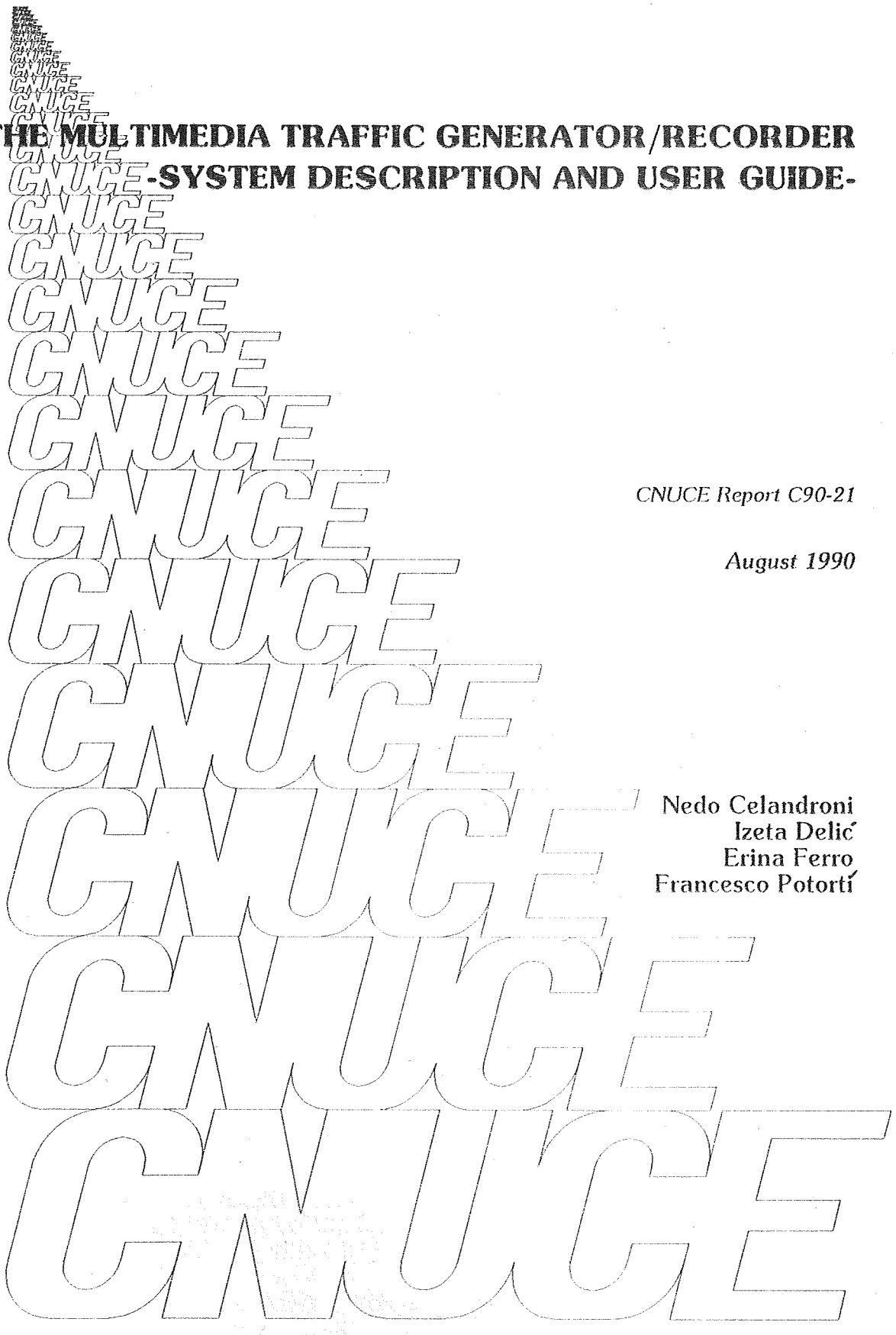
THE MULTIMEDIA TRAFFIC GENERATOR/RECORDER
-SYSTEM DESCRIPTION AND USER GUIDE-



CNUCE Report C90-21

August 1990

Nedo Celandroni
Izeta Delić
Erina Ferro
Francesco Potortí

Table of contents

INTRODUCTION

The Multimedia Traffic Generator-Recorder (MTG/R) system is aimed at supporting the testing and the evaluation of the Satellite Network Access System (SNAS) performances.

The SNAS is based on the Fifo Ordered Demand Assignment - Time Division Multiple Access (FODA - TDMA) satellite access scheme, described in detail in [1], [2] and [3].

The behavioural aspects of different LANs and MANs can be observed and explored with the MTG/R system.

This report is divided in three chapters. Chapter 1 describes some features and important information regarding the FODA system. Part 2 gives the basic characteristics of the MTG/R system, including detailed information about organization and purposes.

Chapter 3 contains a guide to the use of the system. Appendixes A and B show the system software organization and the system software flowchart respectively.

1. THE FODA SYSTEM

A rough description of the internal FODA functioning is here given and some key terms and basic functional blocks are explained.

FODA software - the software for receiving/transmitting data from/to the satellite, running on the RX-TDMA and on the TX-TDMA units, respectively.

TDMA terminal - indicates both the RX-TDMA and the TX-TDMA units consisting of:

a) a burst-mode modem with variable bit rate (1÷8 Mbit/s) and 5 different coding rates;

b) a TDMA controller. The transmit unit of the TDMA controller consists of 3 boards: the user interface card, the control processor (Motorola 68030 running at 25 MHz) and the modem interface card, including the FEC encoder. The receive unit consists of 3 corresponding boards plus the FEC decoder. The receive and the transmit units are interconnected via an SCSI link.

The user interface boards of the TDMA controller support:

a) two input ports (64 Kbit/s, CCITT G703 standard),
b) two output ports (64 Kbit/s, CCITT G703 standard),
c) eight asynchronous ports (0.1- 19.2 Kbit/s, RS-232),
d) synchronous I/O interface (384 Kbit/s, CCITT RS-422),
e) two Ethernet ports.

FODA system - both the FODA software and the TDMA terminal.

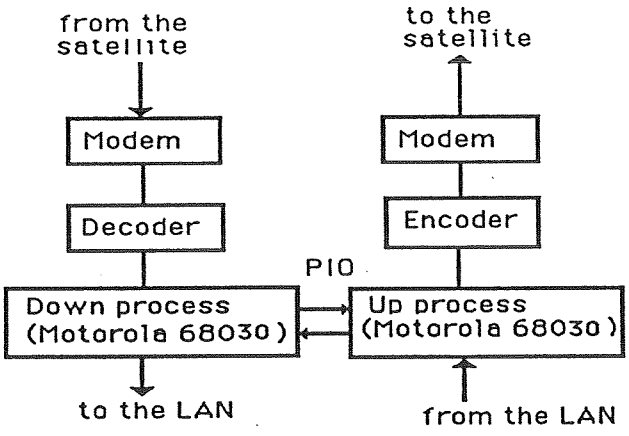


Fig 1. The TDMA terminal

Here is an overview of the main features of the FODA system.

Transit or private network

The system can work both as a **transit network** and as a **private network** with directly attached users.

Centralized control

The channel scheduling and the control are **centralized.**

Datagram and stream traffic

Two different basic types of traffic are supported:

a) ***datagram*** *traffic.* It is generated by bulk data transfers, interactive computer accesses, mailings and data base manipulation. The required channel characteristics are essentially: high throughput and low bit error rate, while the delay is not very critical.

b) ***stream*** *traffic.* It is generated by synchronous applications like the packetized voice and video in full or slow motion. The required channel characteristics are: fixed throughput, very low delay and regular inter-arrival time of the packets.

Access strategy

The access to the satellite is achieved through the **centralized technique** and a **FIFO scheduling discipline** used for the reservations. The transmission time is divided into *time frames* 31.25 ms long.

Some nomenclature follows to make comprehensible Fig. 2.:

a) elementary slot (Sel) 16 octets (8-bit bytes) of information (minimum information item);

b) number of consecutive elementary slots (Ncs) - any number of consecutive elementary slots;

c) stream assignment slot (Ssa) - smallest portion of time assigned for stream;

d) datagram assignment slot (Sda) the portion of time assigned to a station for sending data;

e) control sub-frame (CS) contains 4 slots assigned cyclically to all the active stations;

f) stream sub-frame (SS) - contains stream slots, assigned regularly in time to the requestors, up to an upper limit in the frame;

g) the datagram sub-frame (DS) - contains datagram slots for interactive and bulk data traffic.

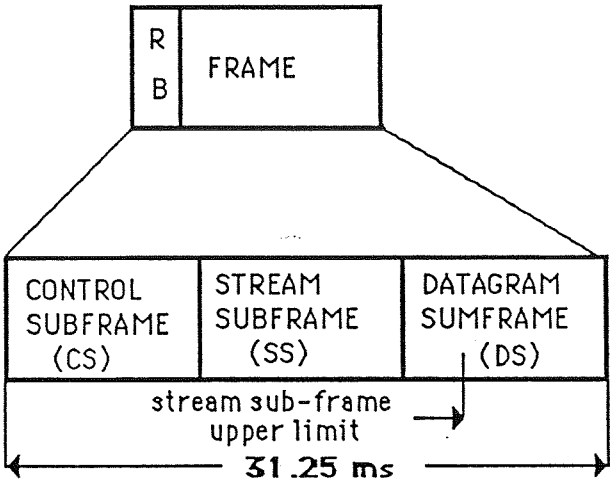


Fig 2. Frame structure

Reference burst

The *reference burst (RB)* is sent by the master station for synchronism purposes and to distribute control information like the allocations in the SS and in the DS.

Framing

Frame is the interval of time containing the RB, CS, SS and DS sub-frames.

Network startup

a) The station starts and listens to the channel to recognize the on-going time frame. If, in a certain time, nothing is detected, the station may start as *master* transmitting the *Reference Burst*. The master station assigns to itself the logical number 0 and indicates in every reference burst the next available logical number and the number of the currently active stations (*slave stations*).

b) Each active station in the system is dynamically assigned a logical number in the range from 0 (reserved to the master) up to the number of the currently active stations, rounded to the next multiple of 4. The logical number is the basis to compute the access to the right control slot.

c) Any further activated station sends the *new born* control message to the master, using its control slot, whose position is computed by considering the *next available logical number* and the number of active stations. Therefore, to access the control slot the very first time, the stations pre-empts the logical number declared available in the reference burst. This number may be changed by the master, when the vacant logical number released by a *down* station must be utilized.

Channel assignment

a) Stream Requests and Assignment

Each slave station sends to the master an allocation request for a number of stream channels. This request is the sum of all the requests received from the applications. Once granted by the master, the request remains valid until a modification (or cancellation) is issued by the requesting station. Datagram may be transmitted by a station in its stream transmit window when no stream data are available.

b) Datagram Requests and Assignment

The requests for datagram allocations generally change in each frame, representing the number of elementary slots needed at that moment. The datagram allocations are assigned in the remaining space of the time frame after the stream. If no stream is present, they start immediately after the control sub-frame.

The TDMA burst structure is shown in Fig. 3.

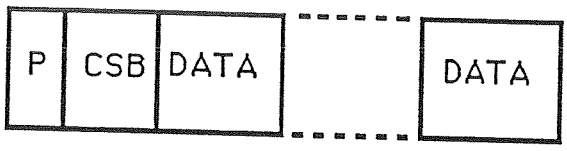


Fig. 3. The TDMA burst structure

P - Preamble sub-burst. It consists of the Carrier and Bit Timing Recovery Sequence (CBTRS) plus a 48 bits unique word (UW) and the guard time.

CSB - Control Sub-Burst. It enables the following sub-bursts to be identified and decoded. In the CSB an area is included to send control

information and requests (channel control area). Following the channel control area, the satellite headers are included, each one associated to its pertinent data sub-burst.

DATA - Data Sub-burst. Any length up to 16380 bytes is allowed.

Miscellanea

- If a maximum of 64 simultaneously active stations is considered, every half a second (at least) one control slot is surely given to each of the active stations, without any risk of collision.
- Mechanisms are implemented so as to keep constant (to a certain extent) the requested transmission quality, in terms of end-to-end delay and bit error rate. This behaviour is needed by most applications.
- Opening/closing of virtual circuits and retransmissions of corrupted packets are demanded to higher level protocols.
- Different FIFO queues are organized according to the different types of data. One stream queue and two datagram queues (the interactive type of traffic is given higher priority with respect to the bulk type of traffic) are provided.
- Fragmentation/reassembling techniques are foreseen. Channel saturation control techniques are also foreseen.

2. THE MTG/R SYSTEM

The satellite controller, the FODA software runs on, is connected to a Delta 3300 system via an Ethernet link. The MTG/R runs on the Delta 3300 machine under the UNIX (SYS - V/rel. 3) operating system.

The MTG/R is able to generate and to record traffic from several uncorrelated *Traffic Generators (TGs)* which are, in fact, different data sources with various characteristics.

Each data packet is sent over the communication medium (Ethernet), across the Up (Tx side) part of the FODA system, and looped-back to the MTG/R through the Down (Rx side) part of the FODA system.

The C language has been chosen as the most suitable and convenient implementation language.

2.1 The Hardware

Fig 4. gives a rough scheme of the hardware structure which embodies the MTG/R operating environment. The TDMA controller performs the interconnection between the LAN and the earth station.

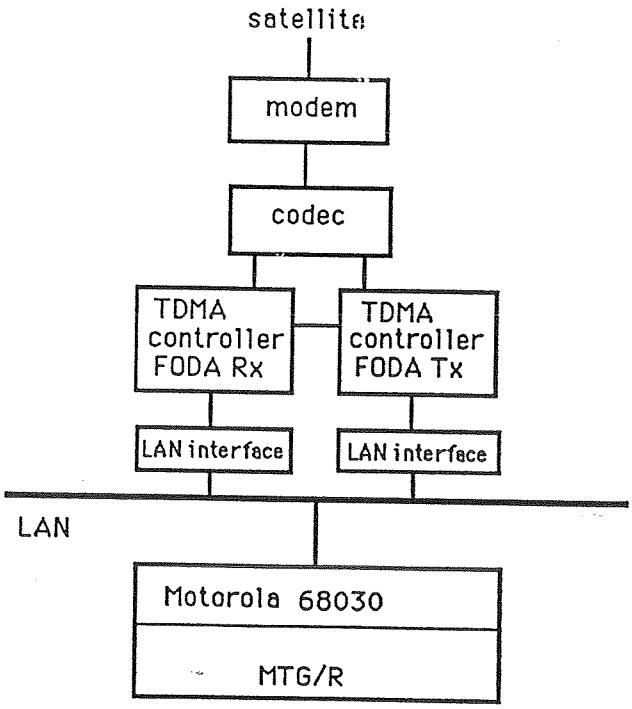


Fig 4. System configuration

Only the key features of the hardware are mentioned. It may be supposed that the final MTG/R hardware environment will be slightly changed.

- **LAN:** currently Ethernet.
- **CODEC:** it must be able to support variable coding rates, switchable from sub-burst to sub-burst. Adapting the code rate and, eventually, also the data rate to the signal/noise conditions, bit error rates better than 10^{-4}, 10^{-6}, and 10^{-8} are offered respectively, for different services.
- **MODEM:** it must be able to support variable satellite link bit rates: 1, 2, 4 and 8 Mbit/s, switchable from sub-burst to sub-burst, with two different types of modulation schemes (BPSK and QPSK) to allow the operation within a 5 MHz bandwidth.
- **DELTA 3300** (MC68030 with 25 MHz clock). A protocol interconnecting the FODA system to the MTG/R system has been here implemented (GA-FO protocol) for data and control messages exchange [5].

2.2 MTG/R General Structure

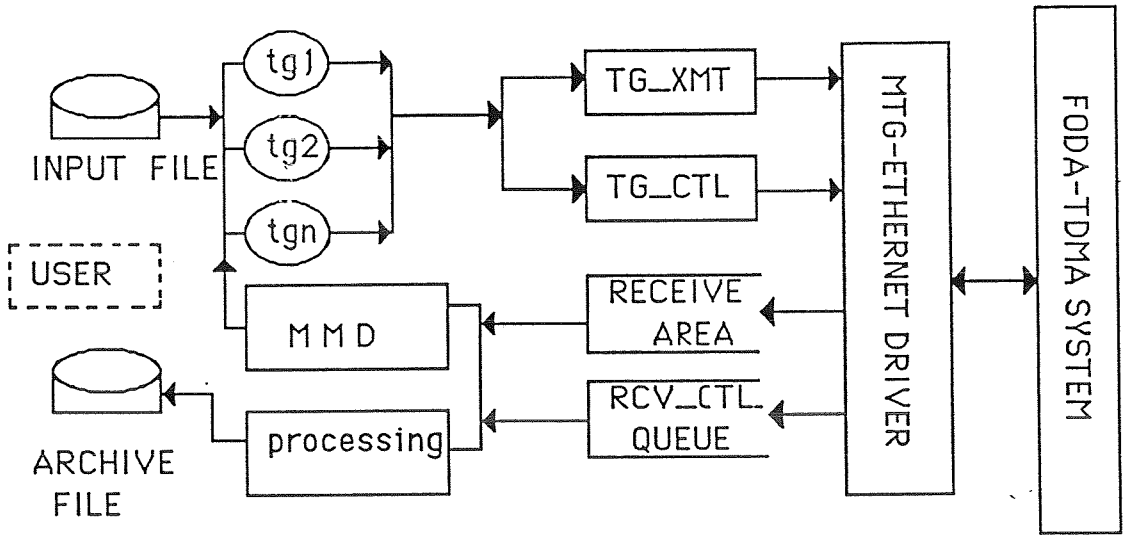


Fig 5. MTG/R general structure

To provide the required amount of traffic, the MTG/R system (Fig. 5) has been organised in two separate, but closely interrelated, layers:

a) *the MTG /R user part* and
b) *the MTG-Ethernet driver.*

a) MTG/R user part

User is the part of the MTG/R system allocated and running in the *user space* of the UNIX operating system.

The Traffic Generator Tables (TGTs) are prepared by the user, one for each TG. Each TGT contains information about the addresses, the lengths and the time intervals relative to the packets to be sent, to accomplish the function of that particular TG.

From the user's point of view, each TGT is described by:

- the Ethernet header
- the GA-FO header [5]
- the MTG header
- the data to be sent
- the TGT and the timetable addresses
- the size of the timetable
- the timetable entries (for the TG_CREATE system call).

Each entry in the timetable contains:

- the length of data to be sent with the current packet,
- the number of ticks which must elapse before sending the next packet. This timetable is terminated with a null entry.

The TGTs are prepared by the user and delivered to the MTG-Ethernet driver by issuing appropriate commands which provide all the needed services on the transmit side (TG_XMT). The interface between the MTG-Ethernet driver and the MTG/R is provided by the Unix *ioctl* system call.

The received messages (loop-backed by the FODA system) are enqueued to the receive side (TG_RCV). One queue for control and error messages and an area for receiving the data messages are provided.

b) The MTG-Ethernet driver

The code of the MTG-Ethernet driver has been included as part of the UNIX Kernel and runs in *supervisor mode*. The MTG-Ethernet driver implements low-level interface to the satellite controller, managing both the transmit and the receive parts. Two sets of interrupt routines are implemented for the MTG servicing:

- *the tick timer and the LANCE Tx side handler.* They initiate the message sending according to the indication contained in the TGT linked list. This list is scanned by the MTG-Ethernet driver which takes care of the physical transmission of the data packets on the LANCE;

- *the LANCE Rx side handler.* It fills the TG_RCV buffers.

The driver works at the highest priority, in order to guarantee accurate timings.

From the driver's point of view, each TGT is described by:

- the TGT starting address (in user space)
- the starting address of the timetable (for the TG_CREATE system call) and the timetable size in a *mtg_ctl* area.

The real time functions of the system, such as packet sending at precise time instants, are performed by the asynchronous part of the driver, which is interrupt driven.

More details can be found in [4], part II.

c) The user/MTG-Ethernet driver interface

The communication between the user and the driver is implemented via standard *ioctl* system calls. The *mtg_ctl union* is filled by the user with the arguments needed by the involved command.

The function *ioctl* performs a wide variety of control functions on devices. The meaning of the argument is driver-specific.

The syntax for giving a command to the driver is:

```
ioctl (special, cmd, arg)
```

where:

- *special* is an open file descriptor referring to a device (mtg_dev);
- *cmd* the command selects the function to be performed;
- *arg* this is the user address of the data required for the particular command.

If successful, the function *ioctl* returns a positive integer value (1). A negative value (-1) indicates an error.

Seven different types of commands are provided:

```
ioctl (special, TG_CREAT, &TGT);
ioctl (special, TG_KILL, &TGT);
ioctl (special, TG_CTL, &CTL);
ioctl (special, TG_OFF, &TGT);
ioctl (special, TG_ON, &TGT);
ioctl (special, TG_RATE, &TGT).
```

The meaning of the commands is:

TG_CREAT -> start a TGT.

The traffic generator relative to each TGT is initiated by issuing the TG_CREAT command The user fills the TGT data packet and the TGT timetable, according to the parameters.given by the operator.

The address of the TGT, the address of the timetable and its size are to the MTG-Ethernet driver via the *mtg_ctl* area.

TG_KILL -> cancel a TGT.

The user gives the address of the TGT to be cancelled to the MTG-Ethernet driver, which in turn gives back the sequence number of the last packet sent. The user then waits until the last data packet is received before erasing its own structures relative to the TGT.

TG_CTL -> send a control message.

The user supplies the address of the control packet to be sent, then erases the control packet.

TG_OFF -> suspend a TGT.

Upon reception of the control message STOP_DAT_PKT (congestion detected) from the FODA system, the user issues the TG_OFF command, giving the address of the interested TGT (datagram data type).

TG_ON -> resume a TGT

Upon reception of the control message RESUME_DAT (congestion overcome) from the FODA system, the user issues the TG_ON command, giving the address of the interested TGT (datagram data type).

TG_RATE -> change the throughput of a TGT

Upon reception of the control message STR_CH_MODIF (reduced or maximum throughput) from the FODA system, the user issues the TG_RATE command, giving the address of the interested TGT (stream data type) and two factors by which the time-table entry members (Dt's and packet lengths) must be multiplied.

When *ioctl* fails, an error number is put by the system in the global variable *errno*. The meaning of the errors follow.

EINVAL occurs when:

- the user-driver protocol is not respected;
- the receive buffer had been already allocated and the user tries to allocate it again;
- no buffer is currently allocated and the user issues the TG_GETRBUF command;
- the tick length is out of range.

ENOSPC occurs when the driver constant TGDPOOL_SZ is too small.

ENOMEM occurs when the driver constant TTPOOL_SZ is too small.

In the following, the error codes passed from the driver to the user control receive areas are listed.

TG_SLOW (code = 0x01)	User is too slow.
TE_LCOL (code = 0x02)	Late collision on transmission.
TE_LCAR (code = 0x03)	Loss of carrier on transmission.
TE_RTRY (code = 0x04)	Retry error on transmission.

TE_SEMA (code = 0x05) Too much backlog.

d) The memory structures for data exchange

A contiguous buffer is allocated by the driver, and is divided in two areas managed as circular queues.

the control-error queue

The error messages are enqueued in the same queue used for the incoming FODA control messages. The driver enqueues error messages as control messages, e. g. by using the same data structure with a different data type.

the data packet queue

The data packets are enqueued in a separate queue for incoming FODA data messages.The MTG-Ethernet driver is responsible for proper message identification and enqueueing.

the contiguous area allocation

The buffer manager handles the contiguous area simply by using the TG_GETRBUF, TG_ASKRBUF, TG_FREERBUF commands of the driver. Each queue consists of a fixed number of buffers. CMP_SIZE is the size of the control-error queue, whose buffers have fixed length CTL_LENGTH. The data packet queue size is DPP_SIZE, and their buffers have fixed length TG_MAXDPL.

The contiguous area (physically contiguous space) for this set of buffers is allocated whole at once at the startup time by using the buffer-related commands of the driver:

- TG_GETRBUF allocates contiguous space.
- TG_ASK_RBUF returns the physical address and size of the allocated space.
- TG_FREERBUF releases the allocated space.

The MTG/R uses these commands only in the *getrbuf()* and *freerbuf()* routines. The sequence for allocating all the needed buffer space follows.

- The contiguous buffer is allocated and mapped in the user space using the *getrbuf()* routine.
- The control-error (CMP) array is allocated.
- The data (DPP) array is allocated.
- The user address of the contiguous buffer and the user addresses of the CMP and DPP arrays are eventually passed to the MTG-Ethernet driver by issuing the TG_INIT command.

e) The algorithm for data exchange

The control-error queue and data queue are implemented as circular linked lists of buffers. During the initialization, all the buffers are marked for write (free).

When the driver receives a message, it takes the buffer from the beginning of the queue, and set its address in control-error or data arrays, depending on the packet type. The buffer then results marked as "readable" (busy). The driver then takes the *next* buffer as the *current* buffer (Fig 6.).

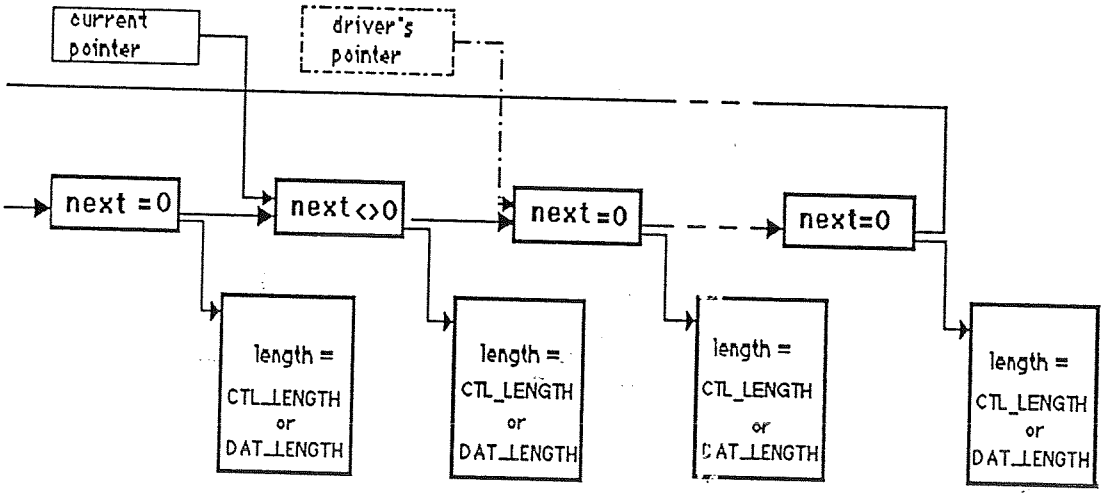


Fig 6. The receive queue area

The MTG/R scans the "readable" addresses (nonzero values) in the control-error and data queues, starting from the *current* pointer. If the current pointer has a nonzero value, the buffer is read. After the processing the address is cleared, so the buffers results "writable" (free again).

Summarizing:

- the driver sets the buffer marked as busy (nonzero value of the pointer);
- the user sets the buffer marked as free (zero value of the pointer).

This mechanism is used for collecting all the incoming FODA packets. The control packets, the error packets and the headers of the data packets are stored in an archive file. The control and error packets are properly processed. At the end of the session the archive file contains

the complete information about the session run. These data are used during the post-run analysis.

f) MTG/R organization

In order to provide an environment for traffic generation and measurement, to exhaustively test the FODA system, an efficient MTG/R must be implemented. To get more information, refer to chapter 3.

Fig. 7 shows the internal organization of the MTG /R system.

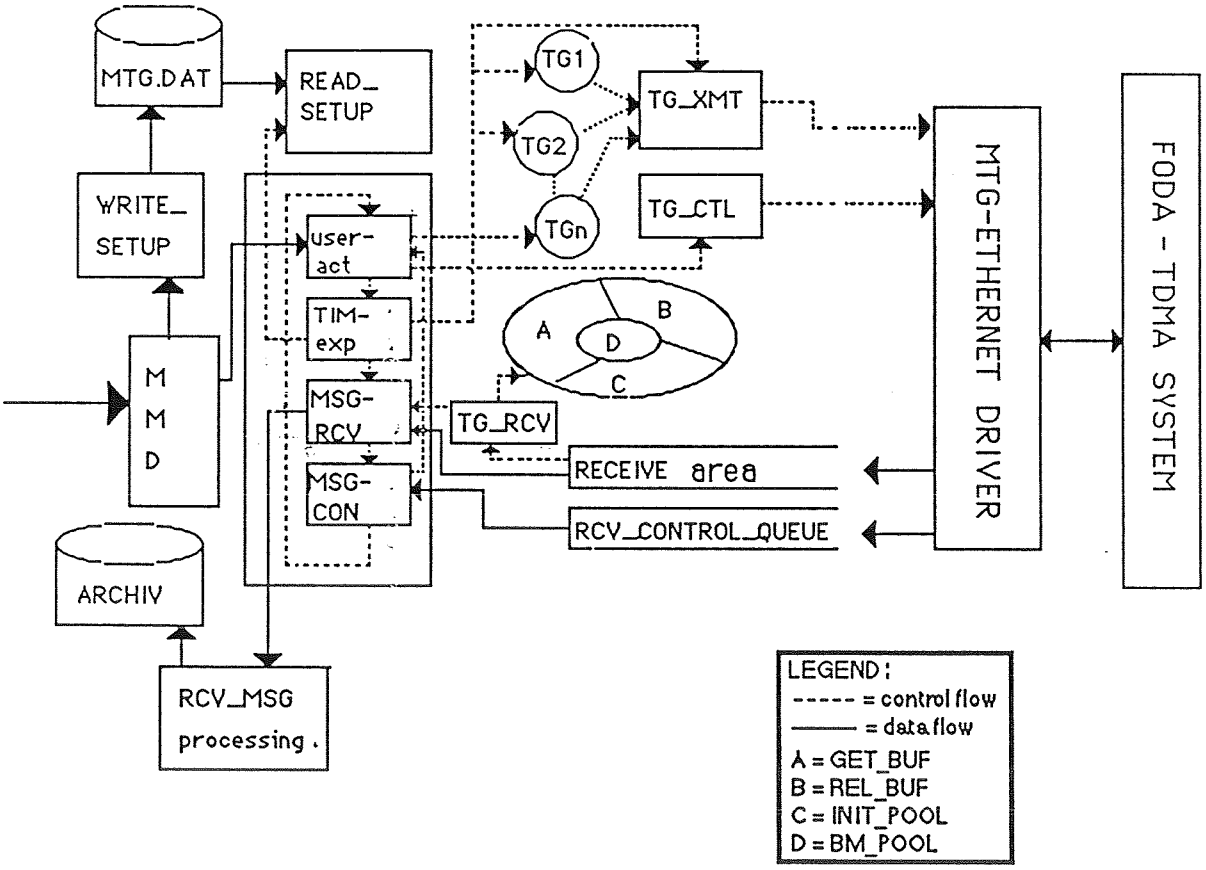


Fig 7. MTG/R organization

In the following, the MTG/R modules are briefly described.

MTG_INI

This part provides the basic information needed to define the traffic generation session and to run it. Each traffic generator (TG) is

defined in an input file describing the general characteristics of the system and giving the detailed description of the traffic generators connected to each station.

DATA TYPE:	Stream / Interactive / Bulk
DISTRIBUTION:	Fixed Rate / Random / Poisson / Voice
PACKET SIZE:	Fixed Size / Variable Size
MEAN THROUGHPUT:	Number of Kbit/s
PATTERN:	Fixed Pattern / Random / Incremental
START TIME:	Absolute / Relative / Operator request / Poisson
DURATION:	Absolute / Relative / Operator request / Poisson
COS:	Data Class of Service
OPTIONS:	Options mask
ADDRESS:	Myself / To a particular station / Broadcast

Fig 8. The TGD parameters

The various TGTs are created during the set-up session, where the operator defines all the required traffic parameters. The READ_SETUP and WRITE_SETUP routines are used to check inputs and set default values for non-defined parameters. The parameters are stored in the input file which supports all the possible combinations of the traffic parameters. Each combination of traffic types is stored in the Traffic Generator Descriptor, (TGD). Many of them can be defined in the input file (Fig 8). Each TGD is record-structured. The record layout is given below :

```
struct name {
     unchar type;          /* <distribution> type */
     unchar dist;          /* distribution */
     unchar typs;          /* <packet size > type */
     uint pkts1;           /* packet size */
     uint pkts2;           /* packet size */
     uint thro1;           /* throughput */
     uint thro2;           /* throughput */
     unchar typp;          /* <pattern> type */
     uint patt;            /* pattern */
     unchar typj;          /* <jitter> type */
     uint jitter;          /* jitter */
     unchar typc;          /* <class of service > type */
     unchar class;         /* class of service */
     unchar typo;          /* <option> type */
     unchar opt;           /* option */
     unchar typa;          /* <address> type */
     unchar addr;          /* address */
     unchar typt;          /* <time> type */
     time_t s_t;           /* start time */
     time_t e_t;           /* stop time */
};
```

The typedef are those defined in the standard C language for the UNIX environment.

MTG_MMD

The Man-Machine Dialogue (MMD) module provides the necessary interface with the operator.

The MTG/R is activated by an explicit operator request. The opening screen contains the basic information about the default values. During the question-answering session, requests and options are acquired by the system and checked for consistency. Upon reception of the RUN command, the MTG/R proceeds automatically.

```
          MULTIMEDIA TRAFFIC GENERATOR
          CNUCE - CNR  Version 1.0  1989

Help [y/n]:? y

Basic MTG Information
Max. number of TGDs in Input file - 10

              Contents  of the TGDs
                   TGD #1          TGD #2          ...
TYPE:              stream          stream          ...
DISTRIBUTION:      fixed rate      fixed rate      ...
          :
          :
          :

Run MTG [y/n]:? y
All TGs or a specific one [a/s]:? a
Please wait - MTG is now in the pre-run phase.
Session start time 10:35 -
The output file name is MTGout.
```

Example startup screen

MTG_SCH

The scheduler is the main MTG/R module. It is responsible for the coordination of all the following user activities:

a) usr_act

This routine processes all the user actions regarding the TGT creation, abortion and redefinition of the TGT parameters.

b) mtg_rcv

This routine processes all the incoming messages.

c) time_exp

This routine is responsible for all the activities scheduled by the system clock (e. g. TG starting, TG ending, timeouts etc.).

TG_XMT

The transmission side of the MTG is organized as an array of pointers to structures, which collect all the TGTs involved in the traffic generation.

Each structure contains (Fig 9.):

- a pointer to the timetable,

- the GA-FO header,
- the TG header,
- the timetable.

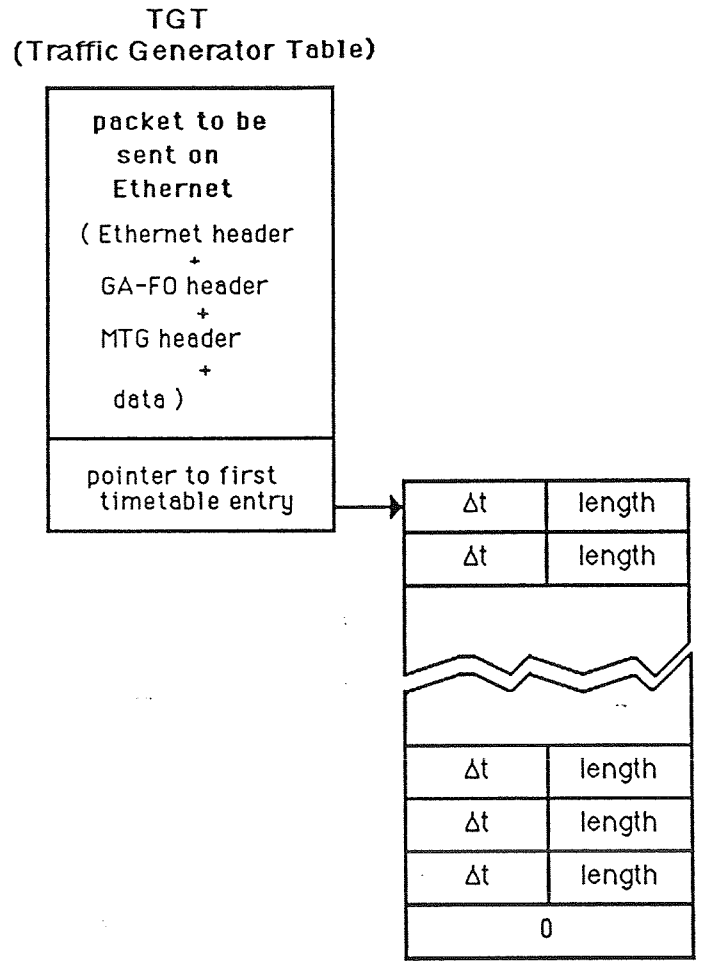


Fig 9. TGT structure

The traffic generation relative to each TGT is initiated by issuing the TG_CREAT command. The traffic generation is ended by issuing the TG_KILL command.

g) Data Structures

Data structures are depicted in two separate parts:
- data structures regarding the communications with the FODA-TDMA controller [1];
- data structures regarding the MTG/R.

Formats of messages exchanged with the FODA-TDMA controller

a) Control messages:

(GATEWAY ----> FODA SYSTEM)

*Stream channels request (code = 0x83)
*Stream channels relinquish (code = 0x89)
*Stream channels modification reply (code = 0x87)
*Build a group of users (code = 0xA0)
*Modify a group of users (code = 0x8A)
*Cancel a group of users (code = 0xB0)

(FODA SYSTEM ----> GATEWAY)

*Stream channels granted / refused (code = 0x81)
*Stream channels modification request (code = 0x85)
*Stop sending datagram packets (congestion detected, code = 0x98)
*Resume sending datagram packets (congestion overcome code = 0x88)
*Destination station(s) not reachable (code = 0xD0)
*Destination station(s) now reachable (code = 0xC1)
*Build a group of users replay (code = 0xC2)
*Extend a group of users reply (code = 0xC3)
*Error in request < error type > (code = 0xC4)

The format of a generic control message is shown in Fig.10.

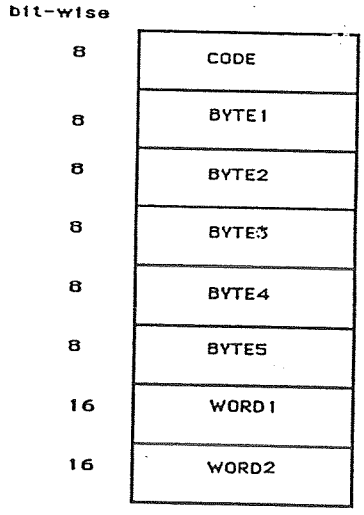


Fig 10. GA-FO header format for control messages

where the meanings of BYTEx and WORDx depend on the code [5].

b) Data messages:

(GATEWAY ----> FODA SYSTEM)

-Send stream data without CRC check (code = 0x53)
-Send stream data with CRC check (code = 0x73)

-Send bulk data without CRC check (code = 0x51)
-Send bulk data with CRC check (code = 0x71)
-Send interactive data without CRC check (code = 0x52)
-Send interactive data with CRC check (code = 0x72)

(FODA SYSTEM ----> GATEWAY)
-Stream data received from satellite without CRC check (code = 0x43)
-Stream data received from satellite with CRC check (code = 0x63)
-Bulk data received from satellite without CRC check (code = 0x41)
-Bulk data received from satellite with CRC check (code = 0x61)
-Interactive data received from satellite without CRC check (code = 0x42)
-Interactive data received from satellite with CRC check (code = 0x62)

The format of a generic data packet in shown in Fig 11.

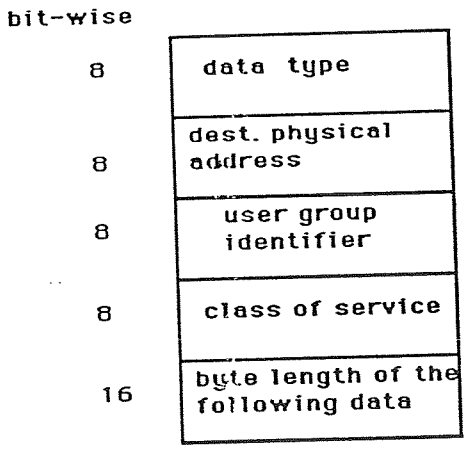


Fig 11. GA-FO header format for generic data packets

Formats of messages regarding MTG/R

c) Traffic generator header

The traffic generator header contains timing and other parameters relative to the MTG and the FODA systems. It is structured as shown in Fig 12.

The meaning of the fields is:
SEQ_num: calculated and set by the MTG-Ethernet driver;

MONITOR: monitor bit set by MTG/R;
FTX_ERR: XMT error from Ethernet;
FRX_ERR: RCV error from Ethernet;
TGT_ID: Traffic Generator ID set by MTG/R;

BC_RATE: bit and coding rate set by FODA;
EBN0: Eb/No at Rx time;
FRX_LEV: down signal level;
FTX_LEV: up signal level;

QUEUEL: Queue length [Elementary Slots];

MTX_time: Time of data transmission to Ethernet by the MTG/R (microticks);
FTX_time: Time of the data reception from Ethernet;
STX_time: Time of the data transmission to satellite;
SRX_time: Time of data reception from satellite ;
FRX_time: Time of data transmission to Ethernet;
MRX_time: Time of data reception from Ethernet by the MTG/R (microticks).

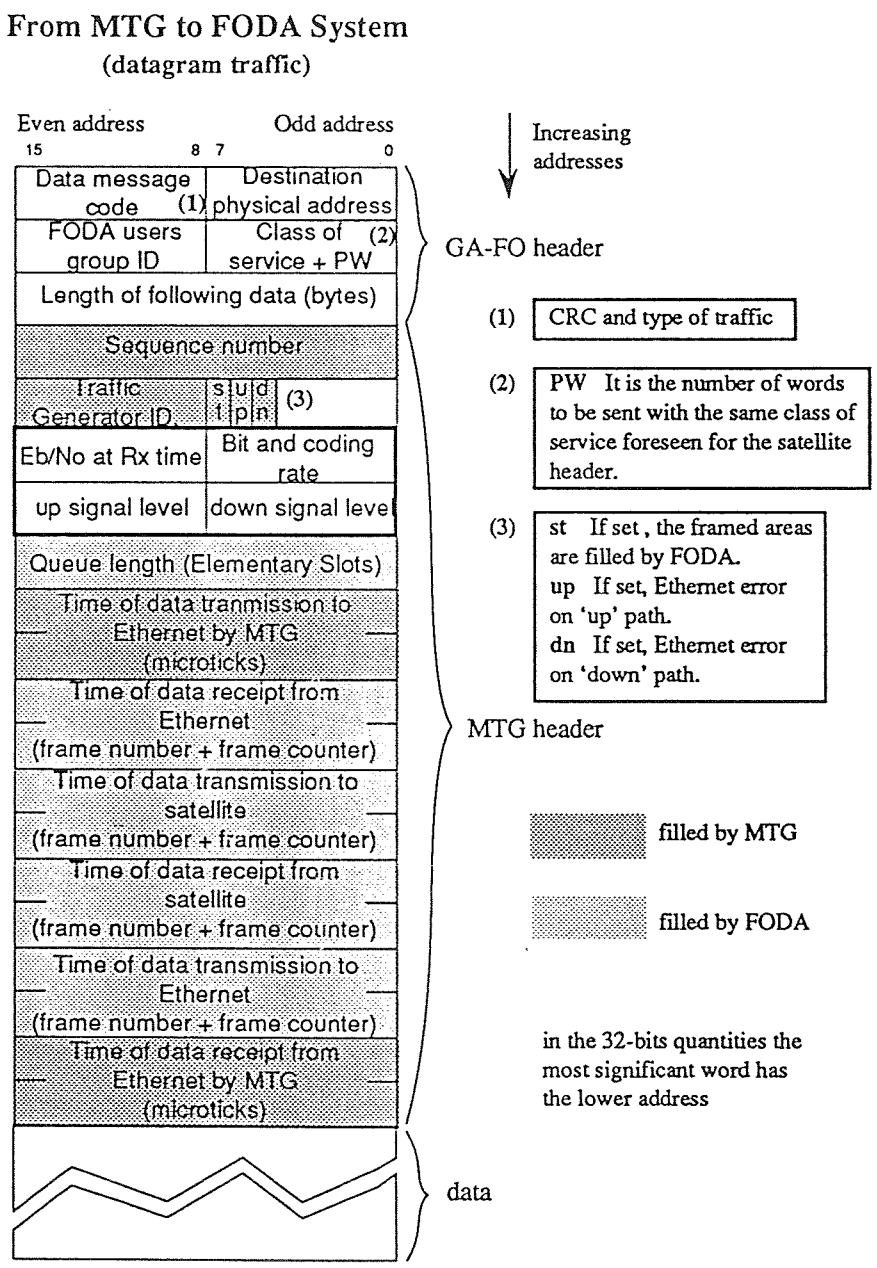


Fig 12 a. Data sent to FODA from MTG/R (datagram traffic)

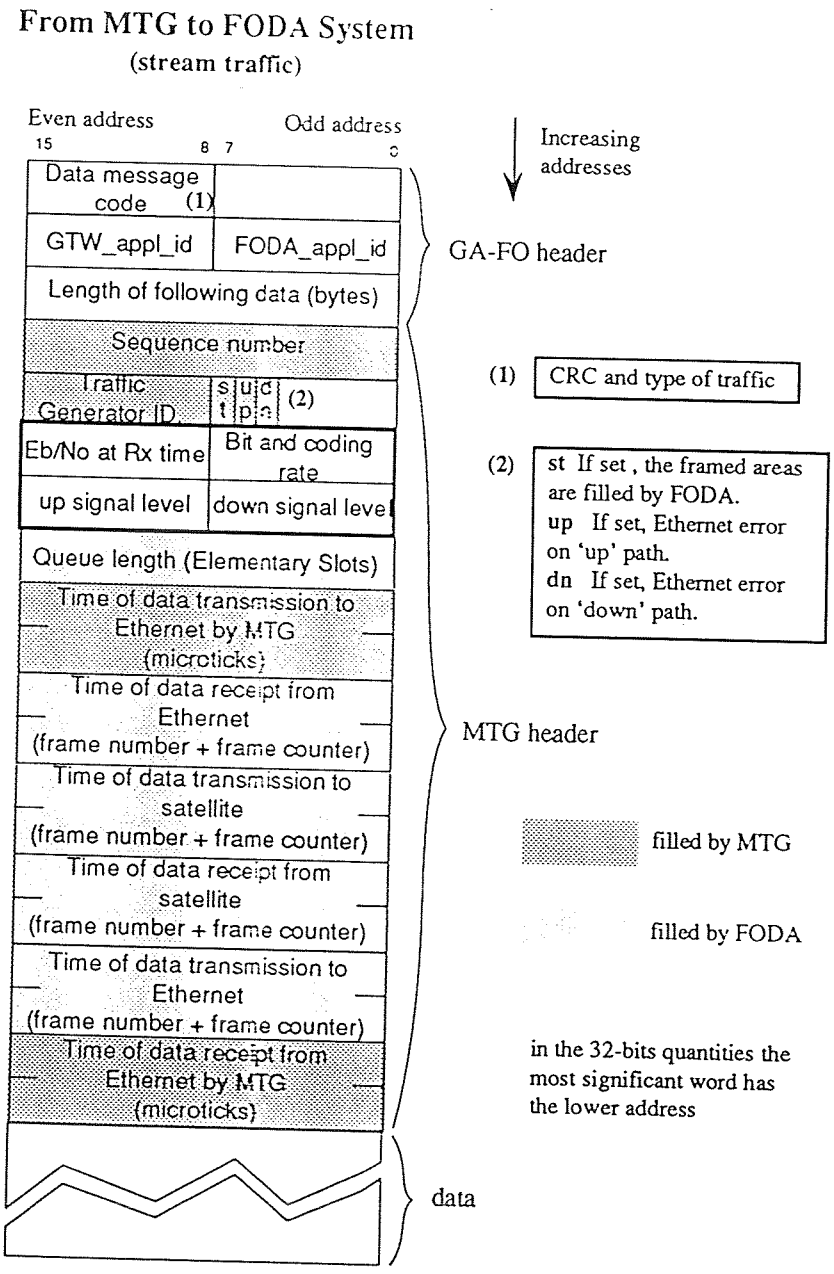


Fig 12 b. Data sent to FODA from MTG/R (stream data)

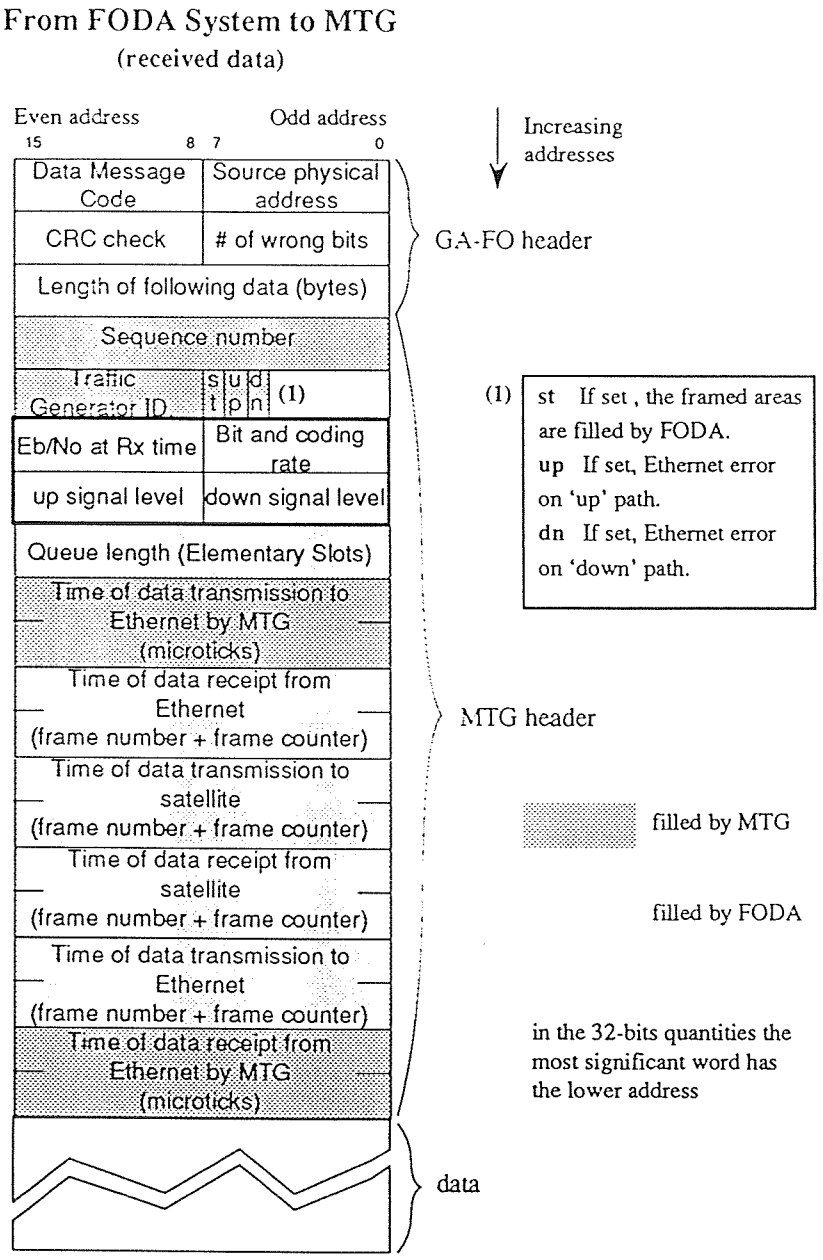


Fig 12 c. MTG/R received data from FODA

Fig 13 shows the control and data message formats.

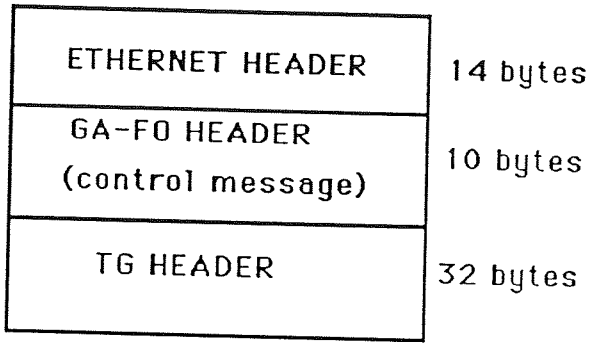


Fig 13 a. Control message format

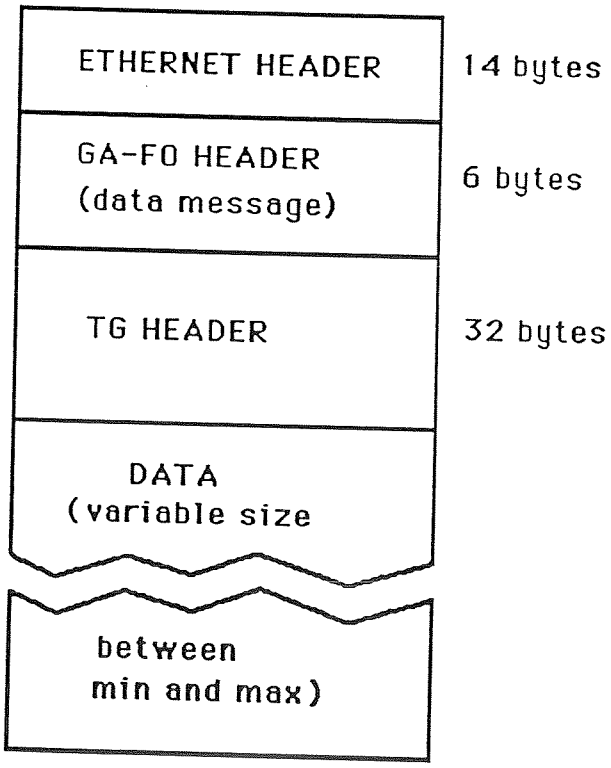


Fig 13 b. Data message format

3. MTG/R USER GUIDE

The *MTG/R (Multimedia Traffic Generator/Recorder)* software consists of the following set of modules, written in C language:

mtg_mmd.c	man-machine dialogue
mtg_ini.c	initialization routines
mtg_sch.c	scheduler routines
mtg_rcv.c	message processing routines
all_list.c	list manipulation routines

The internal MTG/R organization is shown in Appendix A. The system flow chart is sketched in Appendix B.

3.1 Man-Machine Dialogue (mtg_mmd)

The Man-Machine Interface is implemented as a menu driven dialogue, controlled by the MMD module. The Curses-Terminfo System V/68k package is used as the essential implementation tool [7]. The windowing concept is used for information exchange. Three windows are always present on the screen. On the upper side of the screen, a thin window with the basic MTG/R information is placed. The central part of the screen is reserved to display the input file contents and the system messages. The third window, placed at the bottom of the screen, shows the list of the commands available to the MTG/R user. The following commands are currently implemented:

r : start the traffic generator
(a specific TG or all TGs at once).
e : stop the traffic generator (a specific one or all).

The traffic generation can be started or stopped at any time by using one of the following commands.

a : add a new generator
d : delete or modify the TGT parameters
l : display the TGT parameters
q : exit the MTG/R system.

Each new session (MTG/R run) writes on the special file *MTGout*, writing the data from the very beginning of the session.

The *MTGout* file contains the headers of all the received data packets and all the control messages. The header of the MTGout file contains the copy of the current session input file.

During the session all the data are written sequentially in their own arrival order. The MTGout file ends with a special header containing the CODE = 0.

To prevent the file overwriting the user is warned by the following message:

```
*** Warning *** This run will overwrite previous
results.
          Please save your archive file.

 Continue or exit [c/e]?
```

c - old archive file will be overwritten
e - exit from the session to save the old archive file

```
 List input files [ y/n ]?
```

y - list the input files content
n - continue

```
 Enter input file name:
```

Enter the input file name, which will be read by MTG/R

```
 Display, modify or run [ d/m/r ] ?
```

d - display the chosen input file contents
m - modify the input file parameters. An edit session of the input file is forced, so the user must exit issuing the ":wq" command after the modifications.
r - run the session immediately

```
Run [ y/n ]?
```

y - everything is ok and the MTG/R session may start
n - exit from the MTG/R system.

Input file description

On top of the input file a variable size record is present, reserved for the user comments. It is delimited by the character "#". A general information record plus a variable number of TG records follow, each one describing a traffic generator. The end of the file is flagged by the character " $ ".

All the records contain the name of each variable followed by one or more integer values. The name of the variables are for user convenience only. Its length can be from 1 up to 4 characters. All the items are

delimited by a blank character. All the records are delimited by a CR [carriage/return] character.

The general information record describes some parameters common to all the TGs.

MSEM - Semaphore threshold to get warned about its crossing.
TICK - Tick length in μs.
DEST - Ethernet destination address (6 hexadecimal digits).

Each traffic generator is described by the set of the parameters (TG records) described below.

- DATA TYPE (TYPE):

1-*stream* 2-*bulk* 3-*interactive*

1 - real time traffic. It requires a guarantied bandwidth on the satellite channel.
2 - non real time traffic (datagram). The bandwidth allocated on the satellite channel and, consequently, the transmission delay, depend on the overall system loading conditions. Interactive traffic gets higher transmission priority than the bulk traffic.

Additional parameters define in detail the TG characteristics, providing a wide variety of possible traffic combinations. Let us see the meaning of all the fields and their appropriate values. In order to get a proper working of the MTG/R, the data in the input file must be carefully checked.

- DISTRIBUTION (DIST):

This parameter describes the temporal properties of the packets produced by the generator.

1-*fixed* 2-*random* 3-*Poisson* 4-*voice*

1 - fixed. The inter-arrival time between packets is constant.
2 - random. The inter-arrival time between packets is constant, while the length of the packets takes random values in a given range.
3 - Poisson. The inter-arrival time between packets is determined by the generation of pseudo-random numbers following a negative exponential distribution.
4 - voice. The information flow of a packetized speech is simulated. No packet is generated if the "silent" period is long enough. The talk-silent periods are sorted according to a statistical distribution, relative to the English language.

- PACKET LENGTH (TYPS):

Packet size.

1-*fixed* **2**-*variable*

1 - packet length is fixed and equal to: Ethernet header size + GA-FO header + MTG/R header + fixed size of real data.

2 - packet length is variable and equal to: Ethernet header size + GA-FO header + MTG header + variable data size.

The two following values express the maximum (packet_size1) and the minimum (packet_size2) values respectively. If option **1** is selected, these two values must be equal.

- MEAN THROUGHPUT (THRO):

Maximum and minimum values of the throughput [Kbit/s] respectively.

- PATTERN (TYPP):

1-*fixed* **2**-*random* **3**-*incremental*

1 - pattern is fixed and equal to the chosen value

2 - pattern is a random number in the range $0 \div (2^{16}-1)$

3 - pattern is incremented word by word, starting from an initial value.

The second parameter expresses the pattern Hex value (if option **1** is selected) or the initial Hex value (if option **3** is selected).

- START/STOP TIME (TYPT):

1-*absolute time* **2**-*relative time* **3**-*user* **4**-*Poisson*

The six following parameters express the start (hh mm ss) and the stop time (hh mm ss) for the options **1** and **2** and the *call* and *duration* times, respectively, for the option **4**. If option **3** is selected, the TG starts and ends on user command only.

The option **4** si used to simulate phone calls. The start and the duration times are random numbers with a negative exponential distribution (see 3.2) with mean values equal to the *call* and *duration* times respectively.

- CLASS OF SERVICE (CTYPE):

1÷4-Data class of service (requested bit error rate)

1- $BER \leq 10^{-8}$

2- $10^{-8} \leq BER \leq 3 \cdot 10^{-7}$

3- $3 \cdot 10^{-7} \leq BER \leq 3 \cdot 10^{-5}$

4- $3 \cdot 10^{-5} \leq BER \leq 3 \cdot 10^{-3}$

The decimal number that follows indicates the number of data words — to be sent on the satellite — with the same class of service as the satellite header (typically COS=1). It allows to send the terrestrial network packet headers with high protection, in order to ensure the delivery of the packets even if the needed BER of the data is not very low.

- OPTION MASK (OMSK):

XX - sum of all the selected options from the following option list in Hex format. A decimal number follows for a further specification of the options.

0X80 - Set **st** flag.
The st flag is set in the MTG header, so the FODA system fills the following fields:
- Eb/No at Rx time
- Bit and coding rate
- Transmit power level
- Receive power level.

0X40 - Send data with CRC.
Data are requested to be sent on satellite, by the FODA system, appending the CRC. The CRC is checked on reception and a flag is set in the GA-FO header if it is wrong.

0X20 - Compute number of wrong bits.
The received packets contents are compared with the transmitted ones and the number of wrong bits is set in the GA-FO header. This option allows the estimation of the bit error rate.

0X10 - Record data every n^{th} .
One packet header, every *n* received, is recorded onto the MTGout file.
n is specified in the next parameter.
If this bit is cleared, *n* is assumed equal to 1.

- ADDRESS (TYPA):

1-*myself* **2-***selected station* **3-***broadcast*
The Hex value of the selected station address is entered.

3.2 Initialization (mtg_ini)

This module creates two linked lists:

- The *Traffic Generator Descriptor (TGD)* list according to the input file. It describes the general characteristics of the traffic generators.
- The *Traffic Generator Table (TGT)* list, with several uncorrelated traffic generators. Each traffic generator can produce packets with one of the following inter-packet time distributions:

Fixed The intervals of time (**Dt**) between packets are constant.

$$Dt = 8 \cdot \frac{packet\_size1}{throughput1} \quad [ms] \qquad (1)$$

where: packet_size1 [bytes]
throughput1 [Kbit/s]

Random The length of the packets (**L**) is a uniform random variable in a range between a minimum and a maximum value.

$$L = packet\_size1 + (packet\_size2 - packet\_size1)\ RAND \qquad (2)$$

where RAND is a uniformly distributed random variable in the (0,1) interval.

The interval of time (**Dt**) between packets is given by:

$$Dt = 8 \cdot \frac{packet\_size1 + packet\_size2}{2 \cdot throughput1} \quad [ms] \qquad (3)$$

where: packet_size1 and packet_size2 [bytes]
throughput1 [Kb/s]

Since the produced numbers are not truly random at all, as each one depends in a definite way on its predecessor, it should be spoken of pseudo-random numbers. If the seed begins with same value each time the program is run, then the whole sequence of pseudo-random numbers is exactly the same. For that it is preferable to begin by setting the seed to a random value. So the system time is used as initial seed number and then the got random number is used as seed for the next random number production.

Poisson The intervals of time (**Dt**) between packets follows a negative exponential distribution.

The probability density function of the Poisson random variable X, with mean $E[X] = t$ and variance $VAR[X] = t^2$ $(t > 0)$, is:

$$f(x) = \frac{1}{\tau} e^{-\frac{x}{\tau}} \qquad \text{for } x \geq 0 \qquad (4)$$

and

$$f(x) = 0 \qquad \text{elsewhere.}$$

The cumulative distribution function is:

$$F(X) = \int_0^x \frac{1}{\tau} e^{-\frac{x}{\tau}} dx = 1 - e^{-\frac{x}{\tau}} \qquad (5)$$

Applying the inverse transformation to the (5) it is possible to generate exponential random variables [6]. Letting :

$$r = F(X) = 1 - e^{-\frac{x}{\tau}} \qquad \text{for } 0 \leq r < 1,$$

$$F^{-1}(r) = X = -\tau \log (1-r)$$

If the number r is uniformly distributed in the (0,1) interval, then the argument (1 - r) has the same distribution.

Finally **Dt** can be computed by:

$$\mathbf{Dt} = - Dt\_in \log RAND \qquad (6)$$

where:

$$\mathbf{Dt\_in} = 8 \frac{packet\_size1}{throughput1} \quad [ms]$$

packet_size1 [bytes]
throughput1 [Kbit/s]

and RAND is a random number in the (0,1) interval.

Voice The *talk* and the *silent* periods are sorted according to the statistical distribution contained in proper tables. Each entry in these tables is made using a computed random number as a displacement. During the *talk* periods, the interval of time **Dt** is computed exactly like in the **fixed** case.

3.3 Scheduling (mtg_sch)

This module is the main scheduling loop. It :

- creates traffic archive files
- handles buffering
- serves:
  a) terminal entry

b) message reception

in an endless loop.

The scheduler handles all the above events. At the end of each routine the control is returned to the scheduler.The occurring of each event causes the MTG/R to switch into one of the following states:

INI_S	0	Initial state
WAIT_ACK	1	Wait for ACK/REF
ACK_RCV	2	ACK/REF received
SEND_DATA	3	Send data without waiting timer start
WAIT_ALARM_START	4	Wait timer start
WAIT_ALARM_STOP	5	Wait timer stop
ALARM_EXP	6	Timer expired
WAIT_MACK	7	Wait for modification ACK
MACK_RCV	8	ACK received
TRY_AGAIN	9	Disconnect received
DAT_ON	10	Datagram active
DAT_OFF	11	Datagram not active
CON_DET	12	Congestion detected
CON_OVER	13	Congestion overcome
USR_GROUP	14	Build users group
WAIT_GACK	15	Build users reply group
USR_EXT	16	Build extended users group
USR_EACK	17	Build extended users group ACK
CLOSING	18	State after KILL and before TGT_END state
TGT_END	19	TGT run finished
USER_END	20	TGT stopped by user
ERR_TYPE	21	Error type

3.4 Reception (mtg_rcv)

This module manipulates two separate buffer queues:

a) the control-error queue

b) the data packet queue

The **mtg_rcv** module scans the queues and reads the messages, calling the proper routines.

3.5 Data Base Management (all_list)

The **all_list** module manipulates all the linked lists in the MTG/R:

- TGT list
- TGD list
- TCB list

This module sorts the TGD and the TGT in the order in which they occur in the input file. The TCBs are sorted in increasing order using the start and the stop time of the TGTs. Using the *find* and the *delete* routines, with the proper identifier (TGD, TGT or TCB), the TGD, TGT and TCB tables are found and deleted respectively.

3.6 Error Messages

During the MTG/R run, two types of errors may occur: fatal and non-fatal. The fatal errors cannot be handled by the MTG/R software and MTG/R must terminate.

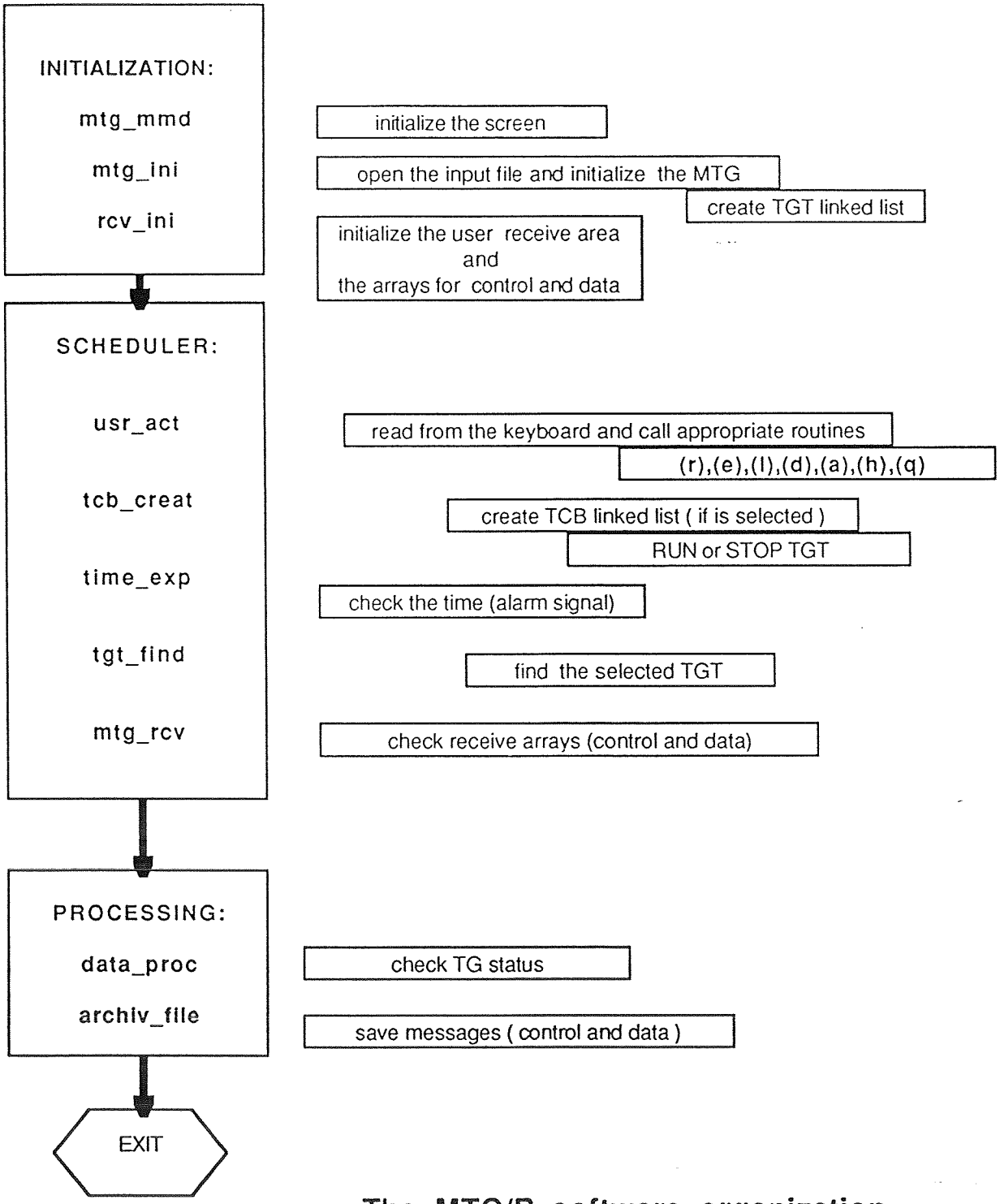
Fatal errors:

1) F_ALLOC	Can't allocate receive buffer
2) F_OINPUT	Can't open input file
3) F_OSPECIAL	Can't open special file
4) F_OARCHIVE	Can't open archive file
5) F_NOT_MEMORY	Not enough memory
6) F_ALLOCRBUF	Failure of the receive buffer allocation
7) F_LOCK	Lock/Unlock Error
8) F_SLOW	User is too slow
9) F_SEMA	Too much backlog

Non-fatal errors:

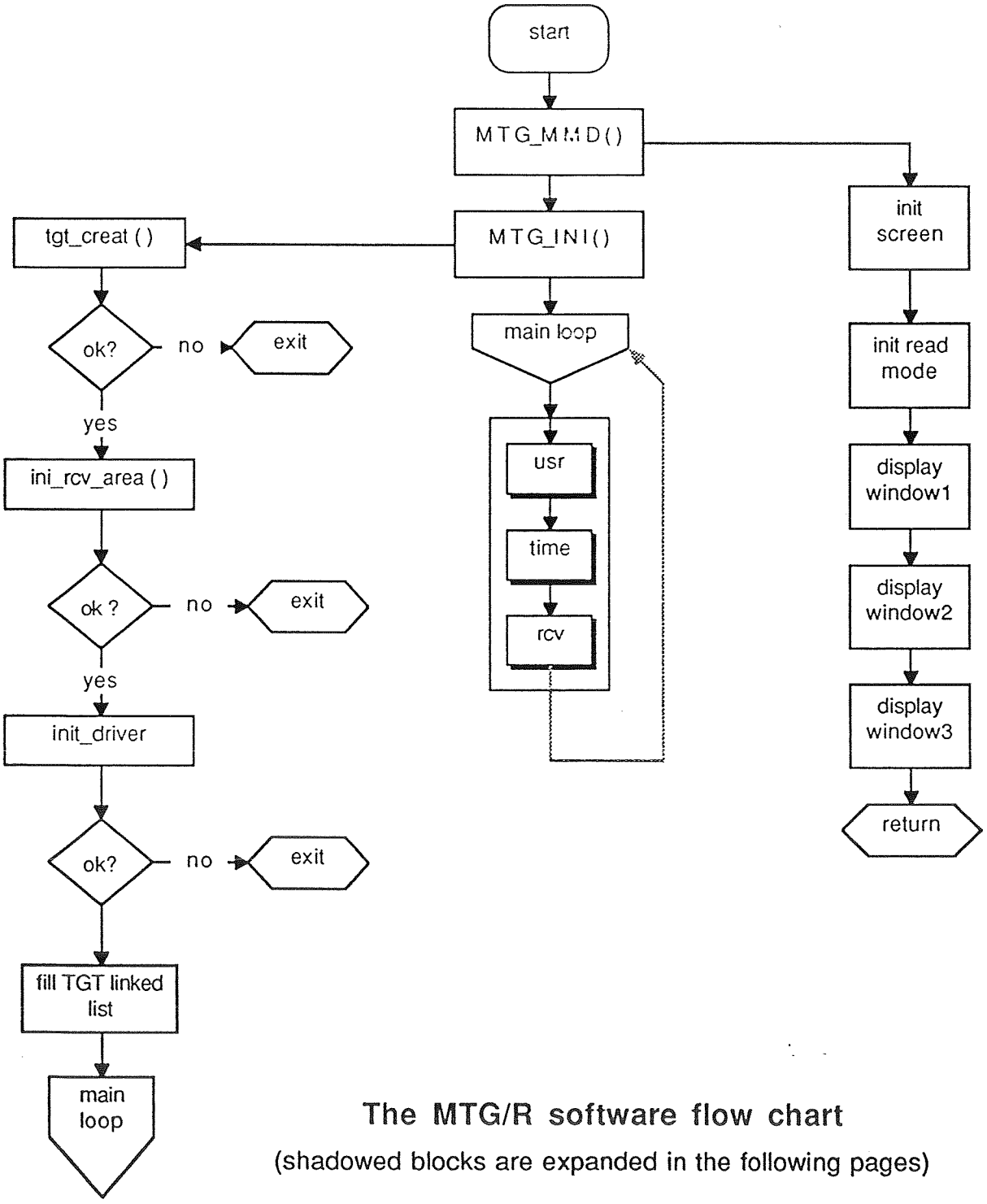
1) E_ENTRY	Invalid entry
2) E_DATA	Invalid data type
3) E_DIST	Invalid distribution type
4) E_PATT	Invalid pattern type
5) E_OPTION	Invalid option type
6) E_ADDRESS	Invalid address type
7) E_TIME	Invalid time type
8) E_JITTER	Invalid jitter
9) E_AACT	TGT already active
10) E_NFOUND	TGT not found
11) E_NACT	TGT not active
12) E_FTCB	TCB flag error
13) E_NCTL	Not error nor control message
14) E_NDATA	Not data message
15) E_STATE	Invalid TGT state
16) E_SEQ	Sequence error
17) E_CURSES	Curses error
18) E_TEXP	Time expired error
19) E_REM	TCB not removed
20) E_STR	Stream control error

APPENDIX A



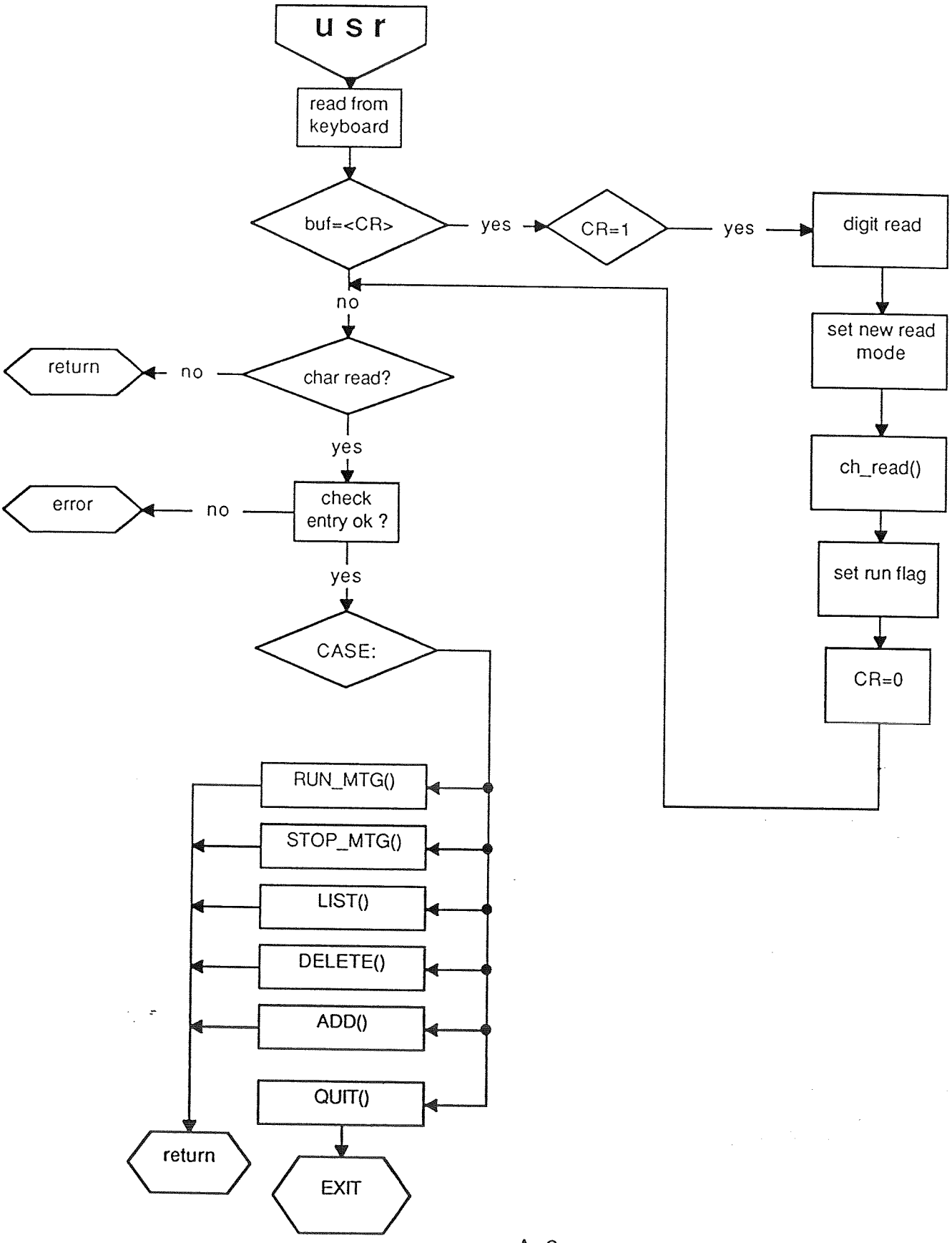
The MTG/R software organization

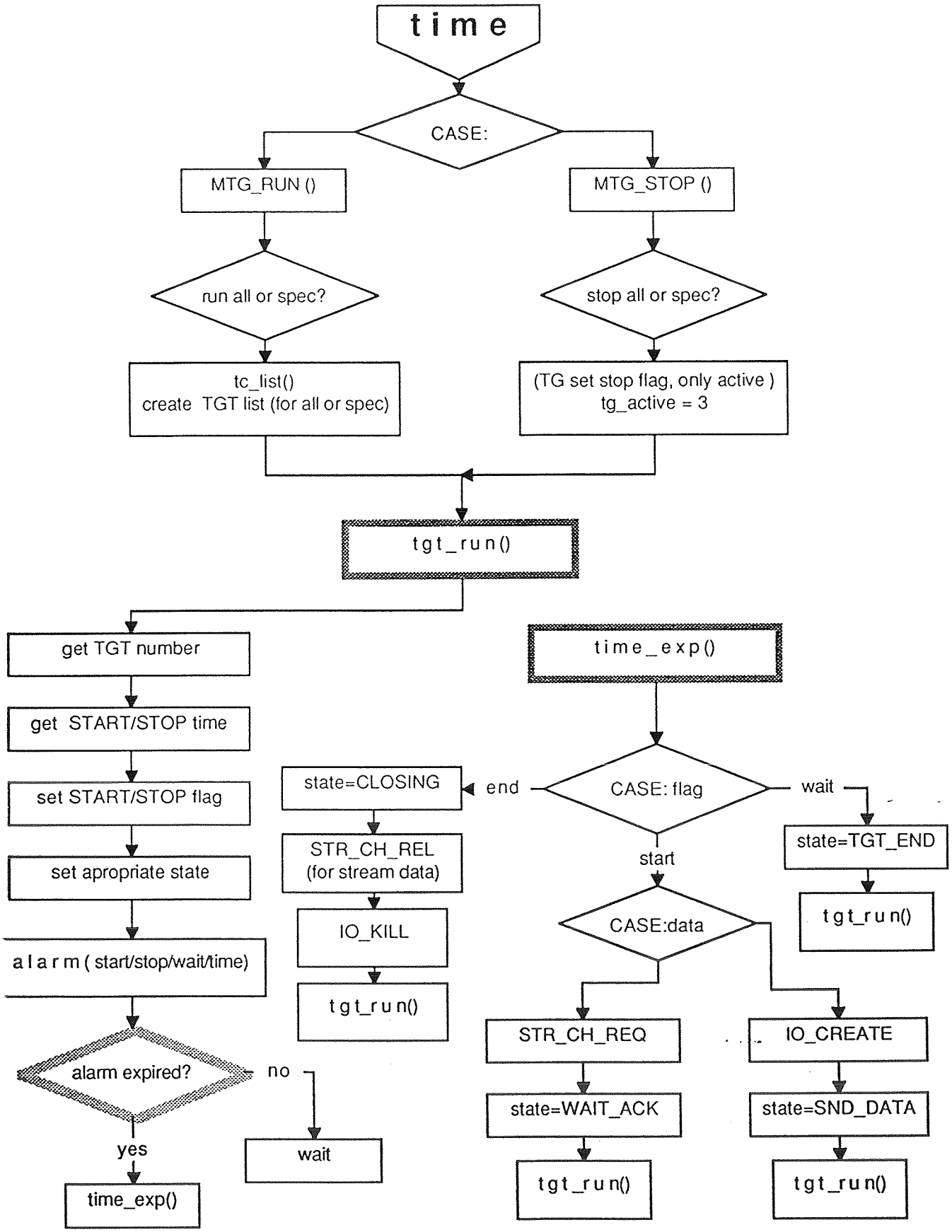
APPENDIX B

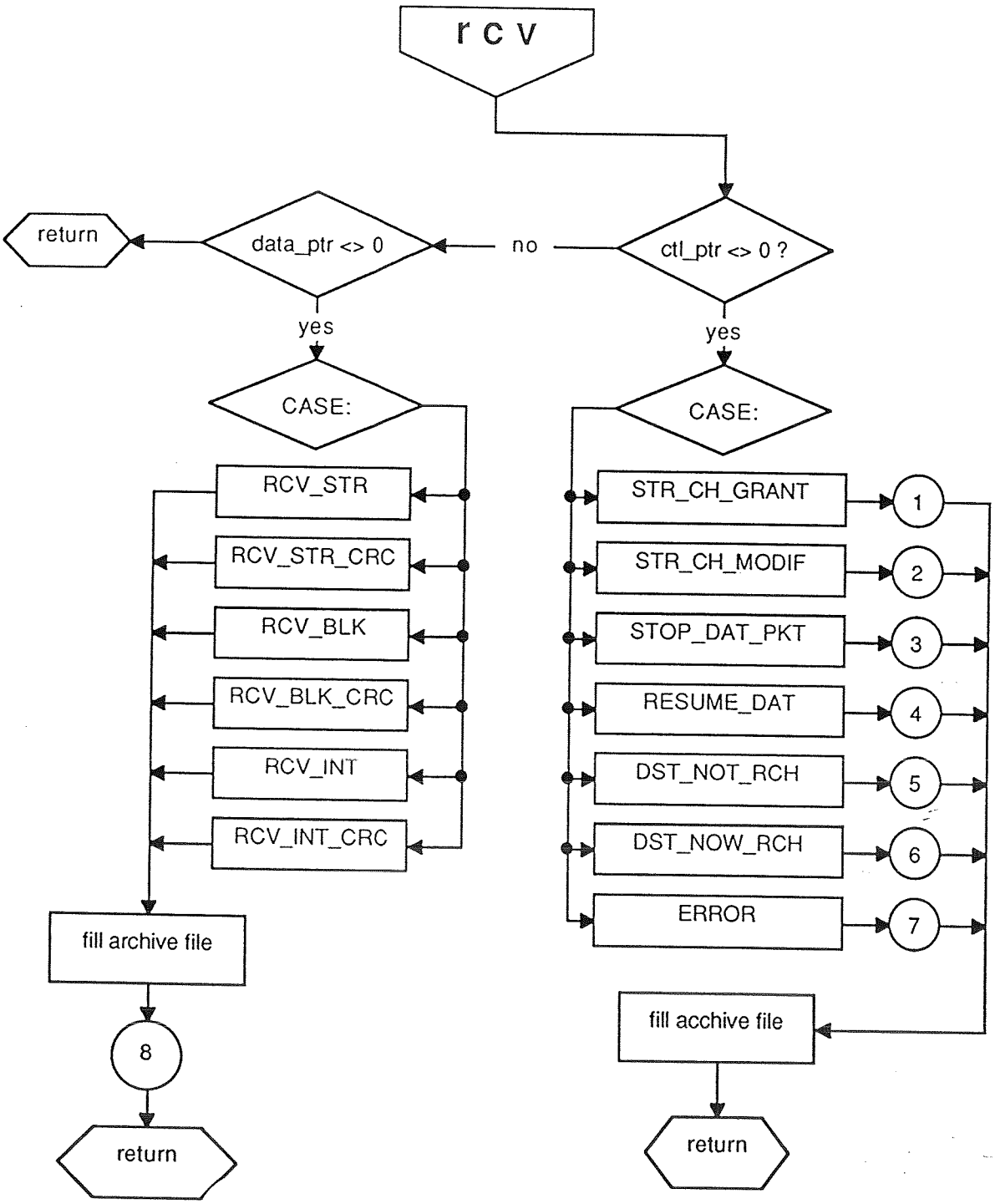


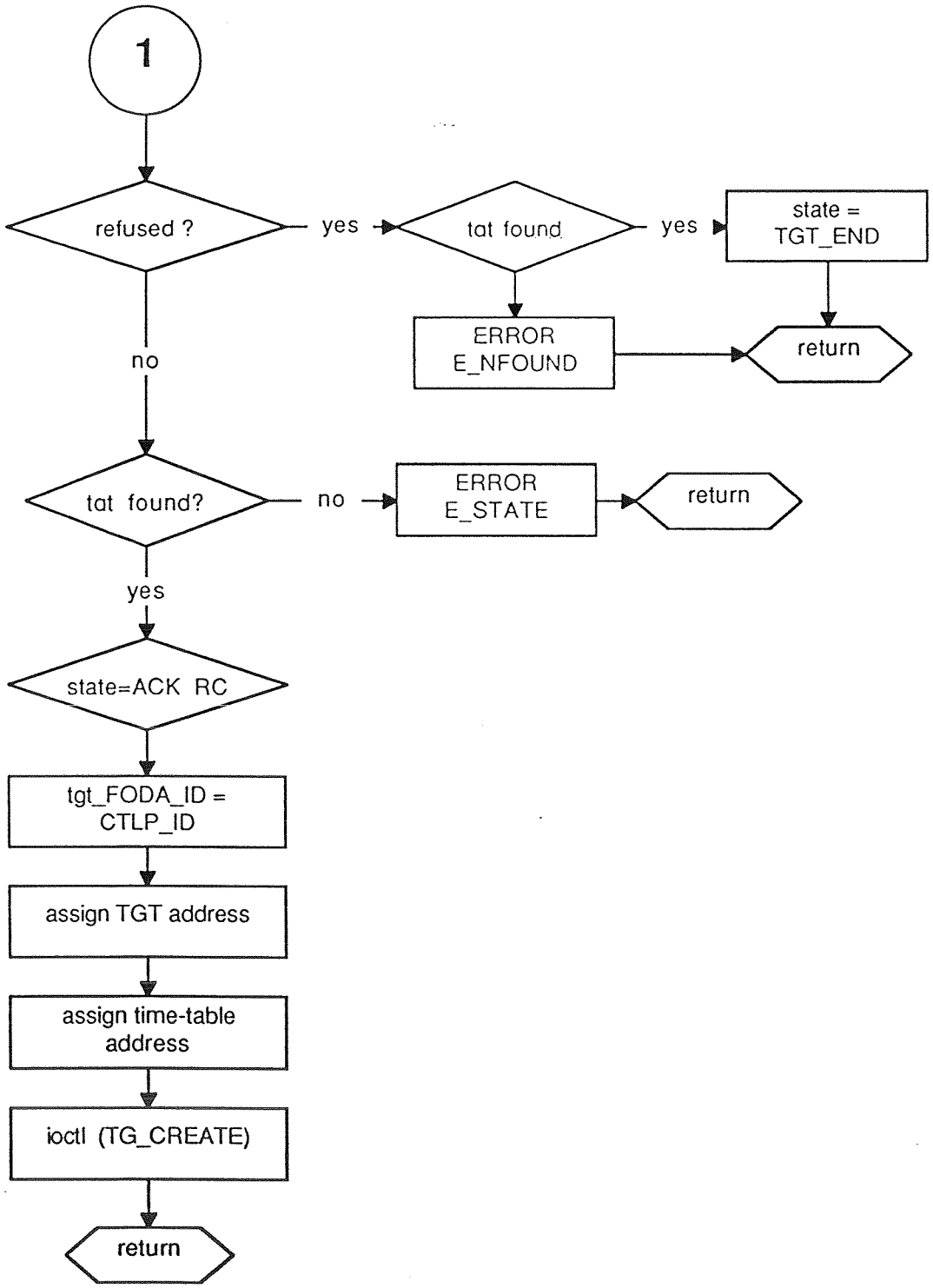
The MTG/R software flow chart

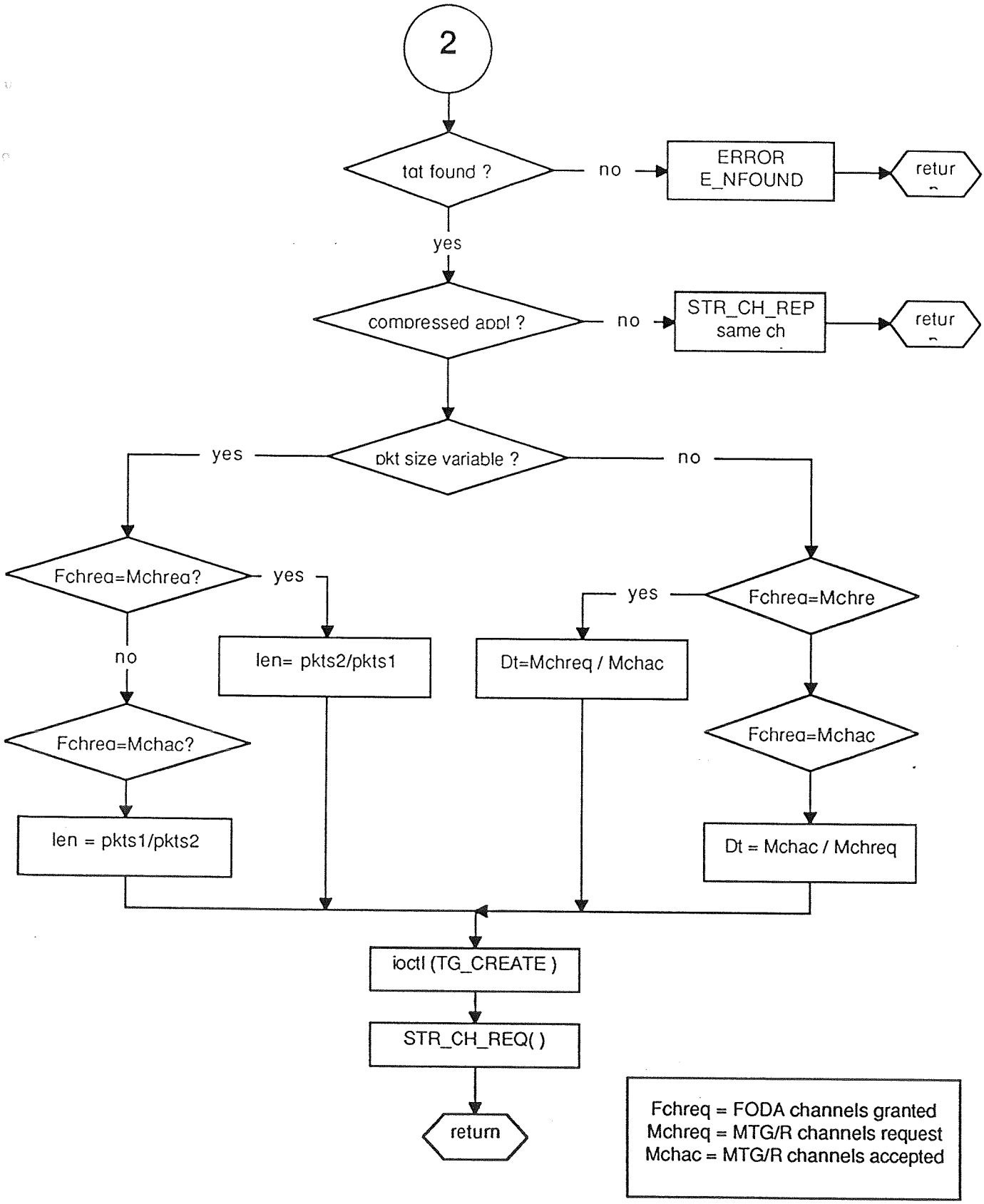
(shadowed blocks are expanded in the following pages)

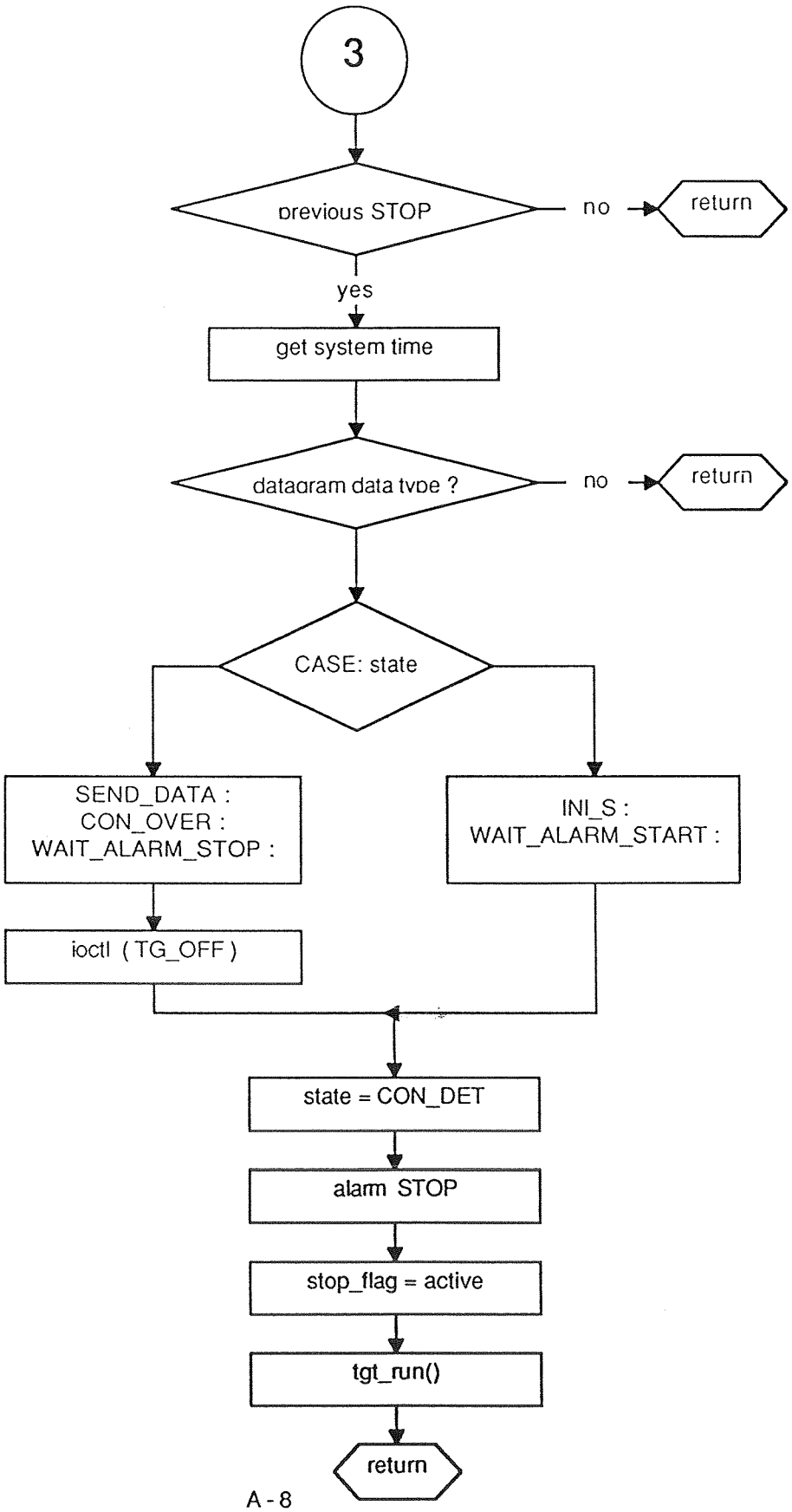

usr
read from keyboard
buf=<CR>
yes
CR=1
yes
digit read
no
set new read mode
return
no
char read?
ch_read()
yes
error
no
check entry ok ?
set run flag
yes
CR=0
CASE:
RUN_MTG()
STOP_MTG()
LIST()
DELETE()
ADD()
QUIT()
return
EXIT

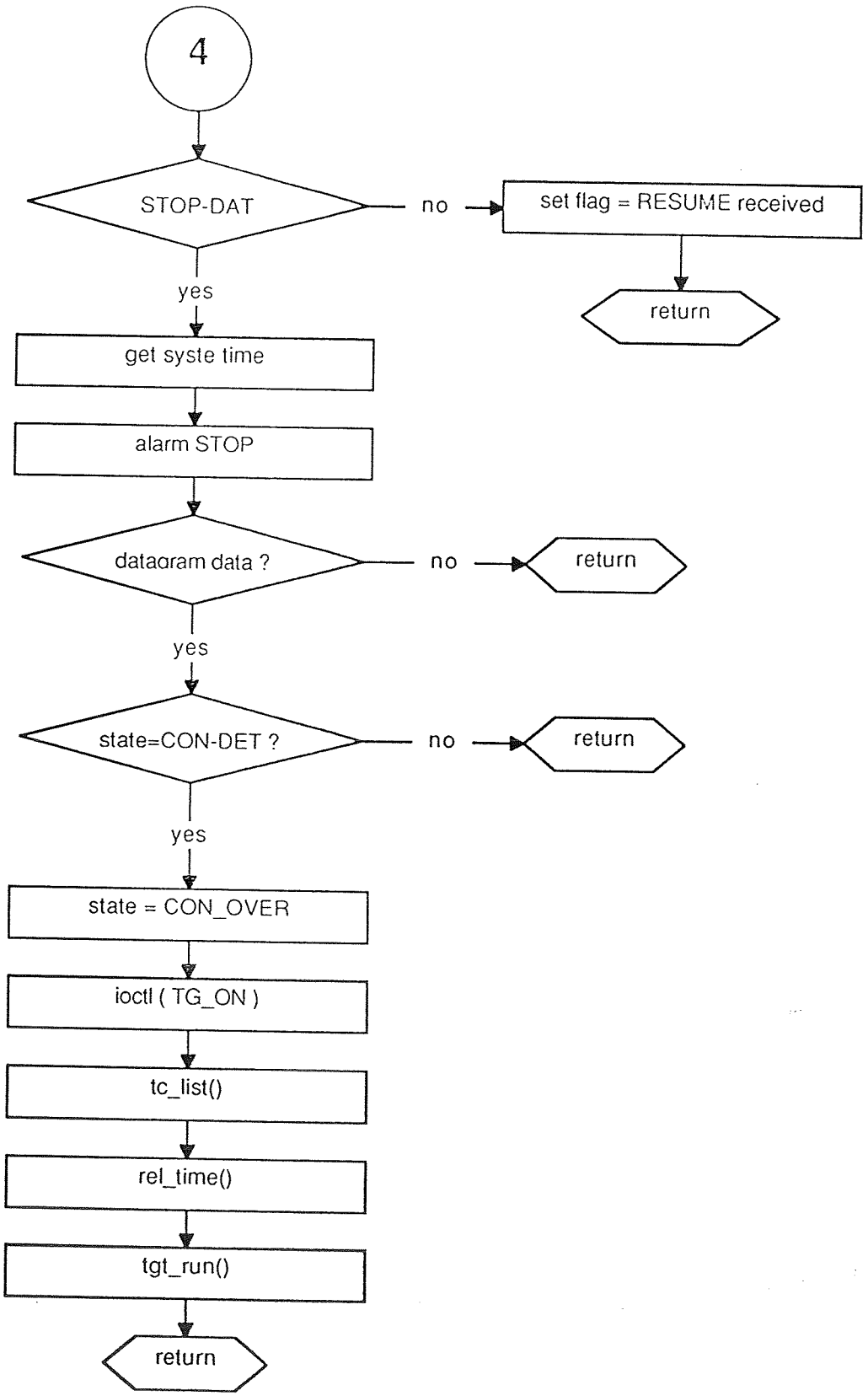

time
CASE:
MTG_RUN ()
MTG_STOP ()
run all or spec?
stop all or spec?
tc_list()
create TGT list (for all or spec)
(TG set stop flag, only active)
tg_active = 3
tgt_run()
get TGT number
get START/STOP time
set START/STOP flag
set apropriate state
alarm (start/stop/wait/time)
alarm expired?
no
yes
wait
time_exp()
time_exp()
CASE: flag
end
wait
start
state=CLOSING
STR_CH_REL
(for stream data)
IO_KILL
tgt_run()
state=TGT_END
tgt_run()
CASE:data
STR_CH_REQ
IO_CREATE
state=WAIT_ACK
state=SND_DATA
tgt_run()
tgt_run()

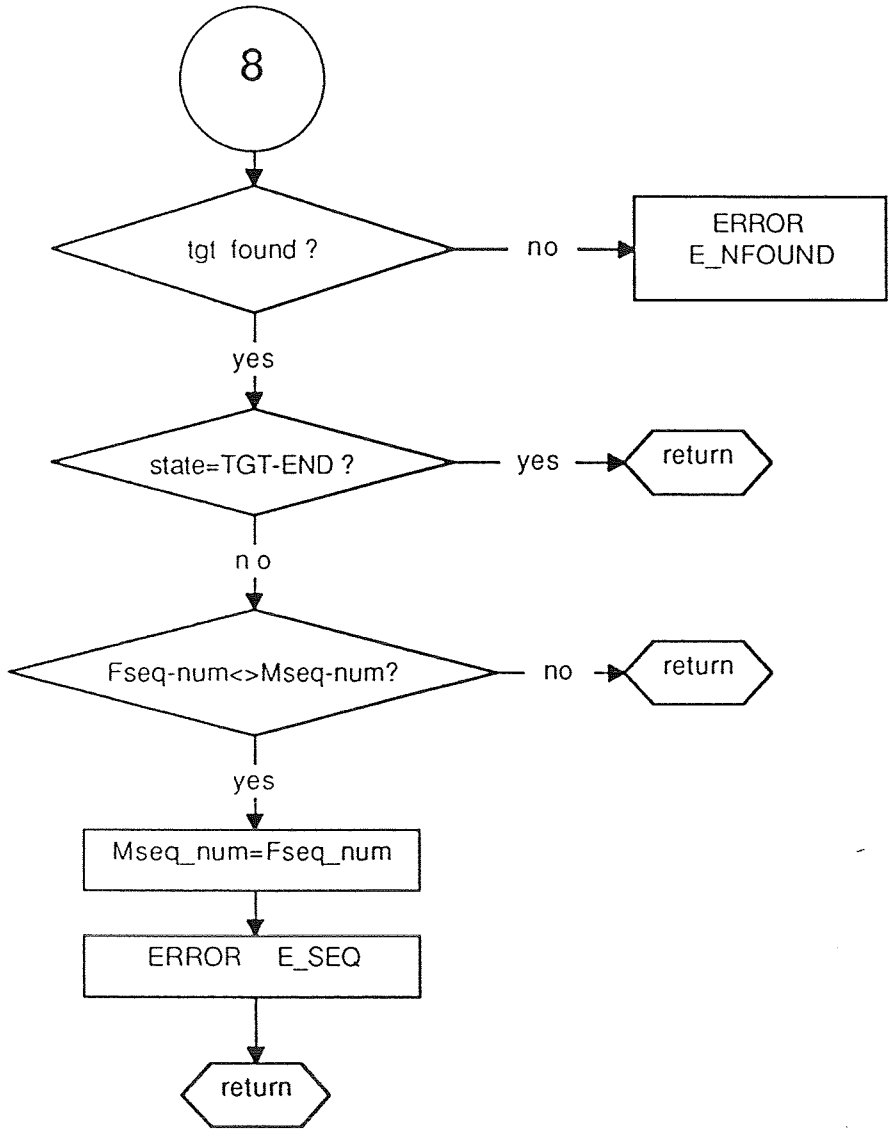

rcv
ctl_ptr <> 0 ?
no
data_ptr <> 0
return
yes
yes
CASE:
CASE:
RCV_STR
RCV_STR_CRC
RCV_BLK
RCV_BLK_CRC
RCV_INT
RCV_INT_CRC
STR_CH_GRANT
STR_CH_MODIF
STOP_DAT_PKT
RESUME_DAT
DST_NOT_RCH
DST_NOW_RCH
ERROR
1
2
3
4
5
6
7
fill archive file
fill acchive file
8
return
return


1
refused ?
yes
tat found
yes
state =
TGT_END
ERROR
E_NFOUND
return
no
tat found?
no
ERROR
E_STATE
return
yes
state=ACK RC
tgt_FODA_ID =
CTLP_ID
assign TGT address
assign time-table
address
ioctl (TG_CREATE)
return


2
tat found ?
no
ERROR
E_NFOUND
retur
yes
compressed appl ?
no
STR_CH_REP
same ch
retur
yes
pkt size variable ?
no
Fchreq=Mchreq?
yes
no
len= pkts2/pkts1
Fchreq=Mchac?
len = pkts1/pkts2
yes
Dt=Mchreq / Mchac
Fchreq=Mchre
Fchreq=Mchac
Dt = Mchac / Mchreq
ioctl (TG_CREATE)
STR_CH_REQ( )
return
Fchreq = FODA channels granted
Mchreq = MTG/R channels request
Mchac = MTG/R channels accepted


3
previous STOP
no
return
yes
get system time
datagram data type ?
no
return
CASE: state
SEND_DATA :
CON_OVER :
WAIT_ALARM_STOP :
INI_S :
WAIT_ALARM_START :
ioctl (TG_OFF)
state = CON_DET
alarm STOP
stop_flag = active
tgt_run()
return


4
STOP-DAT
no
set flag = RESUME received
return
yes
get syste time
alarm STOP
datagram data ?
no
return
yes
state=CON-DET ?
no
return
yes
state = CON_OVER
ioctl (TG_ON)
tc_list()
rel_time()
tgt_run()
return


8
tgt found ?
no
ERROR
E_NFOUND
yes
state=TGT-END ?
yes
return
no
Fseq-num<>Mseq-num?
no
return
yes
Mseq_num=Fseq_num
ERROR E_SEQ
return

REFERENCES

[1] *R. Beltrame, N. Celandroni:*
"The performances of the FODA access scheme: theory and simulation results ", CNUCE Report C86-19.

[2] *N. Celandroni, E. Ferro:*
" FODA-TDMA satellite access scheme: description, implementation and environment simulation", processing of the Tirrenia International Workshop on Digital Communications, meeting on << Satellite Integrated Communication Networks >>, Tirrenia, Pisa (Italy) September 14-16, 1987.

[3] *N. Celandroni, E. Ferro*
"A deterministic study on the FODA frame utilization at variable bit and coding rates", CNUCE Report C89-24, October 1989.

[4] *N. Celandroni, I. Delic, F. Potortì*
"A multimedia traffic generator for high speed network performance evaluation", CNUCE Report C89-33, November 1989.

[5] *N. Celandroni, E. Ferro*
"Protocol between the FODA system and the outside environment", CNUCE Report C90-03.

[6] *G. Iazzeolla*
"Introduzione alla simulazione discreta", Serie di Informatica, Boringhieri.

[7] Motorola Unix SysV68 Documentation.