.
The TDMA controller receives data from the users attached to the interconnected LANs, in order to send them to the satellite (Up machine).
The data received from the satellite are then sent to the target users, transferring them via the attached LAN (Down machine).

Next figure shows the FODA system.

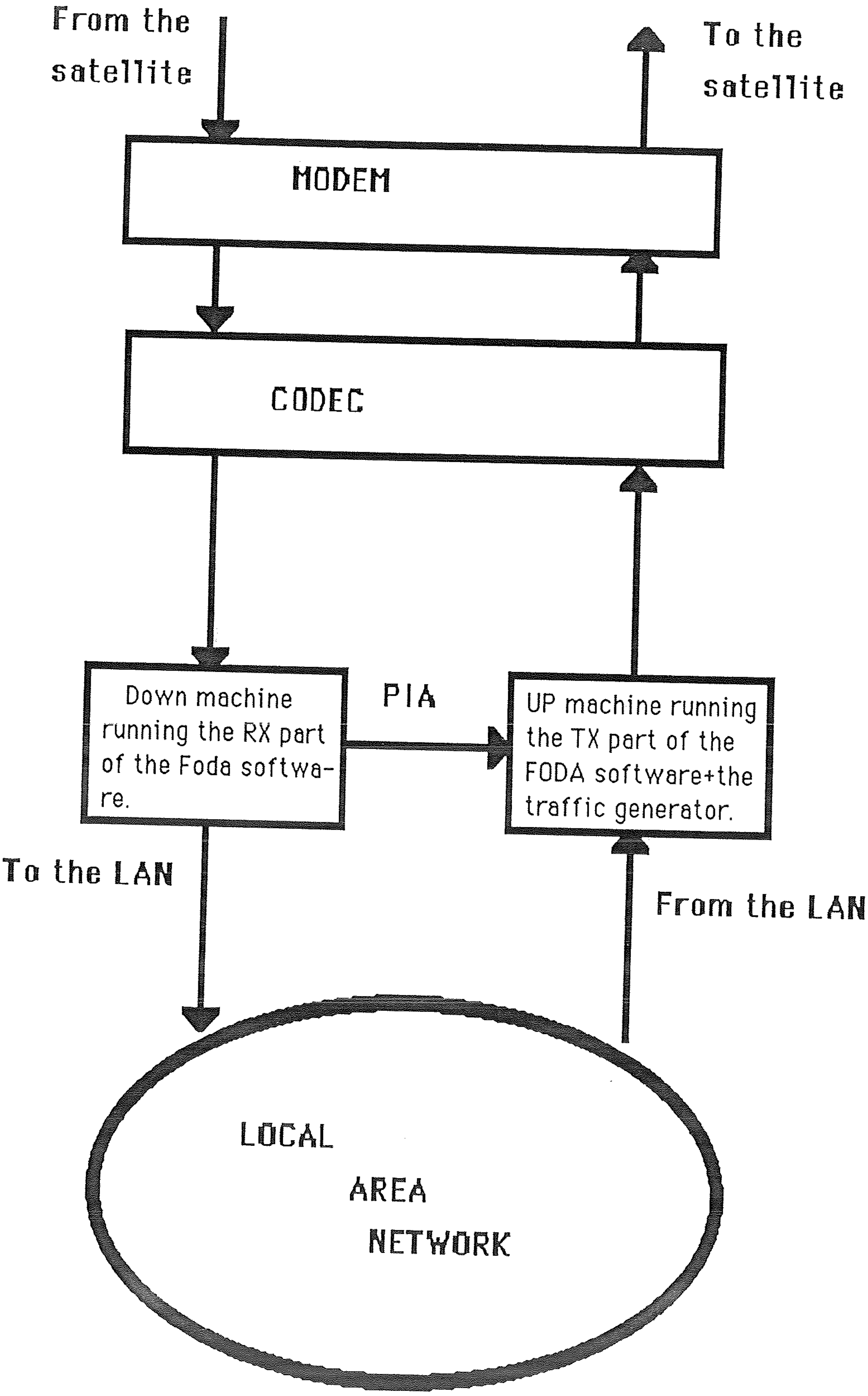
The data entering into the Up machine from the LAN are organized in 3 queues, here listed with increasing priority of transfer:

- *bulkq* for the bulk data,

- *interactq* for the interactive data,

- *streamq* for the stream data.

As it is very difficult to test a so complex system in a real environment, a traffic generator has been created inside the Up machine to test the performances of the system in the worst situations of heavy traffic load.


From the satellite
To the satellite
MODEM
CODEC
Down machine running the RX part of the Foda softwa-re.
PIA
UP machine running the TX part of the FODA software+the traffic generator.
To the LAN
From the LAN
LOCAL
AREA
NETWORK

2. Traffic generator description

A fixed rate traffic generator is here described. The fixed rate is realized in such a way that, at each timer interrupt on the Up machine, the selected quantity of data is generated, simulating its incoming from the attached local area network.

The software of the traffic generator has been written in C language and it is part of the software running on the Up machine.
It consists of the routines contained in the file *TXFRTG . C* plus some instructions added in the *UBU100 . C* file (routine UB101userhand) and in the *TXTIM . C* (routine tx_te) file.

At the initialization phase, the operator is requested whether or not the traffic generator facility must be started.
If yes, the operator is requested to specify a set of parameters on the base of which the traffic generator is able to compute the global traffic load requested in each frame and its distribution among stream, interactive and bulk data.
Then, for each type of traffic, the buffer of the correct length is generated and enqueued in the relative queue, simulating the arrival of the data from the LAN.

The command "**t**" allows the real start of the traffic generation.

A special command "f" has been introduced to stop the traffic generator. This command allows an eventual restarting, via another "t" command" of the traffic generator running with the last specified parameters.

Another new command "**a**" allows a restarting of the traffic generator with new parameters (the operator is again requested to specify the parameters for the traffic generator). The "f" command is included in it.

The software essentially consists in the following set of routines (contained in the TXFRTG . C file which uses the TXFRTG . H header file):

- *TG_PREPARE :* it requests the user to specify the parameters necessary to the traffic generator. The routines TG_INI and TG_COMPUTE are invoked by this routine.

- *TG_INI :* it initializes the fields of the frtg area;

- *TG_COMPUTE :* on the base of the values of the parameters specified by the operator, it computes which is the global traffic load and its distribution among stream, interactive and bulk data in each frame.

The three previous routines are executed at task level, being invoked by the "upuser" task (UB101userhand routine), normally used to enter data from the operator console.

- *TG_BURSTS :* on the base of the resulting traffic load distribution, it prepares, at each TX timer interrupt, the required data bursts.
This routine is executed at the TX timer interrupt level, being invoked by the tx_te routine.
This routine is executed when all the other normal actions (which have to be done at the timer interrupt) have already been done. As soon as this routine is entered, the priority is lowered to zero, just to allow to be interrupted by another timer interrupt. This will happen only if the execution of the routine is longer than the time in between two consecutive timer interrupts.

The data generated by the internal traffic generator are flagged as TEST_DATA (hexadecimal 0xffff) using the "h_length" field of the relative satellite header, in order to be discarded when received by the Down process.

During the sending of the data generated by the traffic generator, the Delay Measurement Packet (DMP) may be sent, just once via the operator command "**m**" or every a fixed number of frames specified by the operator on issuing the "n" command.

The DMP is used to measure the delay of the FODA system on the base of the traffic load. The delay is displayed (in msec) on the output terminal.

3. The used areas

frtg is an area containing the following information:

```
15+                               0-
+---------------------------------+
| channel throughput in percentage |
| of the bit rate                  | thpt_perc
+---------------------------------+
| stream data as percentage of the |
| channel                          | str_perc
+---------------------------------+
| bulk data as percentage of the   |
|  channel                         | bulk_perc
+---------------------------------+
| interactive data as percentage of |
|  the channel                     | int_perc
+---------------------------------+
|    stream data pattern           |
|                                  | tgs_patt
+---------------------------------+
|     interactive data pattern     |
|                                  | tgi_patt
+---------------------------------+
|     bulk data pattern            |
|                                  | tgb_patt
+---------------------------------+
|     stream data coding           |
|                                  | tgs_code
+---------------------------------+
|     interactive data coding      |
|                                  | tgi_code
+---------------------------------+
|     bulk data coding             |
|                                  | tgb_code
+---------------------------------+
| traffic generator bit status     |
|                                  | tg_status
+---------------------------------+
| delay measurement packet bit     |
|        status                    | dmp_status
+---------------------------------+
```

The *tg_status* field is so compound:

```
+15                                                  3   2   1   0-
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   | D | C | B | A |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

A : is_TGSTARTED (bit 0; value = 1); traffic generator is started;

B : is_TGSELECTED (bit 1; value = 2); traffic generator facility selected;

C : is_TGROUTINE (bit 2; value = 4); at the TX timer interrupt, the routine to create the bursts (TG_BURSTS) must be invoked.

D : is_TGRESTARTABLE (bit 3; value = 8); the traffic generator can be restarted.

The *dmp_status* field is so compound:

```
15+                                                          0-
+-------------------------------------------------------------+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | A |
+-------------------------------------------------------------+
```

A : is_DMPSEND (bit 0; value = 1); send a delay measurement packet every DMP_INTERVAL frames, as defined by the operator entering the "n" command.

Other used areas are:

tg_bchunks : 8 bits; number of chunks constituting the bulk buffer;

tg_ichunks : 8 bits; number of chunks constituting the interactive buffer;

tg_schunks : 8 bits; number of chunks constituting the stream buffer;

tg_bel : 32 bits; address of the buffer element for the bulk data;

tg_iel : 32 bits; address of the buffer element for the interactive data;

tg_sel : 32 bits; address of the buffer element for the stream data;

tg_bwlength : 16 bits; number of 16-bits words constituting each bulk chunk;

tg_iwlength : 16 bits; number of 16-bits words constituting each interactive chunk;

tg_swlength : 16 bits; number of 16-bits words constituting each stream chunk;

FRTG_STREQ : 16 bits; it contains the number of requested stream channels.

3.1 Default values

In the TG_INI routine, the fields of the *frtg* area are initialized with the following values:

```
frtg . thpt_perc    =   0
frtg . str_perc     =   0
frtg . bulk_perc    =   0
frtg . int_perc     =   0
frtg . tgs_patt     =   0xAAAA
frtg . tgb_patt     =   0xBBBB
frtg . tgi_patt     =   0xCCCC
frtg . tgs_code     =   0
frtg . tgb_code     =   4
frtg . tgi_code     =   4
STREQ_AREA          =   0
```

4. Specification of the input parameters and their use

If the traffic generator facility is selected, the operator is requested to specify the:

- stream traffic as percentage of the channel,

- bulk traffic as percentage of the channel,

- interactive traffic as percentage of the channel.

At this point, the required throughput is computed as sum of the required stream + bulk + interactive percentages and it is displayed on the monitor.

If the sum of the entered percentages is greater than 100, the *frtg* area is re-initialized and the operator is requested again to enter the correct percentages.

The input specifications continue with the requests of the:

- coding of the stream data;

- coding of the interactive data;

- coding of the bulk data.

For these last three parameters the default is displayed, which is assumed if a carriage return is entered.

If the word pattern of the "stream" or "bulk" or "interactive" chunks has to be changed, the operator must use the ope command to open the **frtg** area (the address is in the TXSYS . MAP file) and change the required word pattern(s) before to issue the **go 8000** command to start the UP process.

On the base of the specified values, the following is computed (TG_COMPUTE routine) for each frame:

- how many bytes have to be transmitted for each type of traffic.
  The data coding is considered in this computation:

$$\text{bytes} = \frac{\text{data percentage} * \text{speed} * 1024 \text{ Mbit/sec}}{100 * 8 * \text{numb. of frames in 1 sec} * \text{coding}};$$

- how many chunks constitute the buffer, for each type of traffic:

$$\text{chunks} = \frac{\text{bytes}}{\text{chunk size in bytes}} \text{ rounded to next integer;}$$

- how many words constitute each chunk, for each type of traffic:

$$\text{words} = \frac{\text{bytes}}{\text{chunks} * 2} \text{ rounded to next integer;}$$

- the required stream channels (on 4 frames):

$$\text{channels} = \frac{\text{stream bytes} * 4}{\text{stream channel byte size}} \text{ rounded to next integer;}$$

5. Differences with the burst generator

In Appendix A a very simple *burst* generator is described.

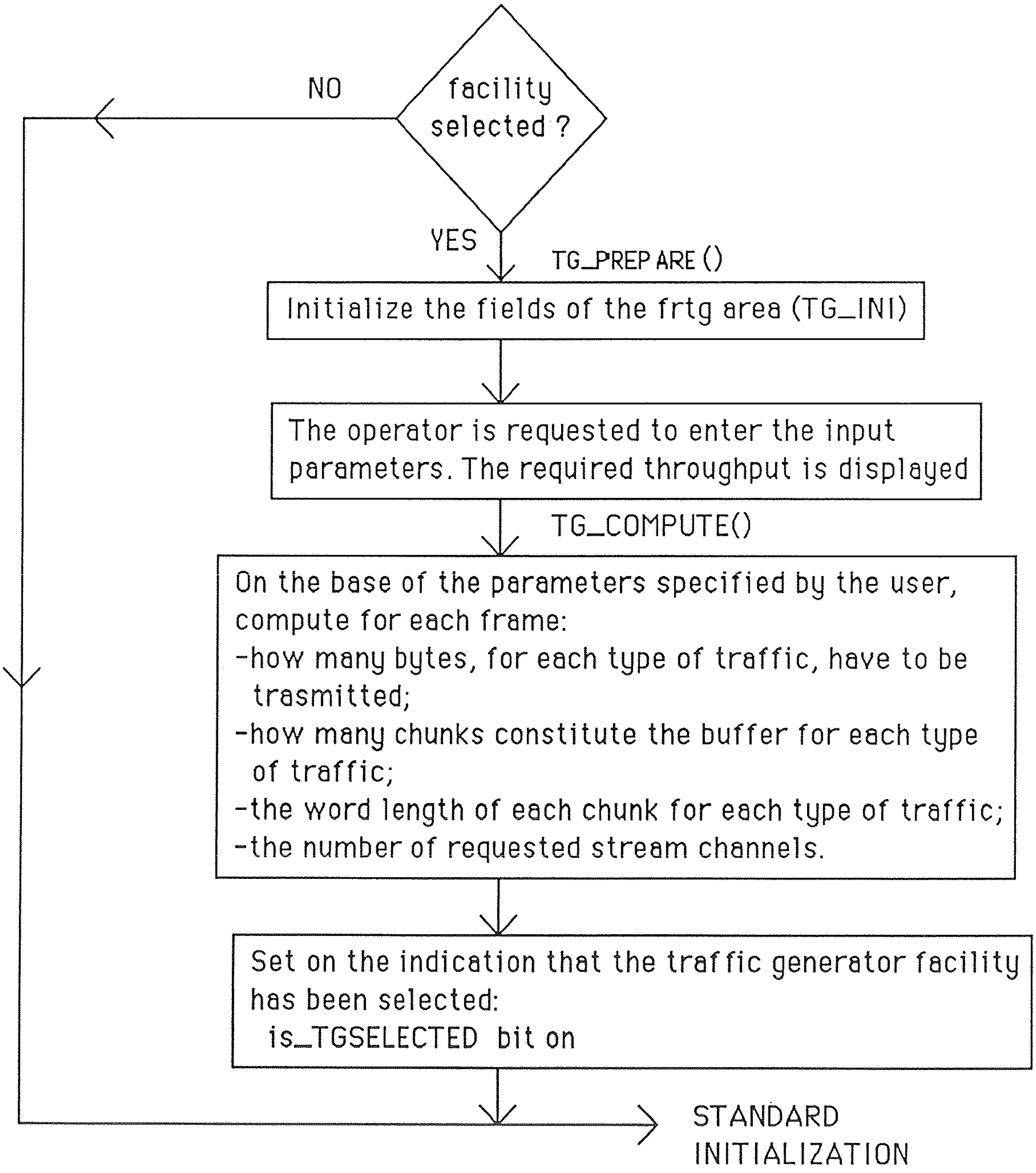
Also in the *traffic* generator case, the selection of the facility is entered via the UB101userhand routine and the user is requested to specify the parameters for the traffic load.

While the burst generator is based on that each time the operator enters a particular command a burst is generated and entered into the system via enqueuing the buffer in the appropriate queue, the fixed rate traffic generator, once specified the traffic load characteristics, enqueues at each TX timer interrupt the required buffers in the correct queues without any other command by the operator.

In the traffic generator case, the traffic generation can be started and stopped in any moment (sequence of "t" and "f" commands) and also a new generation can be dynamically started with new parameters ("a" command).
Moreover, also the commands of the *burst* generator can be in any case used. Very important is the use of the "m" and the "n" commands, used to create and to enter into the system the delay measurement packet in order to measure the delay of the system.

6. The diagrams

6.1 Traffic generator facility selection

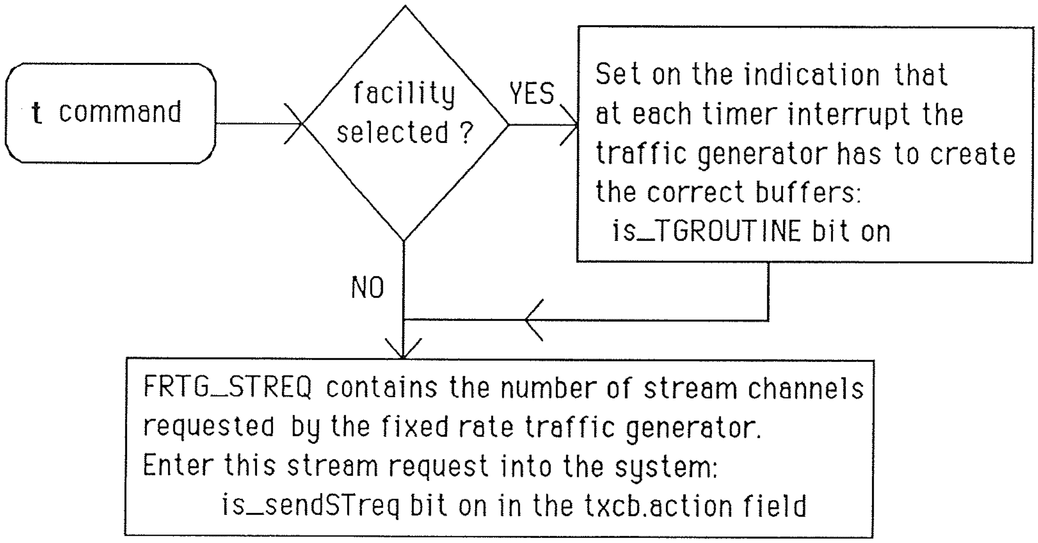


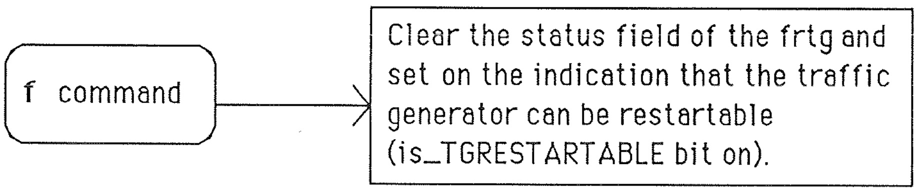
6.2 User Commands

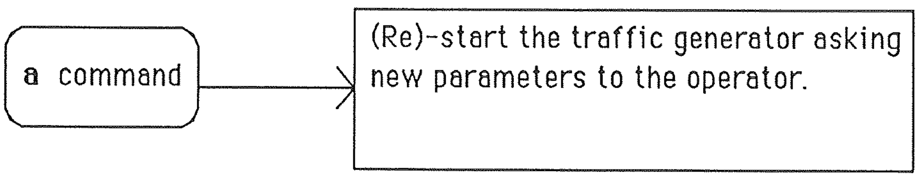
When everything is ready to create the buffers, **nothing will happen till that a "t" command is issued.** This command can be used also by the burst generator to enter the system the stream channel requests. If the traffic generator facility has been selected, the "t" command acts to indicate that, at each tx timer interrupt, some buffers have to be created. Reassuming, **to really start** the traffic generator, the **"t"** command must be issued.

When the operator wants **to stop** the traffic generator, the **"f"** command must be entered. In any case, it allows an eventual restarting of the traffic generation, with the same last specified values, by means of another "t" command.

If the operator wants **to restart** the traffic generator with new parameters **or** simply he wants **to modify** the input parameters, the **"a"** command must be issued. Better to use after a "f" command".
If at the beginning the operator replied "n" (no) to the request whether or not the traffic generator facility was required, he can modify his idea entering the "a" command.


t command
facility selected ?
YES
Set on the indication that at each timer interrupt the traffic generator has to create the correct buffers:
is_TGROUTINE bit on
NO
FRTG_STREQ contains the number of stream channels requested by the fixed rate traffic generator.
Enter this stream request into the system:
is_sendSTreq bit on in the txcb.action field


f command
Clear the status field of the frtg and set on the indication that the traffic generator can be restartable (is_TGRESTARTABLE bit on).


a command
(Re)-start the traffic generator asking new parameters to the operator.

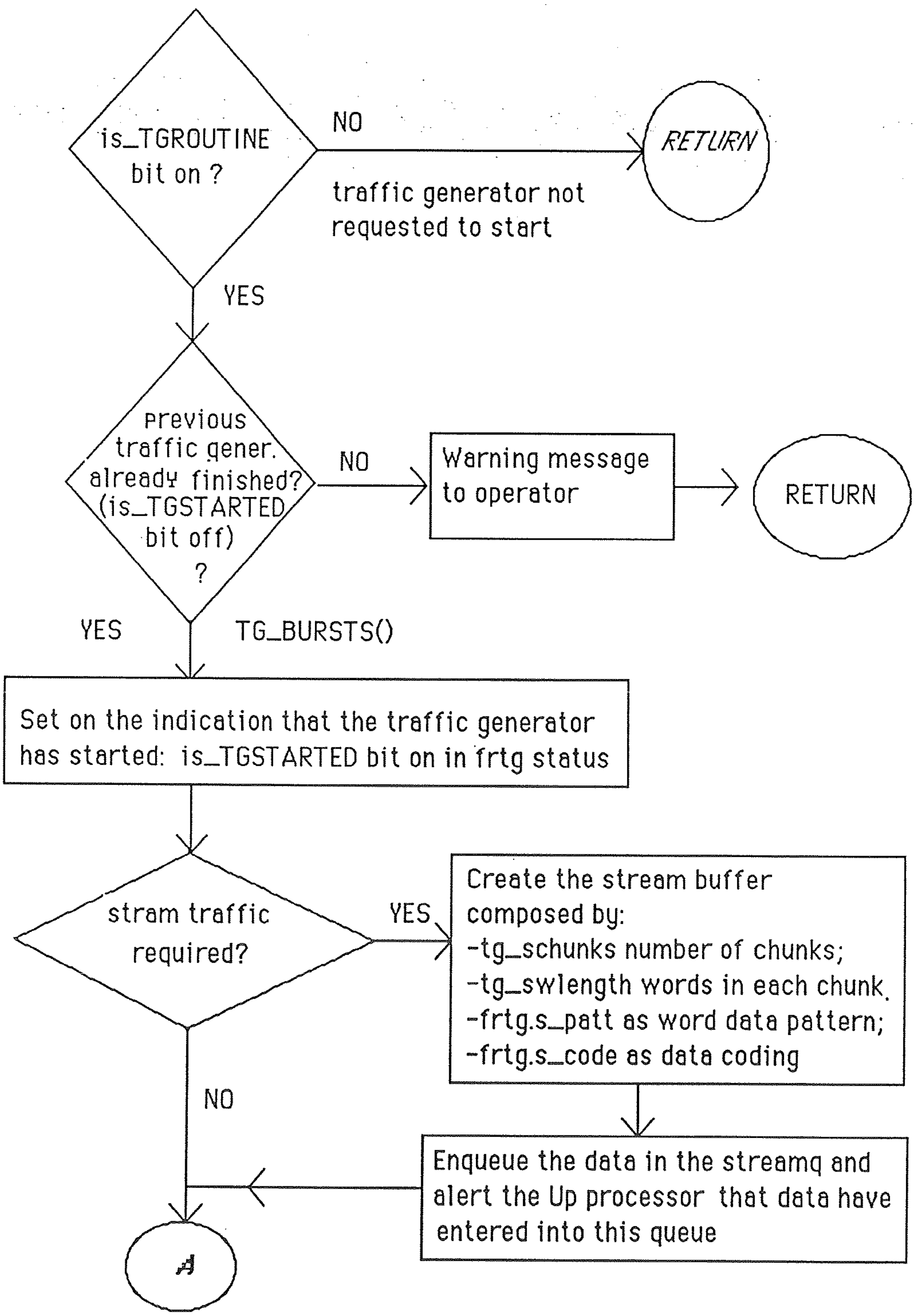
6.3 Actions at the TX timer interrupt

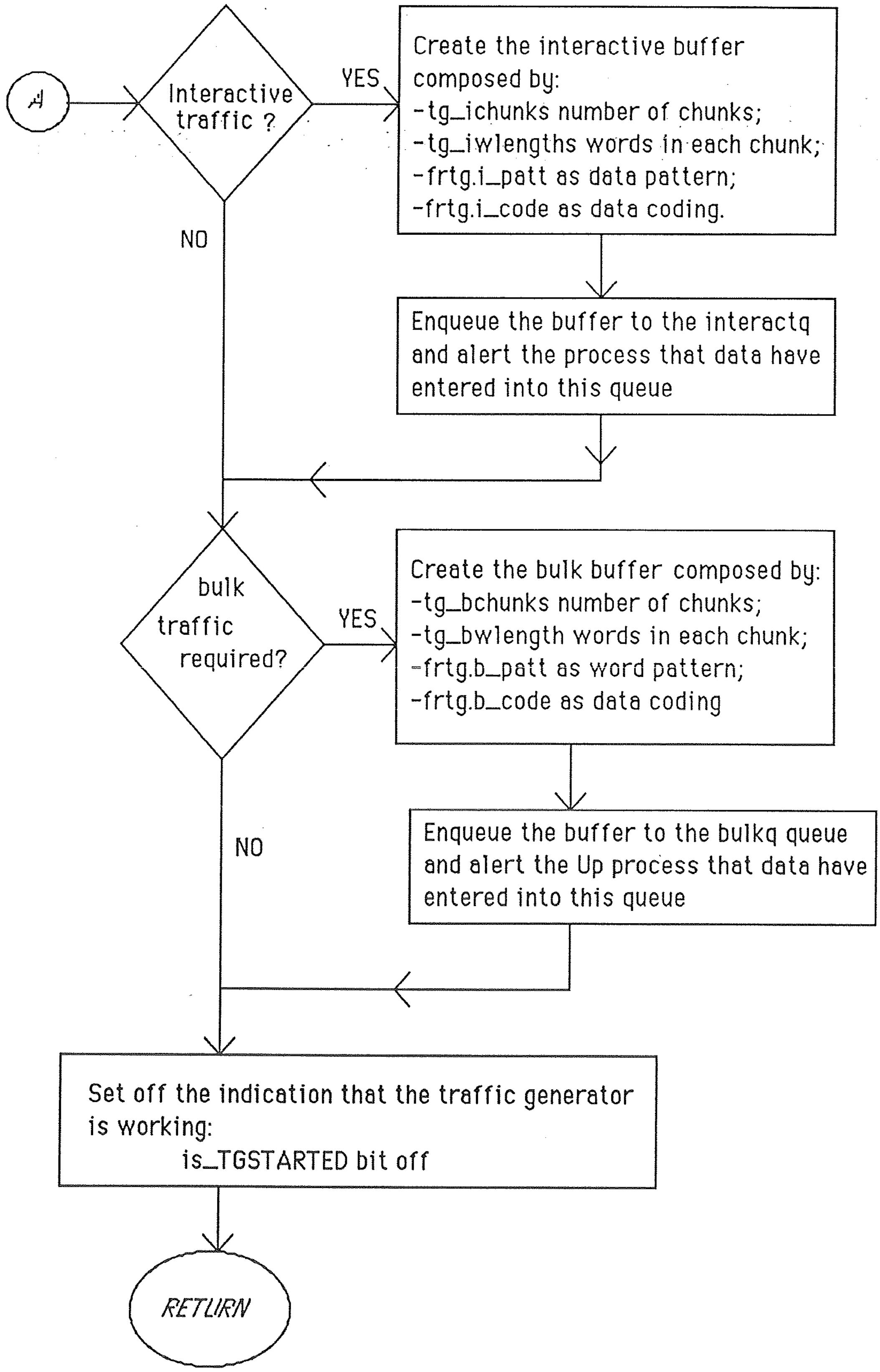
At the end, after the normal work, the "**tx_te**" routine checks whether or not the internal traffic generation has to be started.
If yes (is_TGROUTINE bit on in frtg.tg_status) but if the is_TGSTARTED bit is also on in the same status field, a message is printed on the output terminal and the traffic generator is not started because the previous bursts preparation has not yet finished.

If the "is_TGSTARTED" bit is off, the traffic generation can start and the TG_BURSTS routine is invoked. It creates the stream buffer, the interactive buffer and the bulk buffer (if the relative type of traffic was selected) and puts them into the relative queues simulating the arrival of the data from the attached local area network.

Before to check whether or not the traffic generator is requested, a check is made to verify whether or not the DMP packet has to be sent every DMP_INTERVAL frames. If yes, a special DMP counter is increased till its value reaches DMP_INTERVAL. When equal, a DMP is created and put into the bulk queue. The delay of the FODA system will be displayed on the output terminal when the down process receives the DMP packet and alerts the up process about this receiving.

The following diagrams refer only to the fixed rate traffic generator case.


is_TGROUTINE bit on ?
NO
RETURN
traffic generator not requested to start
YES
previous traffic gener. already finished? (is_TGSTARTED bit off) ?
NO
Warning message to operator
RETURN
YES
TG_BURSTS()
Set on the indication that the traffic generator has started: is_TGSTARTED bit on in frtg status
stram traffic required?
YES
Create the stream buffer composed by:
-tg_schunks number of chunks;
-tg_swlength words in each chunk.
-frtg.s_patt as word data pattern;
-frtg.s_code as data coding
NO
Enqueue the data in the streamq and alert the Up processor that data have entered into this queue
A


A
Interactive traffic ?
YES
NO
Create the interactive buffer composed by:
-tg_ichunks number of chunks;
-tg_iwlengths words in each chunk;
-frtg.i_patt as data pattern;
-frtg.i_code as data coding.
Enqueue the buffer to the interactq and alert the process that data have entered into this queue
bulk traffic required?
YES
NO
Create the bulk buffer composed by:
-tg_bchunks number of chunks;
-tg_bwlength words in each chunk;
-frtg.b_patt as word pattern;
-frtg.b_code as data coding
Enqueue the buffer to the bulkq queue and alert the Up process that data have entered into this queue
Set off the indication that the traffic generator is working:
is_TGSTARTED bit off
RETURN

APPENDICES

APPENDIX A

A Delay Measurement Packet (DMP) and a very simple burst generator on the Up process.

All the operator commands are handled on the Up process by the **"upuser"** task (UB101userhand routine defined in the UBU100.C file) which, after the initialization of some areas, enters in a wait state for reading user commands from the input terminal.
This task can receive commands from the input terminal and can write on the output terminal. It can invoke interrupt level routines but the viceversa is not possible.

In order to measure the delay of the FODA system, a special packet, named **Delay Measurement Packet** (DMP), has been created. It uses the following memory areas, allocated on the Up process, which can be dynamically opened and changed:

m_size : word length of the chunk containing the data of the DMP;
m_pattern : byte pattern of the DMP.

Moreover, a very simple **burst generator** has been implemented, which uses the following areas allocated on the Up process:

b_ckn : number of chunks constituting the **bulk** burst;
b_wlength : word length of each bulk chunk;
b_pattern : byte pattern of each bulk chunk;

s_ckn : number of chunks for the **stream** burst;
s_wlength : word length of each stream chunk;
s_pattern : byte pattern of each stream chunk;

i_ckn : number of chunks for the **interactive** burst;

i_wlength : word length of each interactive chunk;

i_pattern : byte pattern of each interactive chunk.

A) Typing **"m"** , a **DMP** is built. Memory can be opened at the Tx locations **"m_size"** and **"m_pattern"** to decide respectively the word length and the byte pattern of the delay measurement packet.
The DMP will occupy in any case not more than one chunk.

The **satellite header** of the delay measurement packet contains the special value **"DMP_TRAFFIC"** in the **destination sub-address byte** (destsa) in order to be easily recognized by the RX software when the packet will be received. At the moment the destsa byte is unused; therefore it can be used for test purposes without introducing heavy modifications in the standard software.

After the DMP packet has been built, it is enqueued to the **bulk** queue of the TX processor **(UB14qatend** routine invoked**)**. The global TX fields **m_area(0), m_area(1), m_area(2) and m_area(3)** are respectively filled with the frame counter value, the frame number value, the current backlog value and the incoming traffic value stored on the TX side of the satellite bridge after that the enqueuing of the DMP packet to the bulk queue has already been done.
Then the **tx_b_in** routine is invoked, simulating the arriving from the LAN into the TX bulk queue of a packet created to calculate the delay time in between the enqueuing to TX of a packet and its new enqueuing to the LAN on the RX side.

After the transmission on satellite, the Down process recognizes the DMP packet on the base of the particular value contained in the "destsa" byte of the satellite header.

After the enqueuing of the DMP to the LAN handler task and the clearing of the destsa byte of the satellite header, a one word **PIO** message **DELAY_MEASUREMENT** is issued versus the Up process.

On receiving of this Pio message on the Tx side, the **txd_msr** routine is entered and the global areas **m_area(4)** and **m_area(5)** are filled respectively with the frame counter and the frame number relative to the DMP enqueuing to the LAN by the Down process but stored with the same TX clock.

By entering a **"d"** command , the first 6 allocations of m_area are displayed on the terminal connected with the Up processor.

Reassuming :

m_area(0) <----- Tx frame counter at the Tx time;
m_area(1) <----- Tx frame number at the Tx time;
m_area(2) <----- Tx backlog at the Tx time;
m_area(3) <----- Tx incoming traffic at the Tx time;
m_area(4) <----- Tx frame counter at the Rx time;
m_area(5) <----- Tx frame number at the Rx time.

The DMP is always sent with destination physical address equal to that one of the sending station. The DMP word length is 2 words in order to not disturb the real traffic.

It is also possible to send a DMP every a certain number of frames chosen by the operator by entering the "n" command. The frame interval is requested and the input value is stored in the DMP_INTERVAL global memory area. The command "h" is furnished to stop the effect of the "n" command.

B) Typing "b", a buffer of "**b_ckn**" chunks, each "**b_wlength**" words long and with pattern equal to "**b_pattern**" is created and enqueued to the **bulk** queue of the Up process. Then the **tx_b_in** routine is entered to simulate the incoming from the LAN of a burst of bulk data.

C) Typing "**s**" , a buffer of "**s_ckn**" chunks, each "**s_wlength**" words long and with pattern equal to "**s_pattern**" is created and enqueued to the **stream** queue of the Up process. The BG_STREQ global area contains the stream request for this type of test application.
Then the **tx_s_in** routine is entered to simulate the incoming from the LAN of a burst of stream data.

D) Typing "i", a buffer of "**i_ckn**" chunks, each "**i_wlength**" words long and with pattern equal to "**i_pattern**" is created and enqueued to the **interactive** queue of the Up
process. Then the **tx_i_in** routine is entered to simulate the incoming from the LAN of a burst of stream data.

E) Typing "**g**", a bulk burst + a stream burst + an interactive burst are generated (b+s+i).

F) Typing "k", a bulk burst + a stream burst are generated **(b+s)**.

G) Typing "w", an interactive burst + a stream burst are generated **(i+s)**.

Appendix B

Some global memory areas

Here only some global memory locations are presented. They can be opened on the Motorola 68000 by means of the **ope** " memory address" command and their value changed.

RX areas

DRQ	datagram request queue
SRQ	stream request queue
FSQ	free station element queue
SELQ	station element queue
SCBTAB	table of the SCBs
NBQ	new born stations queue
DSQ	down station queue
CCB	channel control block
RXCB	RX control block
SRBINF	information regarding the received superframe reference burst
RBMYDA	my datagram allocations as received in the reference burst
MDALLOC	the datagram allocations for all the active stations
REFBURST	reference burst area
CLNAREA	"change logical number" area
SCLNAREA	"stop sending the change logical number"
NSB	number of received sub-bursts
STATION_OFFSET	master station offset
MAX_RB_MISSED	max number of allowed missed reference bursts
NOW_START_AS_MASTER	number of interrupt timer without receiving reference bursts after which the station can start as master.

TX AREAS

st_off	current station offset
cln	new logical number
sln	dummy area
rbfix	fixed part of the reference burst (datagram allocations not included)
txcb	TX control block
dab	datagram allocation block
wdbtab	window descriptor blocks table
INCOMING_TRAFFIC	the traffic incoming from the LAN
CURRENT_TRAFFIC	the current traffic
TX_BACKLOG	the backlog
null_chunk	address of an area containing zero
AFTIME	time to add a fragment
ACTIME	time to add a control data
CCTIME	time to chain a chunk
CBTRS	bit sequence for clock recovery
PREAMBLE_CODING	coding of the preamble
CSBCHUNK_CODING	coding of the chunk containing the control sub-burst
CSBCW_CODING	coded data control word
CSBCW_UNCODING	unencoded data control word
DCHUNK_CODING	coding of the chunk containing coded data
DCHUNK_UNCODING	coding of the chunk containing unencoded data
FRTG_STREQ	stream channels requested by the fixed rate traffic generator test application
BG_STREQ	stream channels requested by the burst generator test application
VOICE_STREQ	stream channels requested by the Cambridge Ring voice application
STREQ_AREA	total of the stream channels requested by all the stream applications running in this station.

RX/TX COMMON AREAS & FODA SYSTEM PARAMETERS

GT_SIZE	byte guard time between bursts = 10 bytes
IF_GAP_SIZE	inter frames gap (in bytes) = 255 bytes
CSB_SIZE	control sub-burst minimum byte length = 16 bytes
NCTL_SLOT	number of control slots in a frame = 4
CTL_DI	bytes for the small data in the control slot = 64
CURRENTSPEED	megabytes per second = 2
ES	bytes constituting an elementary slot = 16
HALF_RATE	1/2 coding rate
FULL_RATE	unencoded rate
SFL	number of frames constituting a superframe = 64
HEADER_CODING	coding for the satellite header
RB_CODING	coding for the reference burst
STREAM_DI	bytes for each stream channel = 256
FRAME_SIZE	bytes in a frame =
CBTRS_SIZE	bytes for the CBTRS (Clock Bit Timing Recovery Signal) = 36
UW_SIZE	bytes for the unique word = 4
HEADER_SIZE	bytes for the satellite header after coding
PRB_SIZE	bytes for the preamble (CBTRS + UW)
PRB_LEN	word length +3 (hardware requirement) of the preamble
BURST_OVERHEAD	bytes overhead of a burst (GT_SIZE + CSB_SIZE + PRB+SIZE) = 66
RB_OCCUPANCY	byte occupancy of the reference burst after the coding = 140
RB_SIZE	byte used to send a reference burst = PRB_SIZE + RB_OCCUPANCY
RB_SLOT_SIZE	byte size of the slot used to send the reference burst
CTL_SLOT_SIZE	bytes size of each control slot. It must be >=IF_GAP_SIZE and <=BURST OVERHEAD + HEADER_SIZE + CTL_DI
CB_SIZE	how many bytes can be sent in each control slot = PRB_SIZE + CSB_SIZE + HEADER_SIZE + CTL_DI
CTL_SF_SIZE	bytes reserved for the whole control sub-frame = 4 * CTL_SLOT_SIZE
RB_START_TIME	after how many bytes the reference burst starts = IF_GAP_SIZE
CTL_START_TIME	after how many bytes the control sub-frame

	starts = RB_START_TIME + RB_SLOT_SIZE
STR_START_TIME	after how many bytes the stream sub-frame starts = CTL_START_TIME + CTL_SF_SIZE
STR_SLOT_SIZE	byte dimension of a stream slot = BURST_OVERHEAD + HEADER_SIZE + STREAM_DI
CW_SIZE	control word byte length = 4
AMOUNT	minimum byte occupancy for sending an elementary slot
ASS_MIN_SIZE	minimum byte assignment for datagram
WDB_OVERHEAD	bytes overhead for each transmitting window
HDRLEN	real satellite header word length
RB_STRUC_SIZE	real byte length of the fixed part of the reference burst
RBFIX_WLEN	real word length of the fixed part of the R.B.

Some considerations about the master fault recovery procedure.

This procedure has not been implemented in the 2 Mbit/sec case bacause hardware modifications are required. From the software point of view, its implementation is very easly.
Here are reported some considerations.

In order to limit as much as possible troubles to the slave stations when the master station falls down, a master fault recovery algorithm is necessary.
The basic idea is that the possible must be done in order to continue at least the stream transmissions.
When a master fault is detected, no datagram transmission is allowed before that a new station starts as master.
When a new master station is active, the slave stations must send again their datagram requests (a datagram congestion may appear in some station in the time between the master silent and the new master start).

The rules the master fault recovery algorith is based on are:

1) The active station with the current lowest logical number is designed to become the new master station.

2) A slave station must wait for a time

K * its current logical number

long, during which no reference burst is received, before to decide to start as master because of the master fault. K is here a constant indicating a fixed number of frames.

3) In order to be ready to become master, each slave station must save the last received superframe reference burst (SRB) because it contains the last valid stream assignments. If just a SRB is lost, the first valid reference burst belonging to that superframe must be saved.
It is not necessary to copy the datagram allocations because they refer only to a frame. Datagram allocations are in any case lost during the master fault.

4) For the same reason of point 3, each slave must maintain the queue of the stream requests (datagram are not necessary because the are lost in any case) and perform the stream assignment algorithm as if it were master. In this way, each slave has an as much as possible updated situation of the stream requests and assignments.
Of course, nothing can be done to avoid that a slave does not receive a stream request which is instead received by other stations.

5) The slave station which has to become master, must:
- change its role from slave to master (set on the MASTER bit in the rxcb.role area);
- initialize the DMA as master;
- assume the logical number 0;
- take the scb #0 , clearing the previously used scb.
  It is necessary that the pointers to the streamq present in the previously used scb be correctly changed. The pointers to the datagram queue must be cleared;
- clear the "missrbcounter" area in the rxcb;
- set its old logical number as "hole" in the SRB (the last saved which will be the first sent);
- send the SRB to the UP process via the MASTER_SRB PIO message.
  Also the datagram allocations must be passed to the Up process via the MASTER_DALLOC PIO message.
  In this way the UP process realizes that now he must act as master and that therefore its logical number is zero.
  Also the use of the control slot (which is handled by the logical number) is affected by the fact that now the logical number must be zero.

Appendix D

How to build the UP/DOWN processes for the Motorola 68000 machine using a PDP11.

This appendix is useful for those implementations of the FODA system where the software developing machine is a PDP11 running the RSX11M operating system and the target machine is the Motorola 68000 using the C-EXEC operating system Version 1.5.
The C-EXEC operating system is a portable operating system which is produced by Whitesmiths. It consists of:

- a C-cross compiler;
- a linkage editor;
- various utility programs;
- the C-EXEC operating system as a number of objects modules linked together in the LCEX . 68K library;
- a few essential routines given in source form.

The C-EXEC operating system is normally developed in a cross fashion. Programs are developed on other hosts, linked together and then down loaded into the target machine.
In the CNUCE implementation, we had the **C-EXEC Version 1.5** system running on a PDP11/70 machine under RSX11M V.4.0. The goal was to produce on the PDP two down line loadable files (Up and Down processes) for the Motorola 68000 machines.

In the CNUCE implementation, all the software necessary for the "satellite bridge" experiment was under the following UICs:

(1,1) contains the LIBCVDS.68K C-Cross library;

(1,54) contains the following tasks:
LINK.TSK /task=...LNK to combine object files;
REL.TSK /task=...REL to examine object files;
HEX.TSK /task=...HEX to translate object file to ASCII format;
LRD.TSK /task=...LRD to order libraries;
LBY.TSK /task=...LIB to maintain libraries;
A68.TSK /task=...A68 the MC68000 assembler;

P68.TSK /task=...P68 the C code generator for MC68000's;
CPP.TSK /task=...CPP preprocess "define" and "include" used by the C compiler;
CP1.TSK /task=...CP1 to parse C programs.

(5,2) contains all the files constituting the down line loader from PDP to the Motorola;

(5,3) contains the whole software necessary to the "satellite bridge" experiment. This UIC includes a big number of files, so divided in classes on the base of their use:

RX *.C	files constituting the FODA Down process;
TX *.C	files constituting the FODA Up process;
RXASS .S	MC68000 assembler file for the FODA Down process;
TXASS .S	MC68000 assembler file for the FODA Up process;
UBD *.C	files necessary to the Down process;
UBT*.C	files necessary to the Up process;
UBx*.C	files common to the UP and the Down processes;
UBA1.S	MC68000 assember common file;
XC10A68K.C	ring driver (xc) iorb-handling routines;
XC11A68K.C	ring driver (xc) interface routines;
XC13A68K.C	ring driver (xc) dummy routines;
RXIN100.S	C-EXEC startup code for the Down process;
TXIN100.S	C-EXEC startup code for the Up process;
* .H	header files;
* .X	external references file (header file);
* .R	routine list header file;
MEASURE .*	files for traffic measurements (test only);
SATLCEX .68K	C-EXEC system library modified for the experiment;
ULIB .68K	JMI Portable C library;
ULINK .O	module to provide the user interface to the C-EXEC;
UPBLD .CMD	command file to link the C-EXEC nucleus with the user UP task. This CMD uses the following other CMDs: TX1 .CMD TX2 .CMD TX3 .CMD
DOWNBLD.CMD	command file to link the C-EXEC nucleus with the user DOWN task. This CMD uses the following other CMDs: RX1 .CMD RX2 .CMD

RX3 .CMD.

Some notes must be written about SATLCEX.68K, ULIB.68K and ULINK.O. The original C-EXEC system library was called LCEX.68K and, in the CNUCE implementation it was put under the UIC (5,6). No sources of the library modules were furnished; the only sources were for the devices and the clocks handling which optionally can be added on the base of the system configuration.

For the "satellite bridge"' experiment the LCEX.68K library was modified. First of all, it was copied under the UIC (5,3) with the new name of SATLCEX.68K, then 3 new modules were added at the end and a pre-existing module was substituted.

The added modules are:

WAKE .O	(source WAKE .S);
EC6850 .O	(source EC6850.C) for the 6850 boards;
CLK68K .O	(source CLK68K.C) for the clock handling

and the replaced module is SYSCFG.O .

Wake.o is a module for the "wake" device which wakes up the ring handler process when the VMI/1 interrupt occurs. VMI/1 is the Logica VTS Cambridge Ring interface from Multibus to Polynet equipment.

The satellite bridge programs use the omnibyte 68000 board serial driver OM6850 to drive the terminal line and the serial line between the two bridge halves. This was done in the module EC6850.o.

To add or to replace modules in the SATLCEX .68 library, the LIB task was used with the following commands:

```
set /uic=(5,3)
ins $lby/task=...lib
lib satlcex.68k -r wake.o
lib satlcex.68k -r ec6850.o
lib satlcex.68k -r clk68k.o
lib satlcex.68k -r syscfg.o
```

The ULIB.68K was only copied from its original UIC under the (5,3) UIC but no changes were done. The same is for the object file ULINK.o whose source was not provided.

APPENDIX E

<u>The Foda system installation under C-EXEC Release 2.1b.</u>

This installation has been made on an IBM PC-AT. It is the current working implementation of the FODA system at 2 Mbit/sec. As the previous implementation was on a PDP11/70 machine, running the RSX11M V4.0 operating system and the Motorola operating system was a C-EXEC V.1.5, this appendix is useful to describe how to pass from the PDP implementation to the PC-AT implementation.

Note: / referred to a directory, is used instead of the "back-slash".

1. THE DIRECTORIES

Three directories have been created on the IBM PC:

a) c:/c/ containing the C-Cross compiler for the Motorola 68000;

b) c:/c-exec/ containing the C-EXEC operating system <u>Release 2.1b</u> as given on the base of the "binary release";

c) c:/foda/ containing the files necessary to link the FODA system to the C-EXEC nucleus.
They are:

RX*.C	source C files for the Down process;
RXASS.S	source assembler file for the Down process;
UBD*.C	other source C files for the Down process;
<u>RXCONFIG.C</u>	file containing the driver configuration table and the user process configuration for the Down process. In the previous version 1.5 of C-EXEC, this

	file was called UBD1.C;
RXIN68K.S	file containing the linking of the down process to the C-EXEC nucleus. In the previous version 1.5 of C-EXEC, this file was called UBD100.S
TX*.C	source C files for the up process;
TXASS.S	source assembler file for the up process;
UBU*.C	other source C files for the up process;
TXCONFIG.C	file containing the driver configuration table and the user process configuration for the up process. In the previous version 1.5 of C-EXEC, this file was called UBU1.C;
TXIN68K.S	file containing the linking of the up process to the C-EXEC nucleus. In the previous version 1.5 of C-EXEC, this file was called UBU100.S;
UBA1.S	common file; PIA interface;
XC10A68K.C	common file;
XC11A68K.C	common file;
XC13DUMMY.C	common file;
UBn*.C	common utility files;
*.H	header files;
*.R	header file; list of global routines;
*.X	header file; external definitions;
HEAP.S	the "heap" file, solved at link time;
SATCXLIB.68K	the standard C-EXEC library, with modules added for the FODA requirements;
ULINK.O	the file necessary to link the user files to the C-EXEC nucleus during the link phase;
WAKE.C	the obj is added in satcxlib.68k;
OM6850.C	the obj is added in satcxlib.68k. In the version 1.5 of C-EXEC, this file was called EC6850.C.
CLK68K.C	the obj is added in satcxlib.68K;
*.TXT	text files. RX1, RX2 , RX3 .TXT are used in the command file to create the down process. TX1, TX2, TX3 .TXT are used in the command file to create the up process.
*.BAT	command files. In particular:

UPBLD to create the FODA UP process;
DOWNBLD to create the FODA DOWN process;
CR68 to cross-compile for M68000 .C files:
cr68 name1 name2 name3 ....
AS68 to assemble for M68000 .S files:
as68 name1 name2 name3 ....

2. THE C-CROSS COMPILER FOR MOTOROLA 68000

The files on the original 2 diskettes have been copied under the directory c:/c and the command file

2buildx

has been invoked to generate the C cross-compiler.

After the generation, the following tasks (.COMM) are of interest for us:

PP	the C pre-prosessor
P1	the C parser
P268K	the MC68000 C code generator
AS68K	the MC68000 assembler
LINK	the Whitesmiths linker
LIB	the Whitesmiths librarian
HEX	to convert object files to hex records
REL	relocatable object modules inspector
LORD	the library ordering process

and the libraries:

LIBU.68K	UNIX style I/O library
LIBCCPM.68K	the C library for CP/M-68K.

3. The C-EXEC operating system Release 2.1b

The furnished release of C-EXEC is 2.1b, so a lot of problems had to be solved in the FODA system to pass from release 1.5 to release 2.1b.

1) The nucleus of the C-EXEC for our IBM PC-AT machine is in the file c:/c-exec/omin68k.s. It has been copied under the c:/foda/ directory,

renamed IN68K.S and duplicated into two files, RXIN68K.S and TXIN68K.S respectively.
In each of the two files, modifications have been done to link the nucleus of the C-EXEC respectively to the down process and to the up process.
The two files substitute the corrispondent RXIN100.S and TXIN100.S files of the 1.5 version.
With respect to version 1.5, the file SYSCNFG.C (which, with modifications, substituted the original module in the satlcex.68K library) does no more exist, but part of it is now included in IN68K.S.

In the IN68K.s file, note that the calling to the comint routine has changed format with respect to C-EXEC version 1.5.

2) The C-EXEC system header files (*.h) have been copied from directory c:/c-exec/ to directory c:/foda/ with the following modifications, where necessary:

STD.H	replaces the previous one. Pseudo-types used by the FODA system have been added;
MACH.H	replaces the previous one. Definitions added;
UPROC.H	replaces UTASK.H
DRIVER.H	replaces DIO.H
TTY.H	replaces DIO.H
DSTAT.H	replaces DSTAT.H
DVCFG.H	replaces DEV.H
IOCTL.H	added
QUE.H	added

3) The C-EXEC library is now called CXLIB.68K (in Release 1.5 it was called lcex.68k).
It has been copied from c:/c-exec/ directory to c:/foda/ directory and renamed into SATCXLIB.68K (instead of satlcex.68k).

The following modules have been added to the library:

a) OM6850.C — has been copied from directory c:/c-exec/ to c:/foda/ and modifications have been added as it was in module EC6850.C of version 1.5. This module is the terminal driver.

b) WAKE.C

already present in the FODA directory.
This module is for waking up other processes.
Modifications with respect to version 1.5:
DEVT---------->TTY
DIO.H---------->DRIVER.H + TTY.H
An empty routine "wakdrv" for the "wake" driver handling has been added. It is referred in the DVCFG table defined in the RX/TX CONFIG.c file. The "wake" driver is not used at the moment.

c) CLK68K.C

already present in the FODA directory.
This module substitutes the original module OM6840.c for the clock handling because in the FODA system the handling of the the system clocks is the following:
clock 1 is used by the FODA system
clock 2 is used by the C-EXEC system.
No modifications apported.

The three previous files have been compiled (cr68 command file) and the relative .o files have been added to satcxlib.68k with the following command:

```
lib satcxlib.68k -r name.o
```

4) Under the c:/foda/ directory, the files **UBU1.C and UBD1.C have been respectively renamed into TXCONFIG.C and RXCONFIG.C** and heavy modifications have been apported to upgrade from release 1.5 to release 2.1b.

The Device Configuration Table has now been generalized in that:

a) it contains no device type specific information;
b) major device number is replaced by a pointer to the device driver entry point;
c) terminal driver buffer sizes are now in a TTYI structure;
d) queue buffer size is now in a QUEI structure;
e) the driver address table DRVTAB previously defined in

SYSCONFIG.c) does no more exist.

The User Task Table is now referred to as the User Process Table and the name of the associated structure has changed from UTASK to UPROC. The header which must be included to obtain the definition of UPROC has changed from <u>utask.h</u> to <u>uproc.h.</u>

The User Process Table no longer contains an index into the Device Configuration Table for standard input, standard output and standard error output for each process. Instead, the name of the device is used.
The priority of a process is defined by its relative position in the user-defined process table, with the highest priority process in the first table entry.

The 68000 version 2.1b of C-EXEC now runs entirely in supervisor mode. This was done primarily to enhance performance. This also eliminates problems on some systems in which processes running in user mode could not access the I/O devices.

Refer to the following manuals:

1) C EXECUTIVE release 2.1b
   Enhancements, changes and upgrade instructions.

2) C EXECUTIVE user manual chapter 7.

5) The routine "FILL" , defined in TXASS.s and in RXASS.s had to be renamed in "WFILL" because "fill" was already a routined used by the operating system. It provoked big troubles at the tty driver initialization!
Therefore, also the "COPY" routine has been renamed in "WCOPY", for security purposes, being its name very common.
In total, the following changes have been made:

FILL ------------> WFILL

TXASS.s
RXASS.s
UBU100.c

COPY------------> WCOPY

TXASS.s
RXASS.s
TXBURST.c
UBD100.c
UBU100.c

4. USED COMMANDS

a) **CR68** name1 name2.... to cross compile for M68000 C files;

b) **AS68** name1 name2... to assemblefor M68000 S files;

c) **UPBLD** to link to the C-EXEC operating system the user files for building the UP process;

d) **DOWNBLD** to link to the C-EXEC operating system the user files for building the DOWN process;

e) **SMARTERM** to run the utility program to load on the UP/DOWN bridge the UP/DOWN process created with the phase c/d.

The loadable modules are:

RXSYS.HEX on the down side of the satellite bridge;
TXSYS.HEX on the up side of the satellite bridge.

Before to start the transfer of the loadable module, on the selected side of the satellite bridge the following command must be issued:

re; -x = <carriage return>. The receiving data are displayed
or
re; = <carriage return>. The receiving data are not displayed

in order to receive the loadable data.

R E F E R E N C E S

1) Beltrame R. , Bonito A.B., Celandroni N., Ferro E.:
"FODA_TDMA. Final report on the new protocol for mixed traffic. Theoretical study and first simulation results", CNUCE Report C85-03, June 1985, 2nd edition.

2) Adams C.J. et al.: "New Satellite Bridge", RAL paper, 1984.

3) Marconi Research Centre:
"Technical proposal for the new satellite bridge hardware for use in project UNIVERSE", Y/212/5949, February 1983.

4) Whitesmiths Ltd: "C interface Manual for PDP11".

5) Whitesmiths Ltd: "C interface manual for MC68000".

6) Celandroni N., Ferro E.:
"FODA-TDMA satellite access scheme: description, implementation and environment simulation.", proceedings of the Tirrenia International Workshop on Digital Communications, Tirrenia (Pisa), September 14-16 1987. Also CNUCE Report C87-17.

7) Beltrame R., Celandroni N.:
"The performances of the FODA access scheme: theory and simulation results", CNUCE Report C86-19, December 1986.

8) JMI Software Consultants, Inc.
"C-EXECUTIVE Release 2.1b: Enhancements, changes and upgrade instructions".

9) JMI Software Consultants, Inc.
"C-EXECUTIVE Release 2.1 user manual".