COMMISSION OF THE EUROPEAN COMMUNITIES

information management

The connection to EIN packet switched network

1977 EUR 5718 EN

COMMISSION OF THE EUROPEAN COMMUNITIES

information management

The connection to EIN packet switched network

S. AMIC, F. SOREL, A. TERMANINI,

Dept. B - Electronics Division

W. BOETTCHER, A. ENDRIZZI, P. MOINIL,
J. PIRE, K. WEAVING

Dept. A. - Informatics Division

Joint Research Centre
Ispra Establishment — Italy

1977

EUR 5718 EN

Published by the
COMMISSION OF THE EUROPEAN COMMUNITIES

Directorate-General
'Scientific and Technical Information and Information Management'

Bâtiment Jean Monnet
LUXEMBOURG

LEGAL NOTICE

Neither the Commission of the European Communities
nor any person acting on behalf of the Commission
is responsible for the use which might be made
of the following information.

A bibliographical slip can be found at the end of this volume

Printed in Belgium

Catalogue number : CD-NU-77-009-EN-C

ABSTRACT

This report presents the design and the implementation of the JRC-Ispra connection to the European Informatics Network (COST 11 Project). Problems raised by the physical and logical links, necessary to connect a large data processing installation to a network are analyzed. Implementation details are well separated from more general topics, so that the document can be considered as a valid contribution to the present discussion on computer networks design.

INDEX

		Page
I.	The COST 11 Project. A European Informatics Network	7
II.	Computer Interconnection Structures: Technical and Political Issues	10
III.	Packet Switching Networks	18
IV.	Issues in Data Transmission Control	19
V.	The EIN System	23
VI.	The Design of the JRC-Ispra Connection	25
VII.	Software Primitives	29
VIII.	The Front End Processor	31
IX.	The HDLC Adapter	34
X.	Transport Station: Ispra Proposal	44
XI.	Subnetwork Test Module: Ispra Proposal	55
XII.	Basic Test Services: Ispra Proposal	67
	Acknowledgements	69
	References	69

I. COST 11 PROJECT - A EUROPEAN INFORMATICS NETWORK

I.1 History

In 1968 the PREST Committee (Politique de la Recherche Scientifique et Technique) of the European Economic Community, chaired by Mr AIGRAIN, proposed a number of projects for cooperative ventures on an international scale. In 1969 the COST group (Co-Operation européenne dans le domaine de la recherche Scientifique et Technique) which includes 19 European nations took up these AIGRAIN proposals and formed study groups to examine them in detail. By mid 1971 the Study Group for Project number 11 had prepared a report and on November 23 in that year an Agreement to establish a European Informatics Network was signed between:

France, Italy, Norway, Portugal, Sweden, Switzerland, the United Kingdom, Yugoslavia and Euratom.

The Netherlands joined in mid 1974 and the Federal Republic of Germany in early 1976.
A keen interest is now being shown by some other European countries.

Two levels of participation in the project are possible:
Signatories may receive information only, or may also nominate and operate a centre connected to the network.

Information only: Norway, Portugal, Sweden, Yugoslavia, the Netherlands and Germany

Approximate Total Cost: 13 million Belgian Francs each Signatory

With Centres: France, Italy, Switzerland, United Kingdom, Euratom
Approximate Total Cost: 100 MBF each Centre.

I.2 Purpose of the Project

The Agreement states:

that the network will facilitate research at the Nodal Centres into data processing problems and will permit the sharing of resources, and that it will:

1) allow the exchange of ideas and the coordination of research programmes,

2) allow the comparison of ideas for national networks and promote the agreement of standards,

3) be a model for future networks, whether for commercial or other purposes, and will reduce the differences between future systems.

The Agreement goes on to say that the hardware and software developed

should be suitable as a basis for any permanent international network which might be built in future.

I. 3 Project Structure

The organization for carrying out the project is as follows:

- Management Committee, set up by the Agreement, comprising representatives of all Signatories, and having ultimate responsibility for the project;
- Executive Body, consisting of a Director who reports to the Management Committee, and three technical assistants;
- Technical Advisory Group, also comprising representatives of all Signatories, to advise the Director;
- Centres Coordination Group, responsible for guiding work that requires close coordination between the Centres;
- ad hoc Working Groups and Special Interest Groups, for example, concerned with drafting the specification and assessing tenders.

I. 4 Contractors

In order to ensure that maintenance and extension of the system can be done with rapidity and assurance, it was decided to entrust the development and installation of the communications sub-network to a contractor. The Signatories proposed about 50 firms in total from which to make the selection. Nineteen firms expressed keen interest, and five Consortia were formed and submitted tenders. Eventually

> SESA (France) and Logica (U.K.) as main contractors with
> SELENIA (Italy) and FIDES (Switzerland) as subcontractors

were awarded a fixed-price contract - signed October 17, 1974.

This is the Initial Contract to design and demonstrate a "Network Switching Centre" (NSC) suitable for the use in EIN (Contract price - 30 MBF). The technical specification was prepared by the TAG, with observers from CEPT; it includes requirements on facilities, performance and acceptance testing. The processor selected is the Mitra 15, from CII. The demonstrations called for under the Initial Contract tasks were completed in October 1975, according schedule.

The Initial Contract gives proprietory rights in the resulting software to the Contractors, but free use is allowed to a Signatory of the Agreement, to the PTT of that Signatory, or to a non-profit making body for its own internal purposes, provided such use is only within the territory of that Signatory.

The Contractors have given assurance that they would make the software available on a fair commercial basis for purposes not covered by the above provisions. Use for other purposes would be relatively easy because of the adaptability of the design and the exceptionally high standard of documentation.

Supply contracts, separately negotiated between the five Centres and the Contractor, covered the installation of NSCs to form the initial network. As planned, this was handed over to Centres on May 26, 1976 (approximate cost for 5 Centres = 35 MBF).

I. 5 The Research Programme

The TAG is responsible for the preparation of the EIN research programme, the second phase of the project, whereas its execution is organized by the Centres Coordination Group. The initial emphasis was on the preparation of sites and adaptation of existing computer systems ready for the installation of the sub-network, and on the development of an end-to-end protocol, and a Virtual Terminal Protocol. But attention is now being given to the wider aspect of computer communications, such as the design of higher level protocols and network command languages and the consideration of advanced techniques for cooperation between the computer systems, joined by the sub-network.

I. 6 Other Applications of EIN

Although the EIN sub-network is intended specifically to meet the requirements of research centres, these requirements are in fact so diverse that a flexible design has been essential. The involvement of CEPT representatives from the earliest stages, and the continuing dialogue about technical aspects, has ensured that EIN reflects contemporary thinking on the best structure for this type of network.

It seems likely, therefore, that the EIN sub-network design could be used for other purposes, and that the techniques being developed by the EIN Centres for using the network will be valuable in a variety of applications.

A particularly important development has been the adoption of the EIN design as a basis for Euronet. This is a project of the European Community designed to link terminals and data bases throughout Europe. PTTs are providing a communications network for the purpose of using EIN switches in Frankfurt, London, Paris and Rome, with additional equipment to implement CCITT X25 interfaces, and to handle character terminals. This network may well become a model for a European public data network.

II. COMPUTER INTERCONNECTION STRUCTURES: TECHNICAL AND POLITICAL ISSUES

This section presents a classification for systems of interconnected computers and isolates the basic criteria which influence the design decisions. There do seem to be trends emerging in designs for several application areas, particularly in interconnection of relatively large computers over geographically long distances.

It is most important that users express their interest and requirements, for networks have raised the need for compatibility and standardization. Computer manufacturers and PTTs start defining what users should agree upon in order to communicate.

A brief summary of the major achievements of the standardization process is also presented, with the analysis of the implied division of responsibility between users, PTTs and manufacturers.

The user's attitude in respect of data processing systems may be effectively described by the following example: Let's take the telephone service; anyone can communicate with anyone else provided he knows the proper number and the partners agree on a common language.
Why should users not communicate with DP services in the same way?
The usual answer is geographical distance, modem, code, terminal to computer and language incompatibilities.
The only way to make DP services accessible to a large user's community is to build an "informatics network" to which terminals, computers and services can easily be connected.

During the last ten years many private networks have been installed.
They all appear to be specialized systems, encapsulated within a rigid structure that puts heavy constraints on possible enlargements of services and modifications of configuration. The supporters of such dedicated networks stress the point of availability and serviceability, but it is well known that only few components of such complex systems are best utilized and that reliability is achieved by heavy investments in redundant equipment.

Indeed, such networks were conceived in the scenario, dominated on the one hand by the third generation computers, whose major characteristic is the centralization of the logical capabilities, and, on the other hand by low speed, unreliable data communication services. This picture is slowly changing, in the sense that the interest for mini-, nano-dedicated computers is growing and PTTs are considering more seriously the market of data transmission.

Will computer manufacturers and PTTs evolve their products in order to match users' needs as stated above?

The main problem is compatibility. It makes sense to remember that incompatibility can be tolerated during a short period in which technology is rapidly changing, but at the present stage of the DP market, incompatibility is nothing but the old commercial trick to avoid the client moving to other suppliers.

The user's interest is to make the best heterogeneous combination of available pieces and to make the systems accessible to the maximum number of clients. The technical solution to incompatibility is either homogeneity (which leads to monopoly) or standardization of the interfaces. The benefits of the policy of standardization will be shared by the users and by the small-medium size manufacturers.

Distributed data processing systems constitute the present challenge. Big technical and political efforts in the field of informatics are requested in order to reduce the influence of the large brontosaurus centralized systems and to wake up PTTs from their proverbial passivity.
The reward for the end user will be a modular, extensible, overall accessible set of "informatics services" offered by public or private organizations on the basis of a fair market. This is believed to be the best evolution of informatics. The problem is what technical solutions should be adopted and what division of responsibility the participants have to agree upon.

Large manufacturer companies, like IBM, plan to invade the field offering global systems in which it is very difficult to identify clean "interfaces" in order to "plug in" other systems. They relegate PTTs to the role of line suppliers and they intend to perform switching functions inside their own components. On the other hand, some PTTs start providing a data service by building public networks. Examples are Transpac-France, EPSS-UK, Euronet-Europe, Telenet-USA, Datapac-Canada. While defining the standard procedures to be followed by the users for transferring data, they deeply interfere with the world of terminals and computer manufacturers. Indeed any logical component that happens to be implemented in such a way that its use can be shared between several users, may easily become part of the "service" and therefore become PTTs' concern.

In the meantime organizations of scientific research bodies are collecting operational experience on computer networks (Arpanet, NPL network, Cyclade, EIN, ESA) or designing new ones (IIASA).
The actions for standardization at international level are discussed within ISO, IFIP and CCITT. At present, many problems are still open.

Users' experience of computing is now increasing rapidly, so that users may well specify more accurately what they require in the future without being directed by manufacturers' or PTTs' own ideas. The following discussion is intended to express our opinion on what can be reasonably expected from the growth of the interest in computer networks.

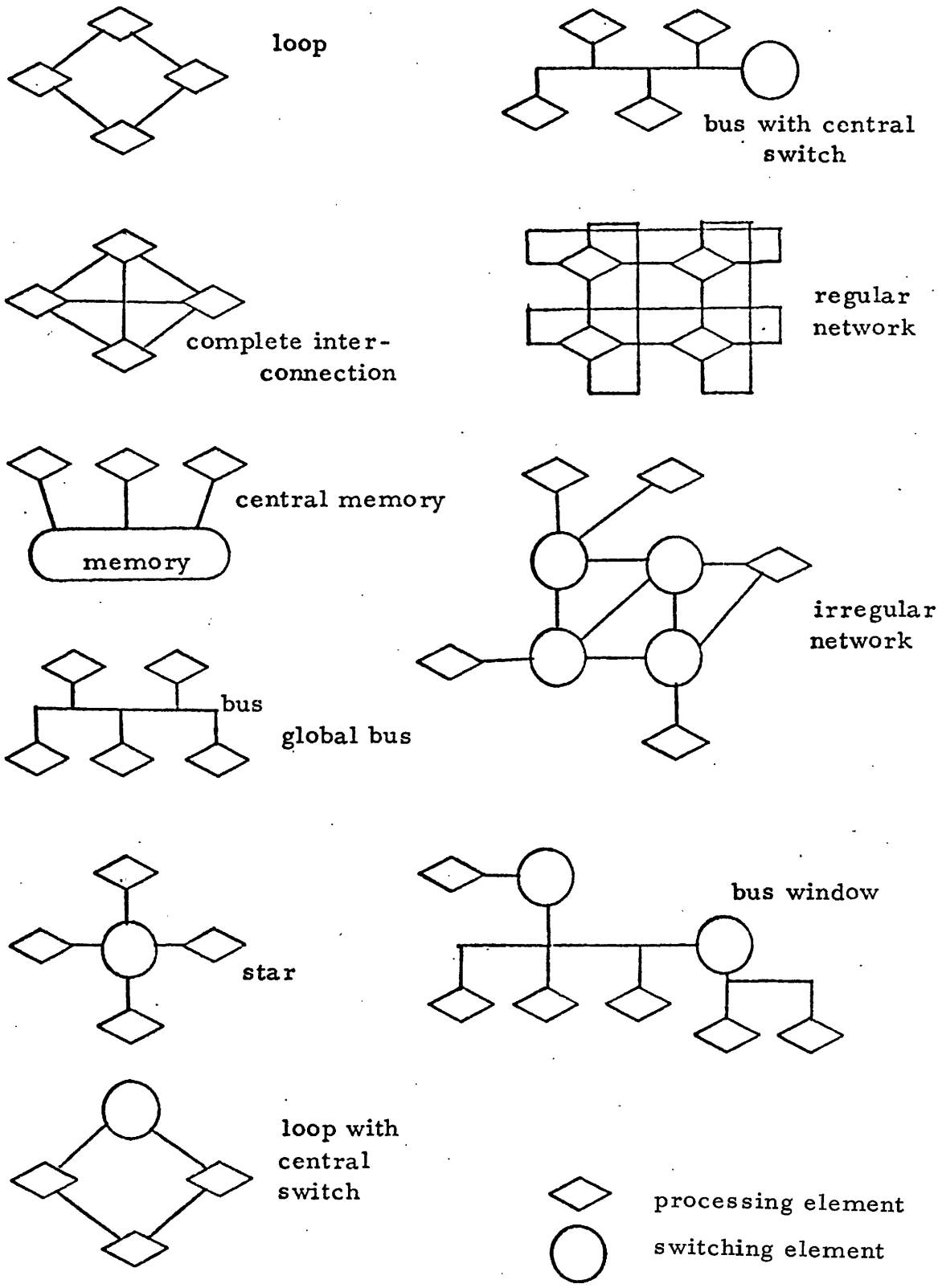


Fig. 1 - Computer Interconnection Structures

Fig. 1 shows a classification of computer interconnection structures[1]. Systems may range in organization from two processors sharing a memory to large numbers of relatively independent computers connected over geographically long distances. The rationale for various architectures can be described and a comparison can be attempted by isolating some fundamental design criteria.

The common underlying idea is to make processors in which processes can execute exchange data blocks, semaphores, service requests (messages) by means of lines, radio links, busses or memories (paths). The first strategic choice concerns the insertion between the sender and the receiver of an intervening intelligence (switch) that affects the message by altering it or by routing it to alternative paths. A further decision is needed between centralization or decentralization of the switching functions. Paths may be dedicated or shared, in which case contention becomes a major consideration.

A rough feeling of the characteristics of the different architectures emerges if some fundamental properties are taken into account like modularity, logical complexity, failure effect aspects, geographical extensibility.

Modularity is the ability to make incremental changes in system capability. The addition of a new resource to the system may logically be prohibited or restricted to certain elements and places. It might involve the addition of other components or bring a shared path to bottlenecking.

The logical complexity refers to the methods by which the switching information is communicated in order to allow the switch to make the proper choice between available paths. The more dynamic the processing environment and the more physically dispersed the system, the more complicated the problem. A decentralized switch takes decisions on the basis of a necessarily obsolete and partial knowledge of the overall system status.

Another important design characteristic is the cost of fault tolerance and the way a system is reconfigured to mask faults in processors and intercommunication paths. The failure reconfiguration measure may range from excellent in systems requiring no overt reconfiguration and having minimal spare hardware, to very poor for those requiring that the entire intercommunication system be made redundant.

Trying to investigate future directions in distributed architectures, there do seem to be trends emerging in designs for several application areas. Loop organizations are used to connect multiple minicomputer systems in university, research laboratory or industrial automation environments where their poor failure characteristics and long message transit times are no problem.

Global bus with serial bussing is becoming the dominant architecture in

real-time control environments. The global bus allows reduced message transit time and fast reconfiguration after failure by simple redundancy.

Irregular networks are suited to applications requiring interconnection of relatively large computers over geographically long distances. Significant processing power is dedicated to the switching function in order to optimize data links utilization. The connected systems are faced with the fundamental problems of interprocess communication protocols and load sharing.

It is the common opinion that the remaining system types (Fig. 1) have each one or more significant weaknesses not sufficiently compensated for by their other advantages. However, with the revolutionary freedom to plant machine intelligence anywhere, system design and usage should see a fundamental change and a reassessment of the entire computing scene is likely to be necessary in the future.

The onrush of large-scale integration has already voided the technological distinction between logic and memory within processors and other concepts held to be distinct, irreconcilable opposites are becoming items for trade-off. Examples are hardware-software-firmware, compilation-interpretation-execution, man-machine, control-cooperation.

Technical discussions on fourth generation systems[2] have pointed out that current systems are so seriously inadequate that a radical change must be undertaken. The majority of users and manufacturers, on the other hand, see only advantages in restraining the rate of change because of the extremely high investment that has been necessary in the design, training and software development.

Computer network is the proper answer to this contradictory situation. Its supporters are:

- old systems requiring multiaccess-facilities to cover investment and running costs,
- user-manufacturers organizations trying to define standards in order to build modern modular systems,
- public or private organizations integrating, centralizing or decentralizing their services.

The existing batch, time-sharing and centralized data base systems ask for the possibility to exchange files and to connect remote peripherals like interactive terminals and RJE stations. It is difficult to sustain that a set of heterogeneous large systems can be involved with networks more than that. Work carried out on ARPA, CYCLADE networks has demonstrated that only interfaces to drive simple terminals and to transfer low level structured files can be inserted and maintained within present day operating systems. Academic research is going on in the field of "net-

work operating systems" which consider memories, processors, programs, stored data structures and system utilities also to be independent network allocatable resources. Stressing the network concept to that level of generality has no practical meaning unless manufacturers themselves collaborate by designing their future operating systems in modular form so that interfaces for remote management of resources can easily be inserted.

Recent trends in computing show the appearance of cheap, dedicated systems specially designed for optimal performance of user's functions like text editing, graphic manipulation, mathematical and engineering problems, consultation of data bases and data collection.

The challenge consists in the fact that complex logical functions are performed without the overhead imposed by large systems. Concepts like stored programs, processor, devices, compilation, execution are no more clearly distinguishable inside their structure. Still, what is needed is the ability to connect them together in order to make the best use of their complementary capabilities. The development of networks will stress manufacturers to standardize the modules handling the communication of such systems with the external world.

We assert that very powerful, extensible systems, harmonizing with new technology and new economic realities can be constructed without the long development time which has been the bone of very large systems.
The structure should show an extreme form of decentralization of the intelligence so that local events and information are not to be submitted for judgement to remote processing. Intelligent autonomous units exchange only "useful" information. Control overhead and intercommunication complexities are therefore strongly reduced.

The communicating partners appear as interpretative machines which exchange messages containing logical names and descriptors. Logical functions and high level semantics are subjects easy to agree upon and therefore they are the best candidates for standardization.

The outcoming picture shows an architecture in which independent asynchronous modules all refer to an Interprocess Communication Mechanism to communicate and to coordinate their activities. Modules performing similar activities will react to a standard protocol.

One fundamental layer of the ICM seems to have been identified: the "transport service". This name refers to the facility that delivers messages in transparent mode to addressable entities. It is assumed that symbolic names can identify processes and that finite streams of bits constitute the atomic units of any conversation. The transport service may provide the correspondents with additional facilities like message sequencing, error

control and sender-receiver synchronization. The proper interface to the transport service has to be installed wherever modules willing to communicate, operate. It contains all the features that allow communication to take place locally and remotely. It can be implemented in various ways: the important point is that such a component can be isolated inside any system and become part of a general addressing facility.

At this stage of the discussion we can move to the analysis of the major achievements of the standardization process. It should be noted that standardization efforts are following two converging directions: from high level programming languages through data structures definition to I/O operation semantics (examples are DDL and DML for data bases) and from line, modem through terminal interfaces into computers. In between there is the jungle of hardware configurations and operating systems. The following is a brief survey of the standardization of data transmission procedures.

A first achievement has been the definition by a joint effort of ISO and CCITT of the Alphanet nr. 5, the 7 bit code, the X 21 (or X 21 bis) universal circuit interface together with the HDLC universal transmission procedure. This allows a large degree of compatibility between terminal equipments, regardless of configuration or communication media.

CCITT is now defining under the label X 25 the rules for accessing networks which are able to support flows of data packets between correspondents. IFIP's working groups discuss the problems of internetwork packet formats and internetwork end-to-end protocol (the transport service).

ARPA, EPSS, CYCLADE, EIN, EURONET propose their own suggestions for the standardization in handling simple terminals, file transfer, remote job entry.

The underlying idea is to provide national and international data transmission services with the proper conventions that will allow classes of remote correspondents to be put in contact.

IRIA and NPL are organizing a series of workshops in Europe with the intention to set up a "User's Association" that should express user's views on standardization in computer networks, particularly in the public domain. Such an association might possibly make input to CCITT and ISO. This initiative should be followed with great interest. The discussion covers network architectures, end-to-end protocol, virtual terminal protocol, file transfer protocol. Those subjects are still open since neither manufacturers nor PTTs have played their final game.

- PUBLIC DATA NETWORKS:

Canada	Bell Canada (Datapac) CNCP (Infoswitch)
France	PTT (Transpac)
Germany	PTT (EDS)
Japan	NTT (DDX-1)
Spain	CTNE network
Great Britain	PTT (EPSS)
USA	TELENET ITT Compak ATT
Europe	CEPT (Euronet)
France, Italy Spain	RECORD III

- PRIVATE NETWORKS:

  General Electric service bureau

  Infonet

  Cybernet

  Tymnet

  Site

  Swift

- RESEARCH NETWORKS:

  NPL

  ARPANET

  CYCLADE

  ESA

  EIN

  IIASA

  RCPNET

- SATELLITES NETWORK:

  ESA - EIN - CERN - CEPT

  IBM Comstad

Fig. 2 - List of the Major Existing or Planned Network Projects

III. PACKET SWITCHING NETWORKS

The architecture described as irregular network is the one we are interested in. One clear division of responsibility is implied in the design: connected computers are not allowed to perform any switching function. This has the important consequence that a switching service (subnetwork) is clearly identified which is responsible for supporting communications in a dynamic way.

Telephone and telex networks have been based on the concept of circuit switching in which a path is established through the network for the whole duration of a call. To establish the path and disconnect it at the end of the call, a separate part of the network is used (control and signalling system). Circuit switching presents the following characteristics:

- there is no delay in transmission other than that caused by finite speed of transmission (at some fraction of the speed of light),
- setting up and releasing a call takes processing and time overhead,
- path capacity is provided whether or not it is being used during the call,
- the network is not concerned with any aspect of the customer's data such as message structure, error control and acknowledgement.

An alternative approach has been used in telegraphy. The telegraph message is handled as one unit and no call is set up to carry it. In effect, the process of setting up the path is carried out by the header at the beginning of the message. The whole message is accepted by a switch and stored before it is retransmitted to the next centre. Message switching presents the following characteristics:

- there are transmission delays which depend on the load of the network,
- there is no overhead or delay for call set-up,
- utilization of transmission channels can be high and no transmission facilities need be occupied during the silent periods of a conversation,
- the network assumes complete responsibility for delivery.

The message is an essential feature of the communication between processes running on computers. Here the data starts from store in the form of a message. In a multipurpose system messages of various lengths can be exchanged at various frequencies between computers and between computers and terminals.

It should be noted that the message concept is only relevant to the communicating partners. In case subnetwork resources are allocated on a per message basis, the transmission of a long message considerably delays

the remaining traffic on shared lines and buffer allocation strategy for supporting the store and forward mechanism becomes a major concern. The solution is to make the logical unit (message) independent of the quantum of transmission (packet).

By breaking the message into packets, the nodes are able to forward the first packet of the message through the network ahead of the later ones. For a message of P packets and a path of N nodes, the delay is proportional to P + N - 1 instead of P * N. This pipeline processing to be effective requires balanced distribution of resources along the path in order to shorten queueing latency at each node.

The idea to split long messages into packets also allows a more dynamic mixing of conversations. For example, long file transfers can be interleaved with more urgent interrupts and conversational transactions. The strategy for multiplexing conversations and the message assembly-disassembly function, should be kept as close as possible to the end users since they are in the best position to judge the relative priorities. The role of the subnetwork is to transmit independent packets with low delay, high throughput and high probability of delivery. There is a fundamental trade-off between the three design goals as it is readily apparent in considering the mechanisms used to accomplish each goal in various implementation of subnetworks. The control of the traffic on data networks is a particular instance of a distributed resource sharing problem.

IV. ISSUES IN DATA TRANSMISSION CONTROL

A sub-system such as a packet network, or a line controller, is shared by a number of traffic streams. There exists no magic recipe to be suggested in order to satisfy all the various service characteristics the end-users may require. The following is intended as an introduction to the analysis of the new problems distributed systems bring about. These problems are due to propagation time and unreliability associated with any form of long distance communication.

Here is a list of topics which are relevant to any data communication system:

Error Control

If an unreliable path is inserted between the sender and the receiver, the sender should be prepared to retransmit the data and the receiver has to acknowledge the receipt. The loss of an ACK may generate duplicates at the receiver side and delay the freeing of resources at the sender side. In a layered structure a path is always embedded inside a logical entity that takes care of retransmission, ACKs, NACKs and recognition of duplicates (example Line Control Protocol).

End-to-End Flow Control

It is the mechanism by which the receiver maintains the sender traffic within limits compatible with its own speed and resources. The connecting path is not only concerned with the transmission of data but also with the transmission of synchronization commands. The path itself may have storage capabilities which may introduce smoothing effects on the flow or even undesirable oscillations. The sender anticipates messages in order to reduce delay and the receiver declares from time to time its availability to accept future messages.

Congestion Control

The intermediate subsystem maintains input traffic within limits compatible with its own resources. The throttling of some sources may be necessary in order to allow delivery of the offered traffic. It is well known that in a dynamic environment congestion is a threshold phenomenon. A certain percentage of the available resources should be kept free in order to offer a stable service.

Sequencing

The ordered flow of data may have a semantic meaning to the communicating partners. Scrambling effects may take place within the subsystem due to error control and routing strategies which act on the packet level. A well accepted design criterion is to perform resequencing at the receiver end. Each packet follows its own route within the subnetwork, so that the possible delay in delivering one single packet has low probability to influence the delivery of the complete message or flow. This follows from the overlapping capabilities of the subnetwork.

In the absence of a universal model, a first step is to investigate and classify the basic components of any implementation of data transmission control. It should be noted that even simple mechanisms can solve some or all of the problems mentioned above in the sense that for example logical tools to allow sequencing also solve the error control and support the flow control problem.

The following is a list of the basic logical tools one can identify in the various implementations.

Stop and Go

The source is controlled by signals of a binary nature. Either it can send traffic without limit, or it is barred from transmitting. This technique is used in most line-control procedures, such as BSC or HDLC.
A stop or go signal takes effect immediately or at the earliest opportunity after the transmission currently in progress. The receiver has to take into account the transit delay necessary to carry stop/go signals. Depending on

this delay a certain amount of traffic may continue to arrive for a while after the stop is sent.

Credit

The sender cannot transmit unless he has received an indication about the amount of traffic that can be accepted by the receiver. This scheme protects the receiver from an excess of arrivals. The amount of acceptable traffic may be indicated in terms of bits, characters, messages, sequence of messages.

Error control schemes are often used as a rudimental form of credit. Indeed, acknowledgements usually carry two meanings: good reception and a credit to send one or more additional messages.

Rate

The sender knows the rate at which the receiver can operate and adjusts the transmission rate consequently. A typical case is the feeding of devices such as terminals that are usually unable to control senders.

Delay

When transit delays increase, some sources give up, like users of an on-line application. In some other cases tolerance is determined by the setting of a timer or by a threshold in the number of unsuccessful attempts. Infinite delay corresponds to discarded traffic. The fact that the sender may keep retransmitting may cause the subsystem to enter an unstable domain since more resources are required to carry the same amount of traffic while degrading the service. If the destination gets blocked input traffic should be prevented. Indeed, more often sender and receiver are coupled via an end-to-end protocol requiring some form of receiver acknowledgement which limits the amount of outstanding traffic.

The subsystem should guarantee the delivery of the message within a maximum delay. This parameter can be used by the subsystem for deciding when the message can be dropped, and by the sender to activate retransmission.

Name Space

The sender has to provide unique names to designate messages in order to recognize ACKs from the receiver. ACKs may get lost so that the receiver should possibly transmit from time to time the complete set of the received names and recognize the incoming duplicates.

It results that sophisticated error control requires a rather symmetrical collaboration between sender and receiver. In order to represent the complete set of names a finite bit pattern is used, each bit representing a re-

usable resource to carry one message. The communicating partners have to agree on the initial setting of those bits.

If sequencing is required the name space is usually represented by a cyclic numbering scheme. The ACK of number N implies proper receipt of the complete sequence up to N. A simple credit mechanism can be implemented by including into the ACK a window size which represents the number of messages the receiver is ready to accept next.

Synchronization

Each of the two communicating partners keeps a status information of the other. The drifting of the partners may occur asynchronously and status messages may get lost or out of sequence. This may cause overflows or deadlocks. For example, if stop-message is lost, oncoming traffic may continue and if go-signal is lost, the sender remains silent. Fuzzy states can be recovered from by periodic status exchange, even in the absence of normal traffic, and by allowing short messages to be exchanged (interrupt-like), at any time to force status exchange. There are only two notions the partners need not exchange information about: time and silence.
The ultimate tool to inform the partner is to stop transmission for a certain period on the assumption that the subsystem will empty the path and the partner converges after a time-out to a wellknown state.

This strategy is used to safely bring down both partners to a neutral state from which reinitiation is possible.

Subsystem Resource Management

The introduction of a shared subsystem brings up additional problems. A sender/receiver pair may attempt to keep smooth steady data flow and still achieve bad results because of frequent packet loss and unpredictable transit delays. It would seem that end-to-end strategies require a global analysis taking into account all resources involved. This is not very practical so that intermediate components may be thought of as simplified models defined by a set of quantitative characteristics like transit delay, band width, error rate.

End-to-end protocols should be designed to be efficient in normal conditions and be safe in case of occasional deficiencies of the subsystem.

The shared subsystem presents less simple problems since it has to cope with dynamic unpredictable requests. Simulation studies carried out at NPL have shown that simple throttling mechanisms of the sources could be sufficient to avoid congestion. The technique has been termed isorithmic control. Intrinsically it is a distributed credit scheme with a constant allocation maintained over the whole network.

Routing

A centralized system may be informed of the status of all the paths in a rather simple way. A switch in a distributed system may take decisions which are not coherent with the current status of the system. This is due to the fact that the switch has an incomplete and out of date view of the available paths and traffic loads. Data packets travel usually faster than routing information in order to reduce overhead under normal conditions.

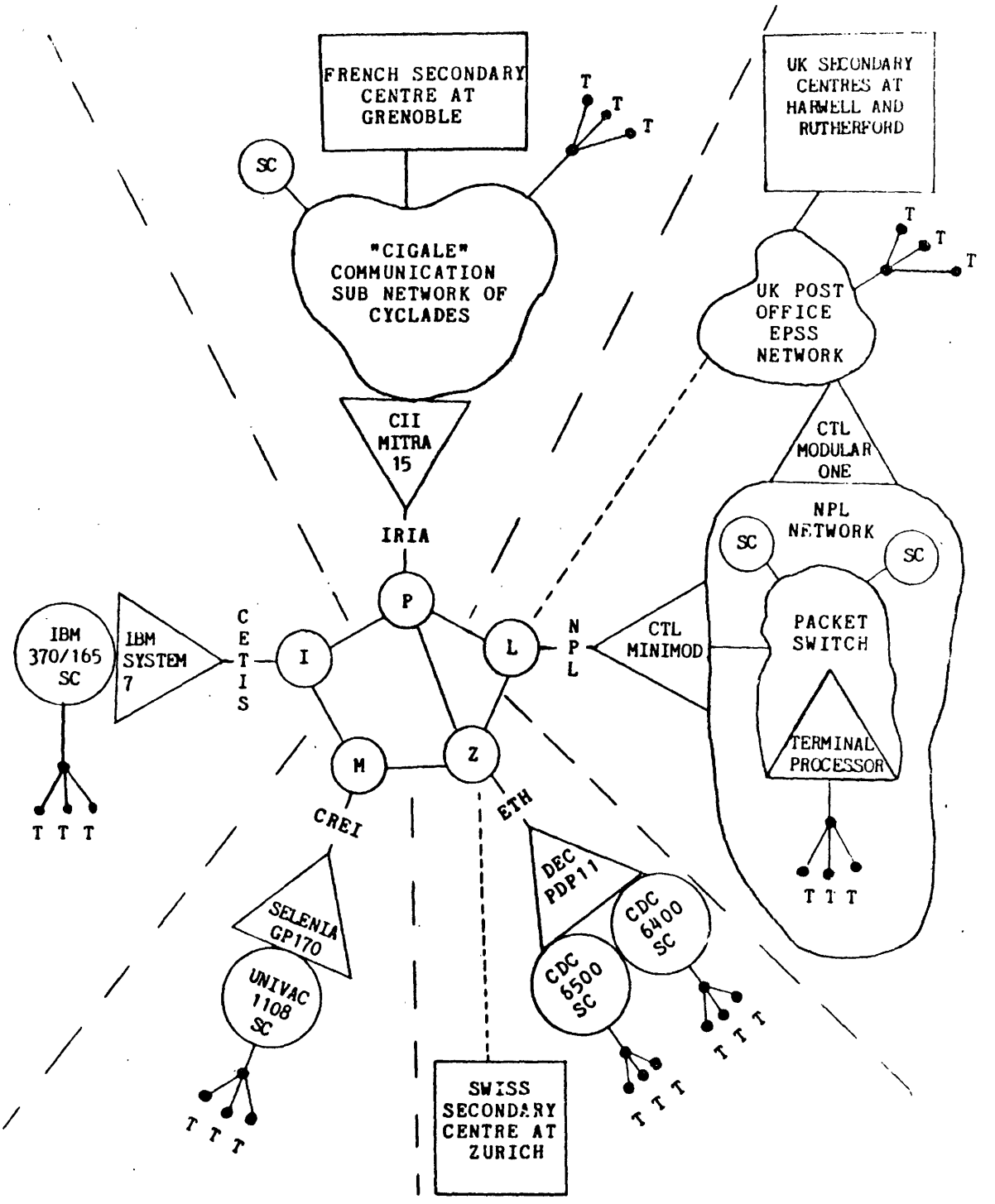
Routing decisions are taken following a distributed algorithm which takes care of the propagation of the information concerning path availability. From time to time each switch is informed about system configuration changes and traffic by the adjacent nodes. The routing algorithm tries to minimize transit time to destination on the basis of the available estimates.

V. THE EIN SYSTEM

Fig. 3 shows a simplified diagram of EIN including Nodal Centres' systems. The centre of the picture shows the EIN subnetwork made of Network Switching Centres connected by lines. All around the subnetwork Nodal Centres' systems are shown separated by dotted lines. As can be seen, some of them are conventional multi-access computing systems with mainframes serving terminals while others are complete private networks of distributed computers and terminals.

The subnetwork design is based on techniques proven by earlier networks with the addition of some design peculiarities. Key points are:

- a Datagram type of service is provided, together with optional end-to-end sequencing and traffic control facilities,
- the HDLC frame is used for transmission,
- adaptive routing is used within the subnetwork,
- connection of a subscriber to more than one node is permitted to enhance reliability,
- well proven hardware is used, as over 1,000 Mitra 15s are now in operation,
- all relevant international standards have been complied with,
- modular software using a high-level language developed for the purpose gives flexibility in implementation and facilitates future use of alternative hardware,
- extensive facilities are available at the nodes for statistics collection, recording of alarms, modification of network parameters, and the modification and remote loading of node software through the subnetwork,
- nodes are capable of entirely unattended operation.



CETIS - EUROPEAN SCIENTIFIC INFORMATION PROCESSING CENTRE - ISPRA
CREI - CENTRO RETE EUROPEA DI INFORMATICA - MILAN
ETH - EIDGENOESSISCHE TECHNISCHE HOCHSCHULE - ZURICH
IRIA - INSTITUT DE RECHERCHE D'INFORMATIQUE ET D'AUTOMATIQUE - PARIS
NPL - NATIONAL PHYSICAL LABORATORY - LONDON

SC - SERVICE COMPUTER

Fig. 3 - Simplified Diagram of EIN Centres' Systems

EIN is a heterogeneous type of network made up of a set of subscriber computers (SCs) whose hardware architecture and operating systems (CDC, CII, IBM, CTL, UNIVAC) varies significantly from one site to another. In such a type of network, a priori, no possibility exists for interaction between processes which are not being executed in similar machines. To render this possible, new conventions and new facilities must be defined. This is one of the goals of the EIN research program and its achievement enables SC to SC communication through the means of homogeneous end-to-end protocols. The foundation of the structure is the Transport Station (TS) that is implemented in each virtual subscriber computer willing to make its processes communicate with remote partners. The transport service provides the actual interprocess data transmission mechanism in the form of lettergrams (independent messages) or liaisons (contexts supporting conversations). A well-defined interface to the TS allows parallel development of higher levels of service like terminal support, file transfer, remote job entry, distributed data bases.

VI. THE DESIGN OF THE JRC-ISPRA CONNECTION

Fig. 4 shows the present connection of the JRC-Ispra data processing installation to EIN.

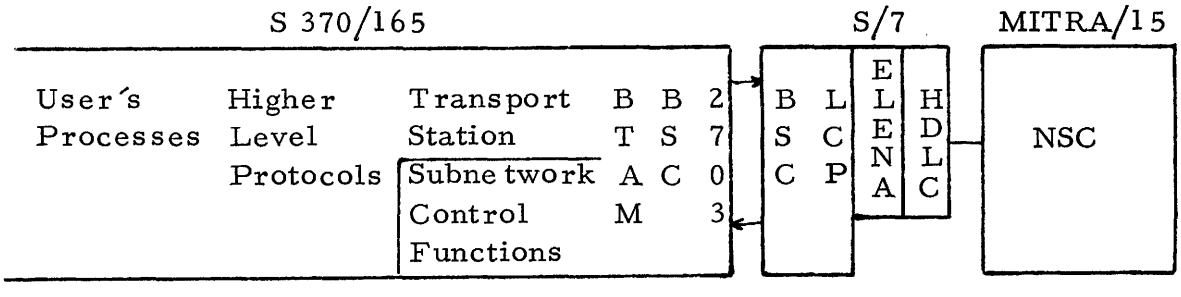


Fig. 4

The Subscriber Computer is an IBM 370/165 working under OS/MVT. As the IBM 370 does not support HDLC, a System/7 is used to convert the HDLC frames to BSC messages and vice versa. This is only a temporary measure, as it is hoped that IBM will eventually support HDLC, at which time System/7 will be released from its task.

VI.1 Line Adapter

ELENA is a home-made piece of hardware that is capable of supporting the full duplex HDLC line transmission. The data exchange between S/7 and ELENA is byte by byte in hand-shake mode, whereas the exchange with the network is a serial stream of bits. The hardware logic performs mainly the following functions:

- generation and detection of flags, annullation characters and cyclic redundancy check,
- serialization and deserialization of data,
- insertion and erasure of '0' in the bit stream to achieve transparency,
- indication of status bits (begin and end of frame, overrun error, etc.).

The equipment operates in full duplex synchronous data transmission at net speed up to 600 Kbit/s. The connection with S/7 digital I/O modules involves input with interrupt (6 bits), input without interrupt (12) and output (16).

A 9600 bit/s version of ELENA has been implemented using a microprocessor. A microprogrammed HDLC-BSC converter with 1 Kb buffer is now under study.

VI.2 Front End

Inside the S/7 two processes, one for each direction of transmission, are responsible for the conversation with ELENA and for the assembly-disassembly of packets into bytes.

The Line Control Procedure is also implemented within the S/7 since real time interventions are required in order to monitor the status of the line and to acknowledge and retransmit packets.

BSC modules are kept as simple as possible by dedicating distinct physical lines to each direction of transmission. S/7 memory provides some buffering capability for packets in transit. This allows S 370 to be loosely coupled to network load oscillations. This point is still controversial since any type of intermediate queueing might introduce undesirable delays into end-to-end conversations.

VI.3 Subscriber Computer Software

The design of the software takes into account portability, simplicity and ease of implementation. Performance is not considered to be a relevant characteristic at this stage of the Project.

Following a top-down approach, a set of design characteristics, which will emerge in the description below, have been defined. As the major impediment to portability is data definition and data structure, it has been attempted to completely separate the data from the algorithms of the program. Instead of accessing the data directly, the program uses a few small routines, giving a symbolic representation, for accessing the various data and data structures. This allows a portable description of the main program, using the data access routines as primitives.

The TS and all network control programs run in a user region utilizing PL/I multitasking facilities.

VI. 4 Data Base

The data base handles variable length strings of data, arrays, queues, dictionaries and trees and employs a "virtual memory-like" mechanism to provide a window in main-core of the total file space available. This mechanism is felt to be important as it is possible for some of the data to remain in memory for many seconds waiting for some event.

Memory allocation is performed within the data base space manager, which in the present generation has at its disposal 20 Kbytes of core store, 1 million bytes of disc store and can handle 60 K structural relationships. It is not expected to have any problems with storage space, as this space is available to all "machines" connected to the network for the storage of working space, and also contains the input queues to the machines. Everything within the data base resides on disc, page swapping being controlled by an optimizing algorithm which employs an approximate form of both the working set and least recently used rules. This gives an overhead which is comparable with that of a virtual memory machine, but an important difference is that the contents of the data base can be retained on the disc between successive runs of the network program.

VI. 5 Transport Station

The TS is implemented as an automaton, reacting on user's commands, commands generated by the line process, and to time-outs generated by the timer process. Following a top-down design approach, it was found that the easiest way to express a protocol is in a state diagram. It follows that the easiest implementation, and probably the most understandable, is to define an automaton driven by a grammar which is a tabulated form of the state diagram. The automaton consists of a set of actions, and an activation mechanism which calls a particular action depending on the current point in the grammar (state of the "machine") and the command being processed. Following the aims of portability, simplicity and the ease of implementation, the only physical interface which the TS has, is with the data base. The data base contains and manages the input queues for all the constituent processes of the network programs.

A closer examination of the TS will reveal that its structure is formed by a series of levels which could correspond to individual processes in another implementation. The grammars for these levels can be identified within the total grammar, and correspond to the concepts of PORT, LIAISON, LETTERGRAM and LETTER.

In the design of the TS, compiler techniques were used and were adapted to the much more dynamic situation which exists within the TS. This usually involves a process of validation before an action is performed. For example, it must be checked that a "liaison" is open before accepting a command to send a letter on the "liaison" from the user to the TS. It is felt that these techniques allow us a great deal of flexibility in the implementation, as in an experimental environment it is important to be able to include modifications quickly, easily and clearly.

VI. 6 Network I/O System

The attempt to bind local processes to remote processes or devices and vice versa raises a series of problems.
Unfortunately, in IBM systems, the binding of the application with the needed resources is performed at several different levels (compilation, link edit, job control, execution). Processes use normally I/O operations to communicate with external environments. The analysis of the IBM access methods reveals that no clear interface is provided between user's code and physical devices in order to "plug in" emulators or standard terminal handlers.

To make matters worse, IBM's own applications do not have any standard access method to the terminals (TSO, APL, IMS which each have their own access methods). It is therefore believed that the only way to connect existing software to the network is to intercept the I/O commands at the supervisor level and to route them to the proper virtual device. By coupling two simple virtual devices, this new access method can also be used for interpartition communication.

Following this approach BSAM and QSAM supervisor routines have been modified and an analogous attempt is carried out with BTAM and RJE station handlers. This allows existing modules to issue read/write operations on simulated units. The connection between the application and the device is defined by JCL cards which refer to virtual units defined at system generation. The IBM virtual device emulator communicates the I/O requests of the application to the proper network standard device handler.

Such an approach, which in our opinion represents the only possible in OS environment, should not be extended to new network applications. They might use a "Network Access Method" as well as the Network Control Language for describing their own environments and resources. But this is something which involves the definition of clean standardized interfaces that manufacturers themselves should provide as components of their operating systems.

A good example in this sense is the Multics I/O system that stresses symbolic, hardware-independent references to devices and files. The scheme allows dynamic assignment of resources and easy insertion of pseudo-devices.

Since it is unlikely that IBM will modify the design of its operating systems in order to define clean interfaces to which remote processes and terminals can be connected, it is our intention to modify thoroughly the present architecture in order to access all subsystems without any modification of the IBM software. The idea is to provide the configuration with a front-end processor that simulates IBM terminals from the hardware point of view. This approach has the advantage that subsystems like TSO, IMS, HASP will be accessed at a well-known level without any software maintenance problem. The front-end processor will also support the LCP, TS and higher level protocols.

VII. SOFTWARE PRIMITIVES

The model for programming a machine contains four basic components: 1) queue of messages, 2) the corresponding event which activates the machine, 3) the internal data structure and 4) the algorithm that performs the actions required by the current status and message.
The code describing a machine uses the following list of primitives:

- wait event,
- get message from the queue,
- manipulate internal data structure,
- send messages to other machines.

The data structure is represented by a tree in which a node may be a dictionary (a set of labelled nodes) or an indexed set (example array, queue, stack of nodes). Terminal nodes contain strings of octets of variable length in which substrings can be addressed.

This definition of data structure implies that each node can be described by the corresponding path into the tree, starting from the common root which is the dictionary of the machines.

This unique node identifier is built by concatenating couples (type, value):

- dictionary , name
- indexed set , index (positive means start from the first)
  (negative means start from the last)
- terminal , index (of the octet within the string).

This naming convention allows to define any type of useful data structure and to address any subcomponent, bypassing the limitations im-

posed by present-day compilers. The primitives to handle the data structure perform, via subroutine calls, the following functions:

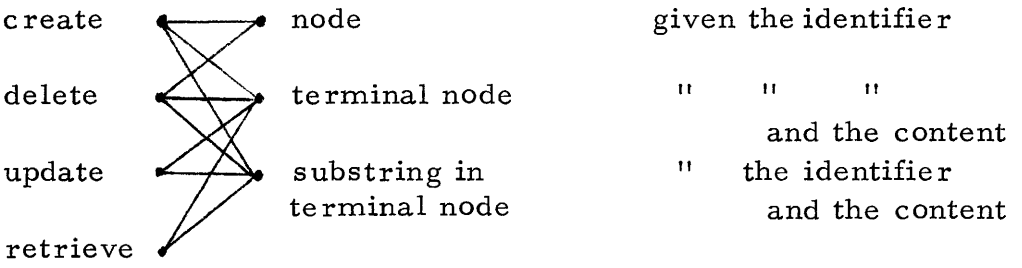


Fig. 5 is an example of a tree used by the TS to reassemble the 3rd packet of the 2nd lettergram received by the port 'A'. The text of the fragment is placed in the terminal node by a call CREATE (name, text).
The primitives to be used for sending a message to a machine or for accessing the input queue are based on the same basic functions mentioned above with the inclusion of a semaphore that prevents concurrent accesses by several processes. The send message primitive also includes the posting of the proper event.

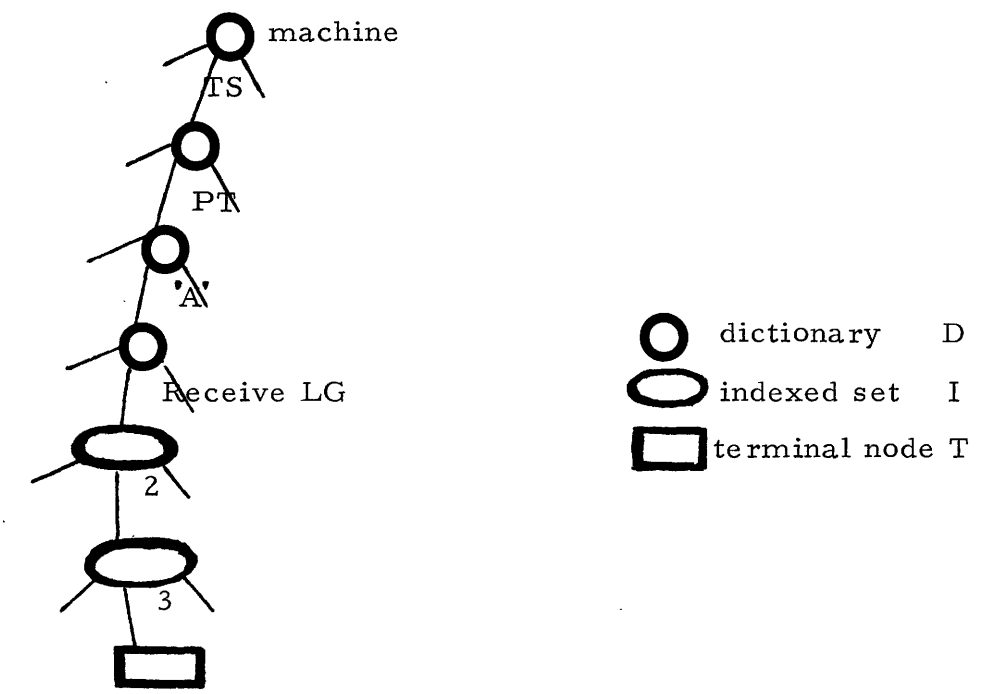


Fig. 5 - Data Structure Example and Data Description Language

VIII. THE FRONT END PROCESSOR

VIII. 1 Introduction

Since the HDLC line procedure used by the Network Switching Centre (NSC) is currently not supported by our Central Computer (SC), we had to interface between the actually supported Binary Synchronous Communication (BSC) procedure and the HDLC.

This was achieved by hardware (ELENA) and some software developed in a peripheral "real time" computer (IBM S/7).
Actually, the S/7 is used for supporting

- special terminals such as graphic devices, minicomputers which work under control of the local TP system,
- the SC - NSC connection.

Consequently, the new software for the EIN connection had to be integrated in the existing operating system of S/7. The S/7's operating system is selfmade, programmed in modular form, using for compilation, link-edit and formatting the standard S/7 host software in the 370.

VIII. 2 Hardware Configuration

The interface in hardware terms is illustrated in Fig. 6. The S/7 operates as a satellite processor linked to the IBM 370/165 via 2 BSC lines in order to perform a full duplex data transfer.

The main characteristics of the S/7 is the provision of modular data acquisition by means of digital I/O devices that allow a real time serving mechanism.
The "Electronic European Network Adapter" (ELENA) performs the transformation of the parallel 8 bit groups (bytes) in serial bit sequences according to the HDLC protocol and vice versa.

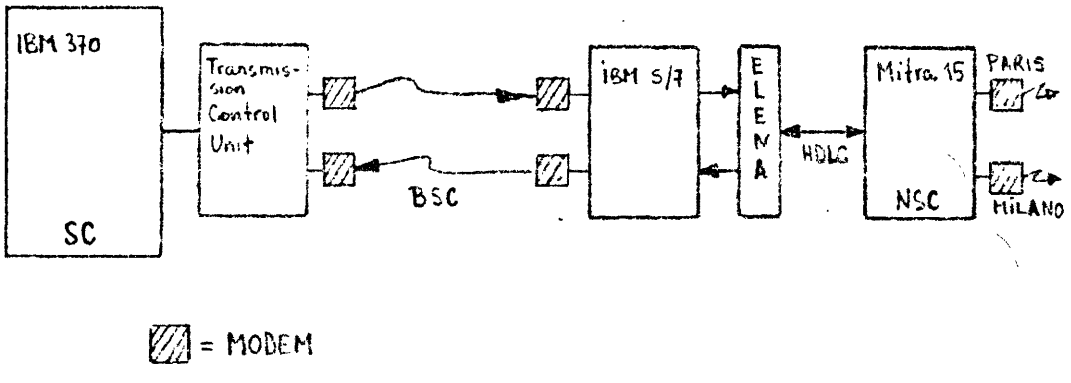


Fig. 6 - Hardware Configuration

VIII. 3 Software Developments

VIII. 3. 1 Communication Between the Central Computer and the S/7

In order to allow a full duplex data transfer, the S/7 will be related to the IBM 370/165 by 2 non-switched point-to-point BSC lines of actually 4800 bits/sec. The line speed will be increased to 9600 bits/sec as soon as the control unit will allow it.
On the 370 side, the 2 BSC lines are accessed by means of 2 Assembler written subroutines, which make use of the Basic Telecommunication Access Method (BTAM).
Those routines execute in a PL/I environment as different subtasks.

In the S/7 a rentrant BSC module handles the 2 BSC lines according to the following BSC protocol:

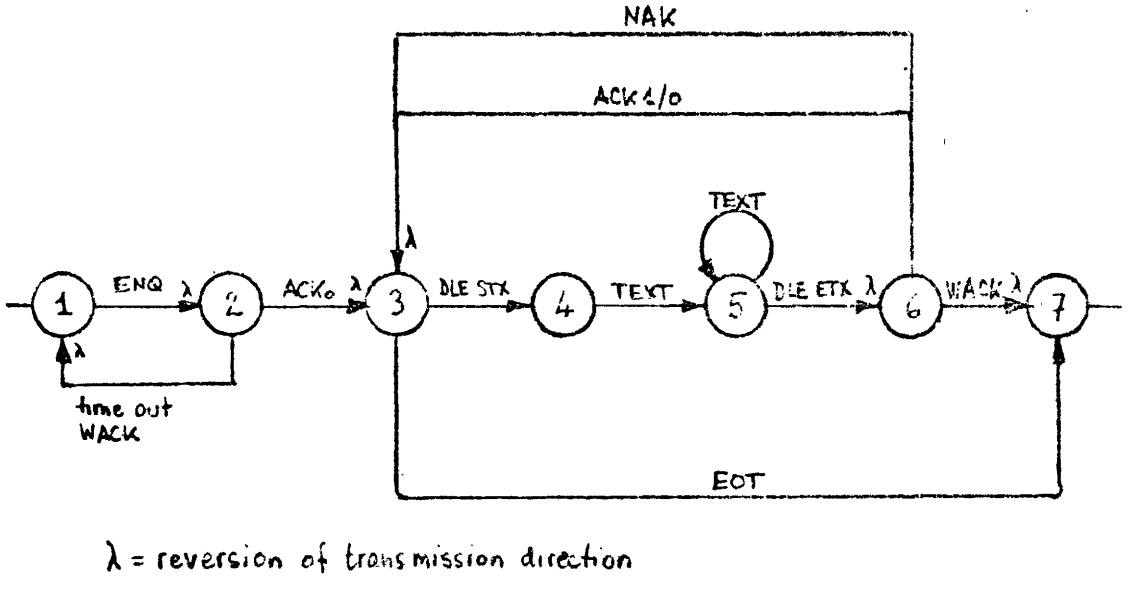


Fig. 7 - BSC - Protocol

VIII. 3. 2 S/7 ELENA Communication

For the communication in both directions, 4 digital Input/Output devices of 16 bits each are used:

DIRW digital input device with interrupt feature,

DIR digital input device,

DOR digital output device,

DOW digital output device.

The meaningful bit positions are as follows:

DIRW	0	HIN	HEU IN SERVICE
	1	NXWR	REQUEST NEXT BYTE IN WRITE flip-flop
	2	OVWR	OVERRUN IN WRITE
	3	GWR	GOOD TERMINATION OF WRITE
	4	RDR	READ READY
	5	NXBR	NEXT BYTE IN READ flip-flop
	6-15		not used
DIR	0	HINR	HIN REPETITION
	1	OVRD	OVERRUN IN READ
	2	CRCER	CRC IN ERROR
	3	ANFL	ANNULATION FLAG
	4-7		not used
	8-15		DATA IN READ
DOR	15	S7R	SYSTEM/7 READY
	14	SNXBR	SEND NEXT BYTE IN READ flip-flop
	13	ERDS	END OF RECORD IN READ SENDED
	12-0		not used
DOW	0	S7W	SYSTEM/7 READY TO WRITE
	1	WRON	WRITE ON USE
	2	NXBW	NEXT BYTE FOR WRITE flip-flop
	3	WAN	WAIT ANNULATION CONFIGURATION
	4	WER	WRITE END OF RECORD
	5-7		not used
	8-15		DATA FOR WRITE

The protocol defining the permitted bit changing events works on a ping-pong basis. Figs. 8 and 9 give a state transition graph for output from and input to the S/7, respectively.

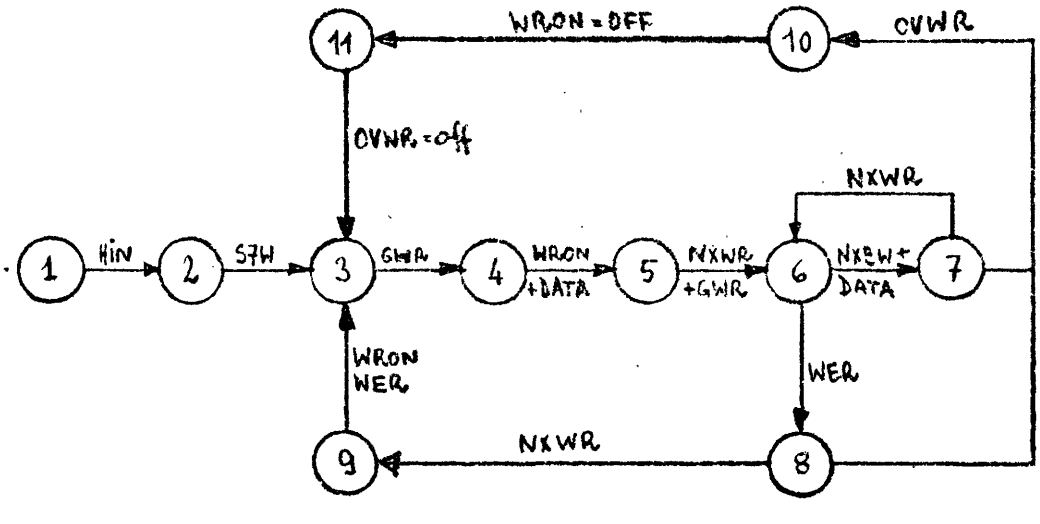


Fig. 8 - Output from S/7 to ELENA

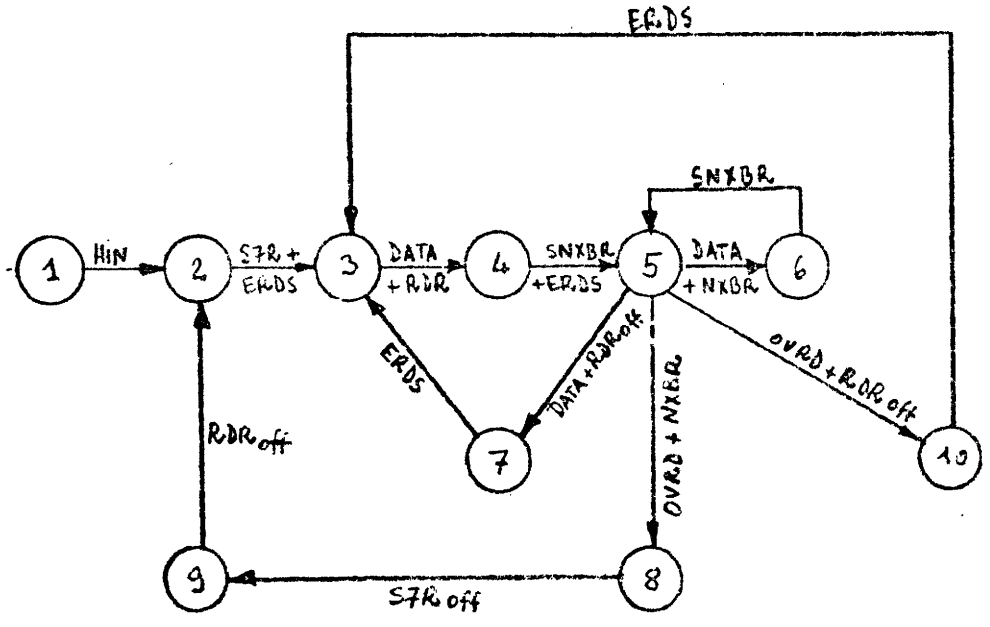


Fig. 9 - Input from ELENA to S/7

VIII. 3. 3 HDLC Line Protocol

The HDLC line protocol is implemented in the S/7, with exception of the logical channel allocation for outgoing SC to NSC packets, which is left to the 370 communication station.

After having received a special start command from the SC, the S/7 tries to bring up the NSC line and maintains the contact with the node by generating bubbles. Packets from the node are acknowledged and forwarded to the SC, while duplicate packets and bubbles are dropped. However significant ACK information is extracted and forwarded to the SC by means of an extra bubble, if necessary.

A security timer controls the NSC communication as well as the BSC Communication to the SC. Exhausting the timer count in either cases will reset the line in the down state. This will be communicated to the 370 by means of a special "down bubble" (FTYPE 8). No messages are transported through the S/7 in the down state, the 370 can, however, try to bring up the line again by issuing the start command.

IX. THE HDLC ADAPTER

IX. 1 Description

The HDLC adapter is a hardware unit, called ELENA (Electronic European Network Adapter), connected between the Network Switching Centre (NSC) and the front end processor of the Subscriber Computer (SC). This adapter which has been developed and realized in the JRC-Ispra, carries

out the HDLC hardware specifications of the network and communicates with the front end processor (IBM S/7) through the standard input/output modules.

The data exchange with the network switching centre is based on a synchronous serial data transmission over modem interfaces whereas the data exchange with the front end processor is character by character in handshake mode.
The main functions achieved by ELENA are:

1) flag sequence: all frames will start and end with the flag sequence; a single flag may be used as both the closing flag for one frame and the opening flag for the next frame.

2) transparency: the transmitter examines the frame content between two flag sequences and inserts a "0" bit after all sequences of 5 adjacent "1" bits to ensure that a flag sequence is not simulated; the receiver examines the frame content and discards any "0" bit which directly follows 5 adjacent "1" bits.

3) frame checking sequence: it is the remainder after the multiplication by X^{16} and then division by the generating polynomial $X^{16} + X^{12} + X^5 + 1$ of the frame content between two flag sequences and excluding bits inserted for transparency.

4) interframe fill time: the time between frames is filled by transmitting continuous flag or annulation sequences or a combination of both.

5) invalid frame: the transmitter of ELENA aborts a frame upon command of the front end processor by transmitting an annulation sequence; the receiver will ignore a frame from the network which contains an annulation sequence.

6) protocol with the front end processor: it includes all signals required for bidirectional data transfer and status indication between ELENA and S/7.

IX.2 Data Exchange Between NSC, HDLC Adapter and S/7

The exchange between the NSC and ELENA conforms to the provisions of CCITT V 24 recommendation for synchronous data transmission. The HDLC adapter includes the timing source for the synchronization of transmitted and received data. A communication interface for modem connection is provided; since the distance between NSC and HDLC adapter is short, both are directly connected.

The signal exchange between the HDLC adapter and S/7 is defined in the previous chapter "Front End Processor" point 2. Since ELENA has been designed to operate in full duplex, the logic circuits for data transfer from S/7 to NSC are completely independent from the logic for data

transfer from NSC to S/7. Only the signal HIN indicating that ELENA is ready to operate, is common to both parts.

The HDLC adapter transfers to S/7 only the frame content between two flag sequences after discarding "0" for transparency. The data flow from the network is a bit stream which contains more bits than the data flow to S/7. The HDLC adapter transmits to S/7 an entire number of octets but receives from the net a number of bits which is not always a multiple of 8, since it depends on the bit configuration of the message (one supplementary "0" for each group of 5 adjacent "1"). The input module of the front end processor accepts data from ELENA character after character (octet) but the message structure of the network is only bit-oriented; there can be interactions between two successive groups of 8 bits on the line if for instance the last three bits of the previous group are all "1" and the two first of the next group are "1" the third bit of this group has to be discarded. These considerations lead to a receiver logic of ELENA which operates on the bit level and converts from serial to parallel only at the final stage of interchanging circuits with S/7. Another problem consists in the possibility that the closing flag of one frame is also the opening flag of the next frame; a supplementary logic takes care of the operations required at the beginning of a frame and at the end of a frame. The transfer rate of characters to S/7 according to the handshake mode must take into account the synchronous band rate of the switching network centre; if there is a delay over several characters an overrun in read occurs.

In the other direction of data transfer, from front end processor to network switching centre, the adapter transmits more bits to the net than it receives from S/7. This logic part is dual to the previous one. Since the transmission mode between adapter and NSC does not accept any time interval between the bits of the frame content, an overrun in write can occur if S/7 does not present the characters in output with a sufficiently high transfer rate.

IX.3 Flow Chart WRITE (see Figure 10)

The write operation is defined as data transfer from S/7 to ELENA and is implemented by the transmitter logic. By applying power all circuits are reset and an annulation character (SAN) is sent to the NSC. The initial conditions of some control signals (HIN, WAN, WER, WRON), are set. Until S/7 is ready to write (S7W), the adapter sends annulation sequences on the line; if S/7 is ready to write flag sequences are transmitted. Data transfer between S/7 and ELENA begins with WRON on.

First one flag sequence is sent, then the data on the output of S/7 is loaded by the HDLC adapter. As soon as the transmitter logic is ready to accept the next character from S/7 it changes the level of NXWR.

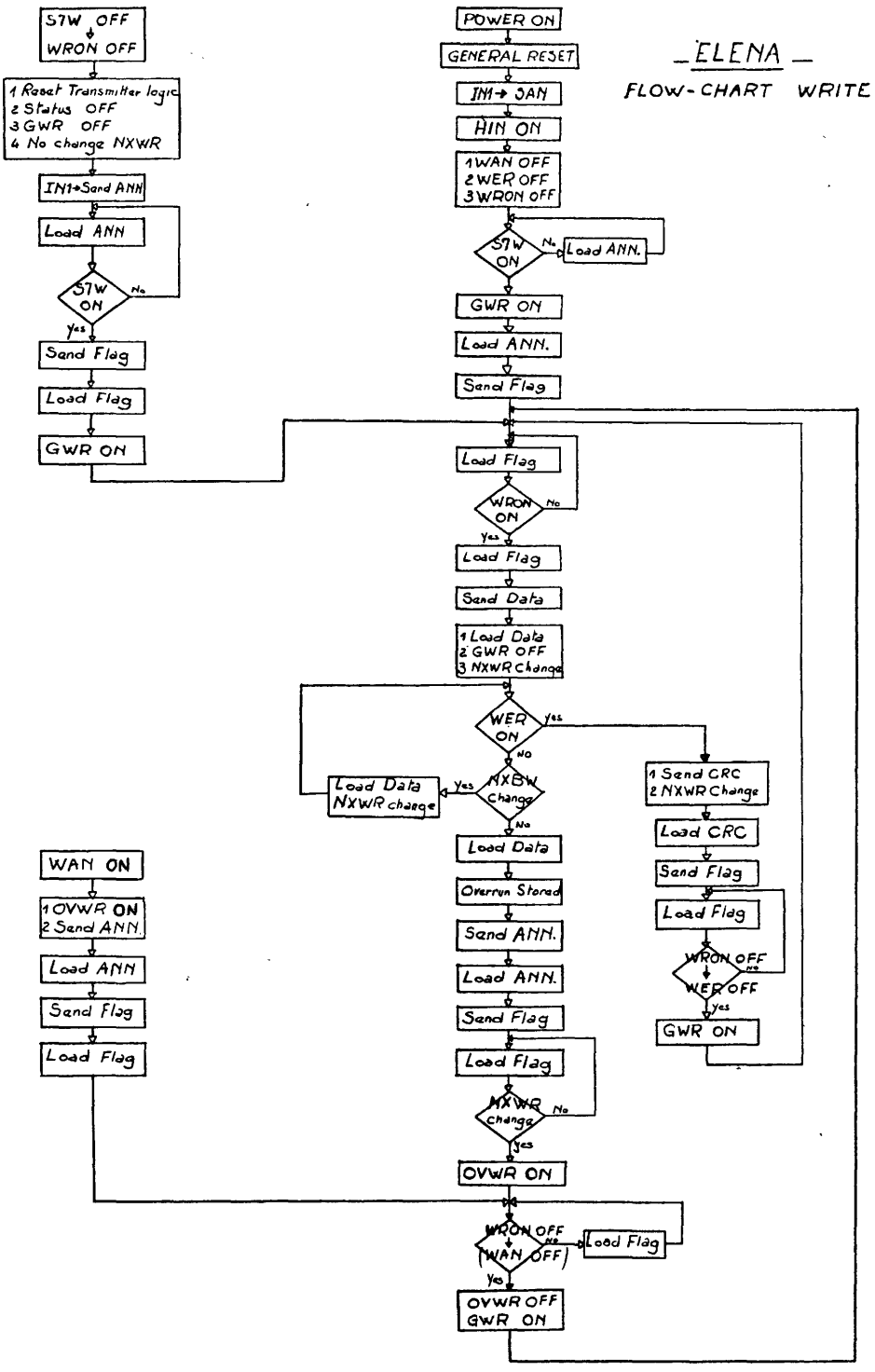


Fig. 10 - ELENA - Flow Chart WRITE

S/7 communicates the presentation of the following character by changing NXBW. This handshake goes on until S/7 turns on WER which means that the adapter has to prepare the end of the frame structure after completion of transmission of the last character. The end of the frame consists in two characters CRC of the cyclic redundancy check and of a flag sequence. The CRC characters are generated during the transmission of all characters in a special shift register, the content of which is appended to the last data character.

Upon acknowledgement of WER S/7 terminates the transmission of this block by turning off WRON and the adapter indicates that the frame structure has been correctly terminated by setting GWR on. From this moment a new cycle can start.

The described logical flow is valid for normal operation. Three conditions can induce the transmitter logic to follow other logical branches: overrun in write, command of aborting a frame, S/7 no longer ready to write. The first condition occurs if two consecutive characters are too much delayed by S/7; in this case the transmitter logic sends an annulation sequence followed by flag sequences to the NSC and sets the status bit OVWR on. The front end processor can abort a frame by turning on WAN. The HDLC adapter follows the same procedure as for overrun. The last condition is set by an anomalous operation of S/7 which turns off S7W; this resets the whole transmitter logic and then follows a similar procedure as the initial one by applying the power to ELENA.

IX.4 Flow Chart READ (see Figure 11)

The read operation is defined as data transfer from the NSC to S/7 and is implemented by the receiver logic. By applying power all receiver circuits are reset and all status bits (ANF, CRCER, OVRD) are off. Only if S/7 is ready to read (S7R) the serial information from the NSC is preprocessed. Upon reception of a flag sequence the circuit of the cyclic redundancy check is cleared. If the following octet is different from the flag sequence, the CRC accumulation begins. The "0" bits for transparency are discarded and the first character is loaded in the output latch of the receiver logic. Through the generation of an internal signal DAR the HDLC adapter turns on Read Ready (RDR) which remains on for all the time of data transfer. S/7 answers with "send next byte in read" (SNXBR) and ELENA changes the level of "next byte in read" (NXBR) if the following character is ready in the output latch. This handshake goes on until the receiver logic detects the last character of the frame content which precedes a flag sequence. This is possible without inserting a wait time because the information towards S/7 transits through several shift registers, so the end of a frame is detected several characters before the transfer of the last character to S/7. At this moment the result of the cyclic redundancy check is examined and the corresponding status bit is set.

_ ELENA _

FLOW-CHART READ

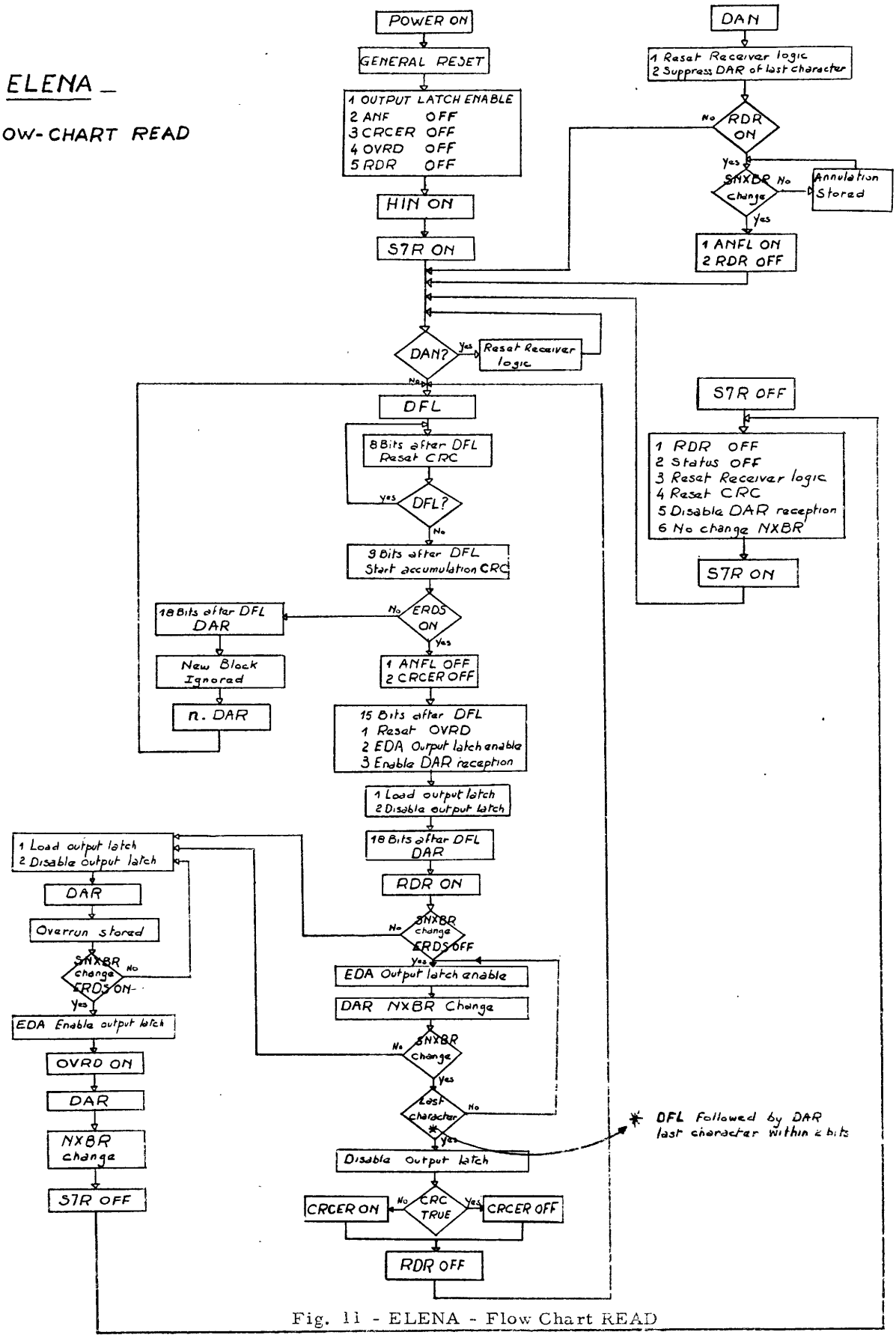


Fig. 11 - ELENA - Flow Chart READ

With the transfer of the last character ELENA turns off RDR.
S/7 answers with "end of record in read sensed" (ERDS). From this moment on a new cycle can start.

This is the logical flow for normal operation. Upon reception of an annulation sequence (DAN) in a frame structure the status bit annulation is set, RDR is turned off and the receiver logic is reset.
If S/7 does not follow the required transfer rate for two consecutive characters an overrun in read occurs. The corresponding status bit (OVRD) is set by ELENA and S/7 acknowledges it by turning off S7R. The same signal can also be turned off if an anomalous operation of S/7 occurs. This resets the receiver logic and follows a similar procedure as the initial reset.

IX. 5 Transmitter, Receiver and IBM Logic

The HDLC adapter is divided into three logical blocks:

- the transmitter logic which transmits data to the network according to the HDLC hardware specifications,
- the receiver logic which receives data from the net,
- the IBM logic which carries out the protocol with IBM S/7.

The transmitter logic (see Fig. 12) sends to the NSC annulation, flag, data and CRC sequences. The selection of the sequence is given by the IBM logic (SAN, SFL, SDA or SCR). The character is serialized in the central shift register, then extra '0' are inserted for transparency and the serial bit stream (TDA) is transferred to the NSC. The status of the sequence in course (LAN, LFL, LDA, LCR) is indicated to the IBM logic.

The receiver logic (see Fig. 13) provides a decoder on the serial data from the net (RDA) for the annulation or flag sequence. The result of decoding (DAN, DFL) is indicated to the IBM logic. The central shift register detects 5 adjacent '1' and discards the following bit inhibiting the upper shift register. The data transfer into the holding register is enabled by the IBM logic (EDA) and the transfer is acknowledged by the signal DAR.

The IBM logic performs the logical operations on all signals to and from S/7 as well as on the described signals of transmitter and receiver logic.

IX. 6 Realization Technique

The HDLC adapter hardware (see photograph) consists of the logical unit, the power supply unit with its control and a test unit which displays all exchange signals with the front end processor and the serial data stream with the NSC.

A first version of the HDLC adapter has been designed and realized in TTL logic with 270 integrated circuits. The wiring of the units has been carried out in wire-wrapping technique which allows easy modifications or extensions. This hardware adapter can operate in full duplex at transmission speeds up to 600,000 bit/sec.

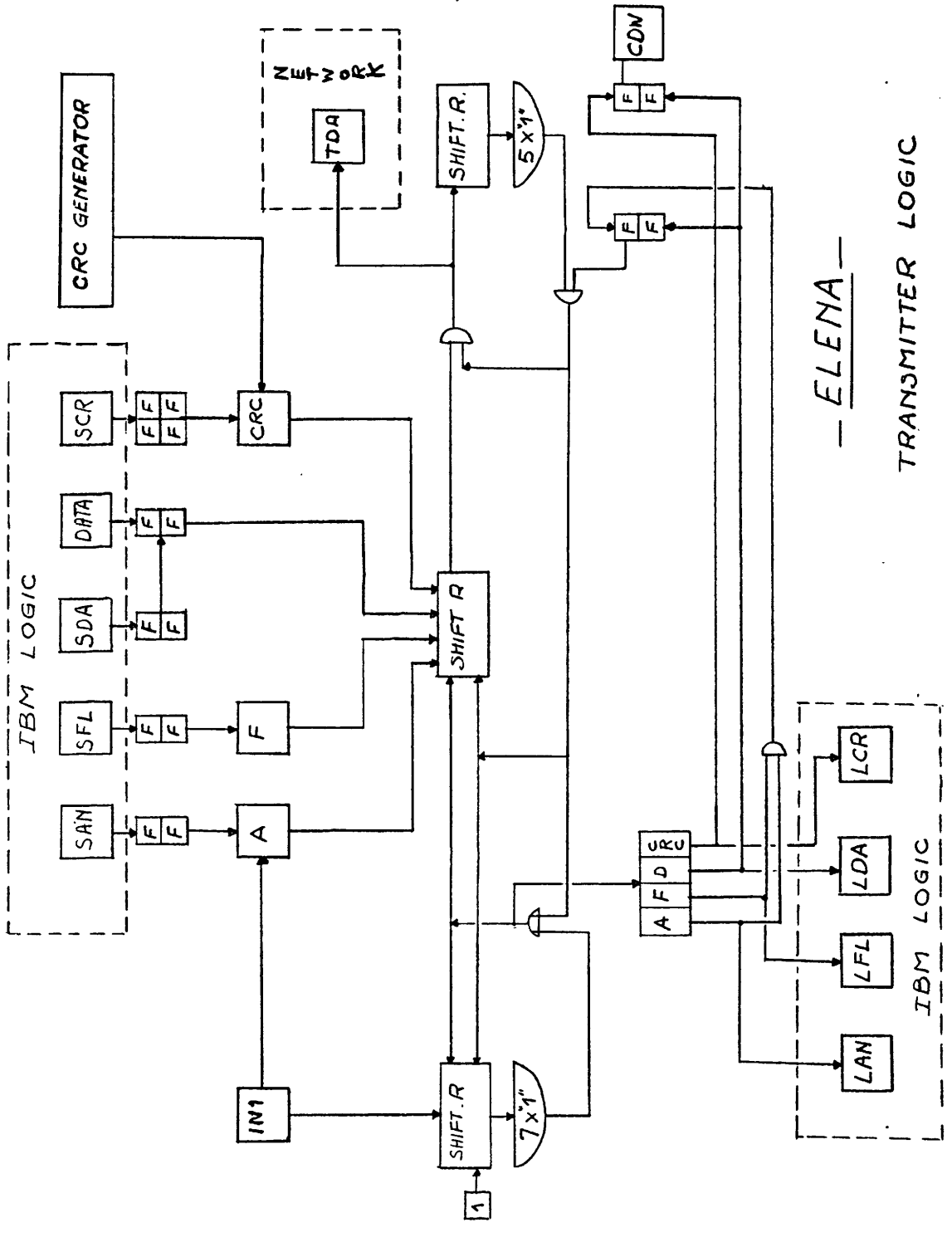


Fig. 12 - ELENA - Transmitter Logic

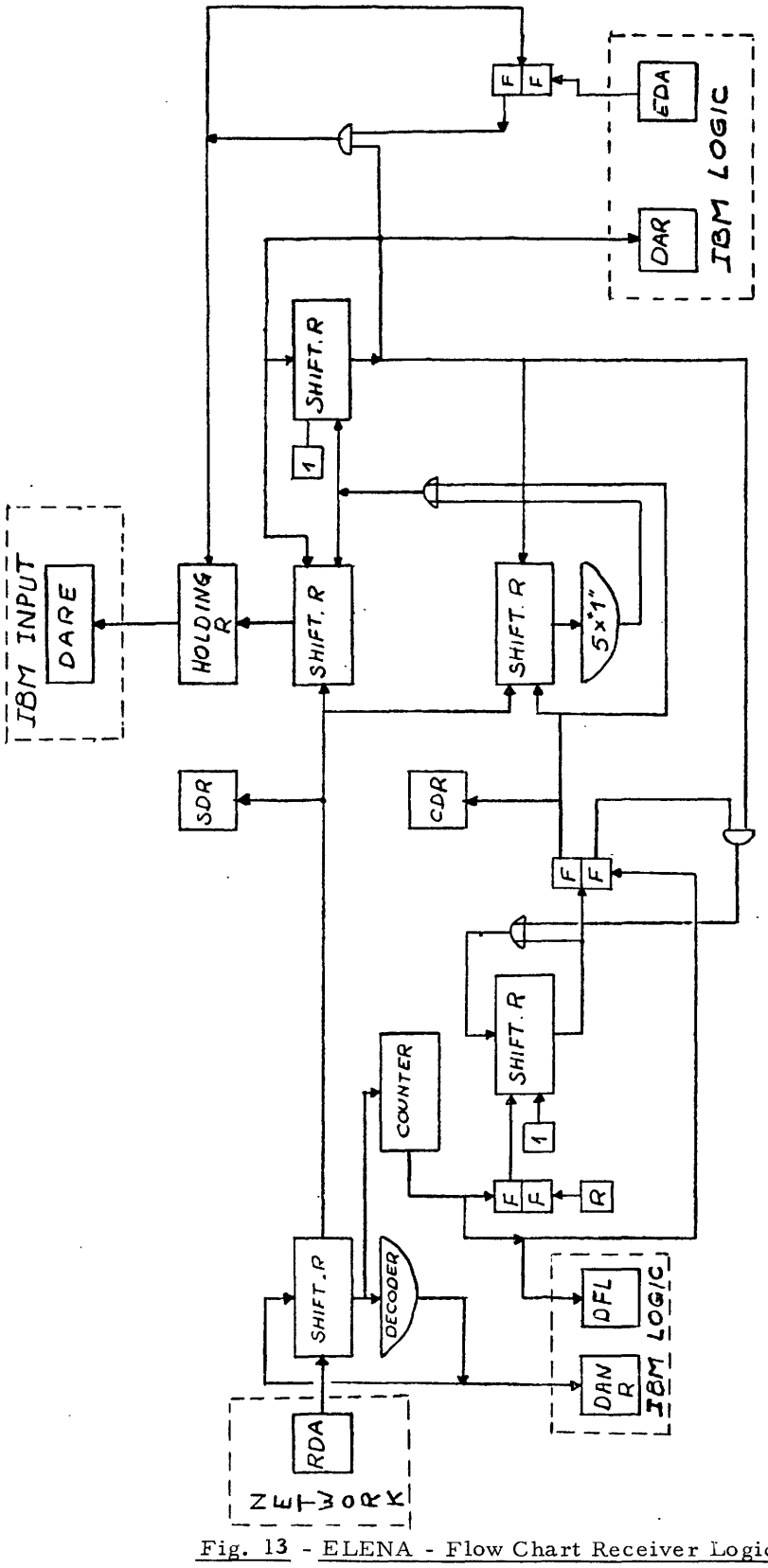
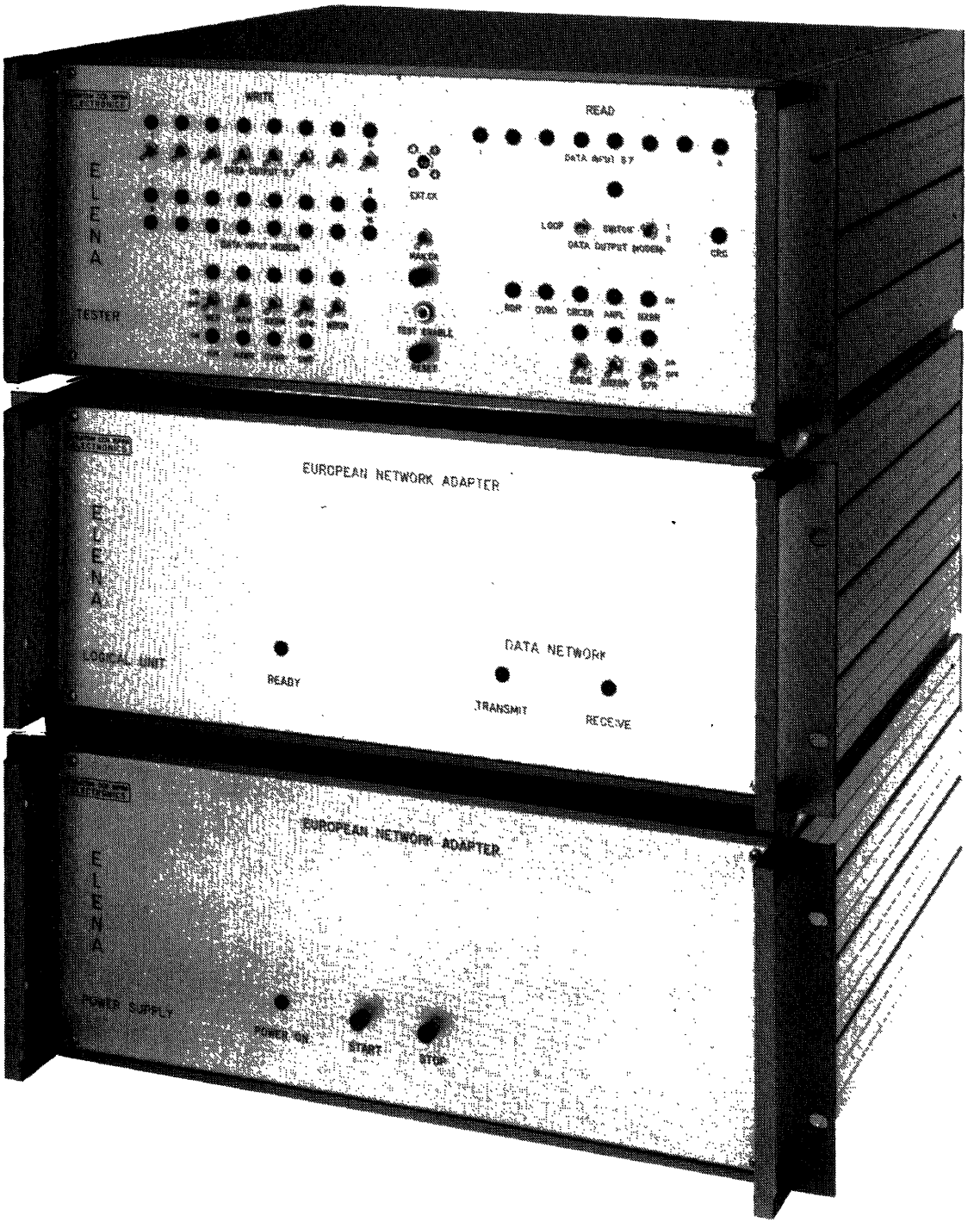


Fig. 13 - ELENA - Flow Chart Receiver Logic

- ELENA -

RECEIVER LOGIC


WRITE
READ
EXT.CK
TEST MODULE
E L E N A
TESTER
EUROPEAN NETWORK ADAPTER
E L E N A
LOGICAL UNIT
READY
DATA NETWORK
TRANSMIT
RECEIVE
EUROPEAN NETWORK ADAPTER
E L E N A
POWER SUPPLY
POWER ON
START
STOP

A second version has been developed with a microprocessor in MOS-technique using 70 integrated circuits. The hardware cost of this version amounts to 60% of that of the TTL-version, but the maximum transmission speed is 9,600 bit/sec. A greater flexibility is assured by its program stored in a programmable read-only-memory (PROM); the program contains 386 instructions stored in 762 bytes of memory.

X. TRANSPORT STATION - ISPRA PROPOSALS

X.1 Lettergram

The Transport Station implements a service of "Lettergrams" exactly like the sub-network supports a service of "Datagrams". It is responsible for expediting letters, that is to say:

1) it chops into packets the letters which are given to it by the user ports;
2) it reassembles the packets in letters in order to give them to the user ports;
3) only unique correct (and complete) letters are delivered.

These characteristics constitute the basic service.
To this basic service, standard options can be added.

1) Non-delivery diagnostic: LNDD

   The transport station will return the letter to the sender with an explanatory diagnostic when the letter has not been delivered to the destination.

2) Delivery Confirmation: LDC

   The transport station returns the letter to the sender with a guarantee that an exact copy has been delivered to the destination.

3) Letter Ordering: LO

   Letters are not necessarily delivered in the order in which they have been sent by the transport station. If a port wants letters to be delivered in the order in which it gives them to the TS, the TSs involved establish a special protocol for these letters.
   We will refer to this protocol by the name "liaison". The ordering does not guarantee that every letter arrives, but only that letter N + i is delivered after letter N.
   The transmission of all letters is only guaranteed if the options LNDD and LO are requested at the same time.
   The combination of the two options guarantees that only the letters preceding the one that is the object of an LNDD have been delivered.

Only one "liaison" can exist between two particular ports. However, a port can establish n "liaisons" with n different ports. The existence of a "liaison" between two ports does not exclude the possibility of sending letters "outside the liaison" to the destination port. The times of arrival of these letters "outside the liaison" cannot be foreseen, no correlation of order exists between the "letters outside the liaison" and the letters "within the liaison".

Note: A "liaison" is unidirectional sending; the existence of an "ordered liaison" in one sense does not imply the existence of an "ordered liaison" in the opposite sense.
The "liaison" is a service given to the sender; the destination is advised of the conditions of transport of each letter which it receives, consequently it can distinguish the "letters outside liaison" (LHL) from the "letters within liaison" (LL).

X.2 Telegram

The transport station also implements a service of telegrams. A telegram is a short message; it has priority of sending over all letters sent by the same port.

No options are available for telegrams. The text of a telegram cannot exceed the contents of a "telegram-packet" (it must fit into a single network packet, taking into account the "telegram header" inherent in the protocol between transport stations.

X.3 Protocol and Commands

All objects transported between two transport stations make use of a bidirectional flow between these two transport stations (connection). The control parameters are negotiated at the establishment of the connection; they can be different for each direction and certain ones can be modified during a connection.

- Parameters

CN-NB	connection number. This parameter only appears in the control messages establishing or closing the connection. If a transport station (TSA) wants to establish a connection with another transport station (TSB), it must have a CN-NB greater than that used previously and which can be memorized by TSB.
LT-LGR	maximum letter length in reception. The maximum length of letters accepted. At the same time as it requests the connection, TSA specifies the

maximum length of letters that it will accept; TSB replies, specifying the maximum length of letters that it will accept.

LT-LGS — letter length on sending.
The maximum length of letters sent. At the same time that it requests the connection, TSA suggests the maximum length that it would like to be able to use for letters which it will send.

Note: LT-LGR is not negotiable. The LT-LGR is accepted or refused; the LT-LGS is only an indication. The LT-LGR cannot be modified during a connection.

CLT-NBS — letter number.
When the connection is established, all letters will be identified by a CLT-NB which is progressive and cyclic (beginning with 1).

CCR-NBR — credit number.
The CCR-NBR indicates the maximum number of letters that a TS is prepared to receive in parallel at a given moment; the CCR-NBR can be modified during a connection.

Note: The CCR-NBR can be reduced to zero; this does not imply the end of the connection, but a "hold" state for all new letters.

CCR-NBS — suggested credit number.
Credit number suggested with the request for opening the connection.
It is only an indication, only the CCR-NBR in the reply is binding.

- Commands

CN-INI1 — connection initiation 1.
FORMAT:
CN-INI1/YR-CN-NB/MY-CN-NB/MY-LT-LGS/
CCR-NBS/MY-LT-LGR/CCR-NBR
Used for requesting a connection. YR-CN-NB can be zero, or any other value, such that the MY-CN-NB used in the reply must be equal or greater than the suggested value.

CN-INI2 — connection initiation 2.
FORMAT:
CN-INI2/YR-CN-NB/MY-CN-NB/YR-LT-LGR/
MY-LT-LGR/YR-CCR-NBR/MY-CCR-NBR.
Used to reply to a connection request.

CN-TER1 connection termination 1.
FORMAT:
CN-TER1/YR-CN-NB/MY-CN-NB/MY-LST-CTRL-NB/
CN-TERM-REASON
The use is foreseen in requesting termination of a connection in a graceful manner.

MY-LST-CTRL-NB gives the number of the last control message sent.
Only exchange of CN-CTRL messages are allowed after sending these messages.

CN-TER2 connection termination 2.
FORMAT:
CN-TER2/YR-CN-NB/MY-CN-NB/YR-LST-CTRL-NB.
Used to terminate effectively the connection.

CN-TER3 connection termination 3.
FORMAT:
CN-TER3/YR-CN-NB/CN-TERM-REASON.
Used to indicate the refusal of a connection, or to reply to an unexpected CN-TERM1 or CN-TERM2 (connection previously terminated).

CN-CTRL connection control.
FORMAT:
CN-CTRL/CN-CTRL-NB/CN-CTRL-TEXT.
Used to send "questions", "replies", and acknowledgements text. The "questions" and "replies" will be discussed later.

CN-FR letter fragment.
FORMAT:
CN-FR/LT-NB/LT-LNG/FR-NB/TEXT.
Used for sending a letter.

TEXT if FR-NB = 1 YR-PT-ID/MY-PT-ID/LT
else LT-TEXT;
LT if ordering is requested: LI-HD/LT-TEXT
else LT-TEXT (for LI-HD see liaison).

CN-TG telegram.
FORMAT:
CN-TG/YR-PT-ID/MY-PT-ID/TG-TEXT.
Used for sending a telegram.

- Liaison

A liaison is established and controlled by the LI-HD (liaison header).

LI-INI1 — liaison initiation 1.
FORMAT:
LI-INI1/MY-LT-LNG/LI-CR-NB/
Used for requesting a liaison.

LI-TER1 — liaison termination 1.
FORMAT:
LI-TERM/TERM-REASON/
Request to terminate a liaison.

LI-FR — fragment of a letter in liaison.
FORMAT:
LI-FR/LI-CR-NB/YR-REF/MY-REF/
Used for sending a letter.

LI-TG — telegram on flow.
FORMAT:
LI-TG/TG-TEXT/
Used for sending a telegram in a flow.

LI-ACK — acknowledge letter within liaison.
FORMAT:
LI-ACK/LI-CR-NB/YR-REF/

LI-ERROR — error on liaison.
FORMAT:
LI-ERROR/ERROR-CODE/FIRST N-BYTES.

Note: the parameters YR-PT-ID/PT-ID are preceding the command code and part of the normal letter format.

- Control Information

In order to guarantee correct functioning and efficiency of a good level the TS must exchange some information. The frequency of these exchanges depend to a large part on the traffic and the circumstances.

The information principally concerned is:

1) The status of exchanged letters;

2) The position of pointers and the value of credits.

The information can be given spontaneously or on demand. The average length of a piece of information is 3 characters (a type indicator and the information). Consequently, a control message can contain 80 pieces of information. Allowing for the fact that control messages can be lost, a TS must be able to request a control message, or the resending of a control message.

All letters sent by a TS must be the object of a control information message on the part of the destination TS (DC or NDD). The DC can be given "en bloc" or individually.

Each control message contains at least the following information: oldest letter in core and credit number.
This information is added just before the message is sent.

The control message packet that collects current control information will be sent at the time one of the following events occurs:

- time out/packet full/explicit request of the receiver/requested

  NDD or DC message on letter is ready.

The control information can be divided into:

i) basic information;

ii) additional information.

All TS must be capable of recognizing the questions related to the basic information, and of answering them.

- Basic Information

ACK-CTRL/CRTL-NB	serves to guarantee that a CN-CTRL has been received.
ACK-LT/LT-NB	serves to guarantee that a CN-LT has been received and delivered correctly.
NACKx-LT/LT-NB	serves as NDD; x specifies the reason.
INF-B/OLDEST-IN-CORE/CR-NB	gives the number of the oldest letter still incomplete and the credit number. All the preceding letters have been delivered or destroyed. If they were sent under ERROR CONTROL, they at least have been the object of an ACK-LT or NACKx-LT (*)
DESTR-LT/LT-NB	order to destroy a letter if it is still in memory.
REQ-CTRL/CRTL-NB	order to resend all the CN-CTRL since the number given, including this number.

(*) Note: Before updating the OLDEST-IN-CORE counter, NACKx is produced for all those letters for which no fragment at all has been received.

All the CN-CTRL must be kept as long as an ACK-CTRL has not been received for them. An ACK-CTRL applies to all the previous CN-CTRL.

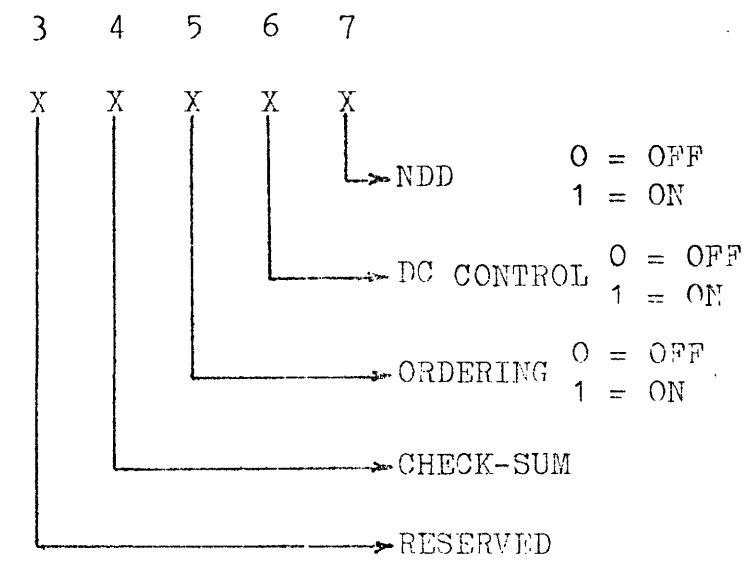
- Additional Information

See tables of codes.

TS PROTOCOL

FIRST-BYTE - OP-CODE - BITS 0-2/BASIC FACILITIES BIT 3-7

	0	1	2
CN-INI1	0	0	0
CN-INI2	0	0	1
CN-CTRL	0	1	0
CN-TG	0	1	1
CN-LT	1	0	0
CN-TER1	1	0	1
CN-TER2*	1	1	0
CN-TER3	1	1	1



* not mandatory

FORMAT OF CONNECTION - TEXT

	1	2	3	4	5	6	7	8	9	10	11	12	13
CN-INI1	OP C/O	YR CN-NB				MY CN-NB				MY CR-NBS	MY CR-NBR	MY MAX LT-LGS	MY MAX LT-LNG
CN-INI2	OP C/O	YR CN-NB				MY CN-NB				YR CR-NBR	MY CR-NBR	YR LT-LGR	MY MAX LT-LNG
CN-TERM1	OP C/O	YR CN-NB				MY CN-NB				MY-LAST CTRL-NB		REASON	
CN-TERM2	OP C/O	YR CN-NB				MY CN-NB				YR-LAST CTRL-NB			
CN-TERM3	OP C/O	YR CN-NB				REASON							
CN-CTRL	OP C/O	CN-CTRL NB	CN-CTRL-TEXT										
CN-TG	OP C/O-4	YR PT-NB		MY PN-NB		TG-TEXT							
CN-LT	OP BP	LT-NB		LT-LNG	FR-NB	LETTER-HEADER OR LT-TEXT							
LETTER-HEADER (IF FR-NB = 1)	YR PT-NB		MY PT-NB		LT-TEXT								

ORDERING - PROTOCOL

LETTER-TEXT WITH ORDERING

BYTES ⟶ 1 2 3

	OP / CR	YR REF	MY REF					
OP-CODES ↓ BITS→	0	1	2	3	4	5	6	7
FL-LT	0	0	1	1	CR-NB			
FL-ACK	0	0	1	0	CR-NB			
FL-TG	0	0	0	0	NOT USED			
FL-INIT	0	1	0	0	0	0	0	0
FL-TERM	0	1	0	0	0	0	0	1
FL-ERROR	0	1	0	0	0	0	1	0
PG-LT	1	0	1	1	CR-NB			
PG-ACK	1	0	1	0	CR-NB			
PG-TG	1	0	0	0	NOT USED			

INFORMATION CODES AND FORMAT

	OP-CODE	
NO INFORMAT. ABOUT	00	LT-NB
LETTER IN CORE	01	LT-NB
ACK * LETTER	02	LT-NB
NACK 1 TOO OLD *	03	LT-NB
NACK 2 DESTROYED ON YR REQUEST	04	LT-NB
NACK 3 DESTROYED ON REQUEST OF PORT	05	LT-NB
NACK 4 DESTROYED PORT CLOSED	06	LT-NB
NO LETTER IN CORE	08	NEXT LT-NB CR-NB
OLDEST LETTER IN CORE *	09	LT-NB CR-NB
NEWEST LETTER IN CORE	0A	LT-NB
ACK CN-CTRL	12	CTRL-NB
ACK ALL * CN-CTRL TILL	16	CTRL-NB
LAST CN-CTRL IS	18	CTRL-NB

* BASIC INFORM.

QUESTIONS AND REQUESTS - BIT 0 IS QUESTION MARK

CODES AND FORMAT

SEND STATUS OF LETTER	80	LT-NB	
DESTROY LETTER	84	LT-NB	
RESEND LETTER	88	LT-NB	
RESEND PARTIAL LETTER	8C	LT-NC	
N FR	8D	N	n FR-NB ... FR-NB
ALL FR FROM	8F	FR-NB	
RESEND CN-CTRL	90	CTRL-NB	
SEND ALL * CN-CTRL FROM	91	CTRL-NB	
SEND NB OF LAST-CN-CTRL	98		

* BASIC REQUEST

XI. SUBNETWORK TEST MODULE - ISPRA PROPOSAL

If we wish to test completely and efficiently the sub-network we need a tool able to provide artificial traffic corresponding to some preset matrix and able to collect data which can be analyzed in order to give an idea of the behaviour and the reactions of the network under various loads.

XI.1 Techniques Proposed for the Study of the Network

1. Produce automatically packets using the various facilities offered by the network.
2. Use these packets under "no load" conditions and measure the response time of the network, verify accuracy of the answer (e.g. timer), verify the routing algorithm (trace) and the sequencing of packet by the ordering mechanism.
3. Produce artificial traffic at the level of letters using the basic facilities of the subscriber computers i.e. DROP, ECHO, REBOUND, EMIT and get statistical results on various loads.
4. Superimpose various traffics and analyze eventual degradation in service.
5. Produce some low-load controlled traffic and study the fluctuation of the response during real use of the network.

XI.2 Production of Packets

We must have the possibility to produce "wrong packets" in order to know how the network reacts, as well as correct packets. It is clear that the errors to be introduced are mainly in the headers, so we need a language to specify easily the contents of the header.

When we need to specify "good packets" we need facilities to specify in an easy way the contents of the header and be sure that incidental errors do not occur.
The text of the packet is generally not of great importance and we need a facility to provide it automatically if needed.

XI.3 Production of Letters

The letters to be exchanged have the main objective of producing traffic load; we must be sure that the traffic is really produced and that material errors are as far as possible eliminated in such a way that the experiment proceeds as smoothly as possible.

XI.4 Experiments

The experiments may be conducted by one centre or in collaboration be-

tween centres. Experiments may be conducted without involving people from other centres, if the basic subscribers are implemented on each computer.

The most important is "Rebound" which allows the creation of traffic on any link from any Control Module.

XI. 5 Hardware Needed

At least a teletype is needed for the conversational control of the experiments. Some medium speed printer will be useful for printing the results.
Moreover a file on tape or on disc would be needed to record the data which can be processed at the end of the experiment.

XI. 6 Command Language

The Control Module is conceived as an interactive process obeying commands given from a terminal (teletype writer or video). Mainly the command language is related to:

1. description of the contents of single packets,
2. description of the contents of single letters,
3. description of liaisons to be opened,
4. description of the sequence of packets and letters to be sent,
5. commands to start and stop sequences and open liaisons,
6. commands to get information about the experiences.

As the experiences must be repetitive and reproducible, we must have the possibility to memorize the elements in a library and to combine elements in various ways to rapidly produce new experiments. The way we propose is to give names and declare the type of elementary fields or functions. After having declared and defined the elementary elements they can be used to compose more complex elements and so on.

The command language may be formally defined as follows:

⟨command⟩ ⊬ ⟨object⟩ ⊬ ⟨parameters⟩;

The commands are elements of a predefined set of actions to be performed. A command acts on an object designated by name.
The parameters give more details on the action to be performed.
The various parameters are separated by ',' and the last one is followed by ';' .
The objects are variables designated by name.

The variable names are composed of 1 to 8 alphanumeric characters. The value of a variable may be considered as a string of bits, a positive binary number, a string of ASCII characters or a combination of these types.

The values may be described in 6 ways:

1. - '<hexadecimal digits>'
2. - '<decimal digits>'
3. - "<ascii characters>"
4. - <name of a previously defined variable>
5. - concatenation of values
6. - ORing of values.

Concatenation is expressed by writing the various values one after the other separated by at least one blank.
ORing is expressed by including the various values to the ORed between a pair of parenthesis and separating them by '/'. The values to be ORed must have the same length.

- Define Command

A value (and a length) is assigned to a variable by the DEF command:

DEF <variable name> <value>

The type of the variable must be declared before the value is assigned.

- Declare Command and Type of Variables

The type of the variable must be declared in order to allow verification of the format correctness of the parameters, its use and the value being assigned to it.

The command to declare a variable is DEC

DEC < type > <parameters>

< type> is a set of predefined symbols the value of which explains to the interpreter the way in which a value must be interpreted.

< parameters> are the names of the variables which must be attached to the list of the nominated type.

The predefined types are:

BITS variable length string of 8 bits elements.
The value is expressed by a hexadecimal string.

BINx the value is given in decimal digits and transformed in a binary representation and right justified in a field of x bytes.

CARA variable length string of printable ASCII characters.

TEXT variable length string of bytes; it is a combination of BITS, BINx, CARA or other TEXT variables.

PHED packet header; the interpreter knows the structure and the length of a packet header and verifies the correctness of the length (12 bytes).

PACK structured variable composed of

i)	a PHED variable,	header
ii)	a BIN1 variable, (optional)	text length
iii)	a TEXT variable; (optional)	text of a packet.

The value of the BIN1 variable is inserted in the value of the PHED variable in place of the "text length" field of the header.
If the value of the BIN1 variable is zero, the value of the "text length" field of the header is not changed. If the BIN1 variable is omitted, the effective length of the TEXT variable is inserted in the "text length" field of the header.
The TEXT variable may be omitted.

LTGR is structure composed of

i)	a BITS variable,	destination port
ii)	a BITS variable, only the 4 left bits	facilities are used
iii)	a TEXT variable;	text of a lettergram.

LIAI is a structure composed of

i)	a BITS variable, of 4 bytes	destination port,
ii)	a BITS variable, of 1 byte	facilities,
iii)	a BIN2 variable,	buffer length,
iv)	a BIN1 variable,	credit,
v)	a TEXT variable, (optional)	letter text to be sent,
vi)	a BITS variable or a BIN2 variable; (optional)	telegram to be sent on a liaison.

RECU is a structure composed of 3 BIN2 variables.
We will explain later the use of this structure and the meaning of the BIN2 variables.

- List of Commands

1.1 DEC is used to declare the type of the variables.
A variable must be declared before being defined and used.

1.2 DEF is used to define the value of a variable.

1.3 UPD is used to modify the value of a structure; only the variables to be updated have to be explicitly expressed.

1.4 DEL is used to delete the name of a variable or a structure from the dictionary.

1.5 GIV is used to print the contents of a variable or a structure.

2.1 SND is used to request sending of a packet or a lettergram.

2.2 OPN is used to request the opening of a liaison.

2.3 CLO is used to request closing of a liaison.

2.4 TEL is used to send a telegram on a liaison

2.5 PRX is used to request the printing of the messages received correlated to a packet or lettergram.

3.1 EXQ is used to start execution of a procedure(1).

3.2 STP is used to stop execution of a procedure(1).

4.1 CMN is used to gain access to the Control Module.

4.2 CMF is used to close access to the Control Module.
Note that the Control Module continues to work but it does not accept new commands except a valid CMN command.

5.1 END is used to request the Control Module to terminate.

(1) The concept of procedure will be explained later.

- Grammar of the Command Language

1.1 DEC ♭ <type> ♭ <variable names> ;

<variable names> : = <variable name> <variable names>,

<variable name>,

<type> : = BITS | BIN1 | BIN2 | CARA | TEXT | PHED |

PACK | LTGR | LIAI | RECU |

1.2 DEF ♭ <variable name> ♭ <values> ;

<variable name> : = <var name> | <stment name>

<values> : = <value> | <values> , <value>

<value> : = <val> | <val> <val> | (<values>/<value>)

<val> : = '<hex numb> '|' <dec numb> '|" cars "

<hex numb> : = <hex cif> | <hex numb> <hex cif>

<dec numb> : = <dec cif> | <dec numb> <dec cif>

<cars> : = <car> | <cars> <car>

<hex cif> : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D |

E | F |

<dec cif> : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

<car> : = A | B | ..... | Y | Z | 0 | 1 | ..... | 9 | . | , | ; |

+ | - | / | | * | S | NL

1.3 UPD ♭ <variable name> ♭ <values up> ;

<values up> : = <value up> | <values up> , <value up>

<value up> : = <value> | <null>

1.4 DEL ♭ <variable name> ;

1.5 GIV ♭ <variable name> ;

2.1 SND ♭ <transmission unit name> ♭ <recue def> ;

<transmission unit name> : = <packet name> | <lettergram name>

<recue def> : = <recu name> | <recue val>

<recue val> : = <first time> , <interv time> , <n time>

<first time> : = <dec numb> (i)

⟨interv time⟩ : = ⟨dec numb⟩ (ii)

⟨n time⟩ : = ⟨dec numb⟩ (iii)

(i) gives the number of seconds, the Control Module must wait after receiving the command before executing it for the first time;

(ii) gives the number of seconds, the Control Module must wait before repeating the operation;

(iii) gives the number of times, the command must be repeated.

2.2 OPN ⌀ ⟨liaison name⟩ ⌀ ⟨recue def⟩ ;

2.3 CLO ⌀ ⟨liaison name⟩ ⌀ ⟨recue def⟩ ;

2.4 TEL ⌀ ⟨liaison name⟩ ⌀ ⟨tel def⟩ , ⟨recue def⟩ ;

⟨tel def⟩ : = ⟨telegram name⟩ | ⟨null⟩ | ⟨telegram value⟩ (i)

⟨telegram value⟩ : = ⟨bin2 value⟩

⟨telegram name⟩ : = ⟨bin2 variable⟩ ⟨bits variable of 2 bytes length⟩

(i) if ⟨null⟩ the telegram field of the liaison structure is taken by default.

2.5 PRT ⌀ ⟨transmission unit name⟩ ⌀ ⟨qualifiers⟩ ;

(see on-line printing).

- The PRO Command

The PRO command is used to suspend immediate execution of the command of the second group, (SND, OPN, CLO, TEL) and third group (EXQ, STP).

It allows the definition of a procedure name representing the whole set of commands of the second group which are given after the PRO command and before the PRE command.

The PRE command terminates the definition of the procedure.
The procedure is recorded; its execution is started by the EXQ command and may be stopped by the STP command.
EXQ and STP may be used during the definition of a procedure allowing the nesting of procedure execution (not definition).

Syntax

PRO ␣ <procedure name> ;

PRE;

EXQ ␣ <procedure name> <recur ex> ;

<recur ex> : = <recur def> | <null>

STP ␣ < procedure name> ;

Notes:

1 - When the procedure is started by the EXQ command, the interpreter scans the procedure and the commands are executed in the order they were defined.

2 - A command which is illegal is treated as no operation (e.g. CLO a liaison which is not opened, TEL on a liaison which is not opened) but the cycle is maintained and repeated at the next occurrence.

- Protection

The Control Module is accessed by a terminal. To prevent illegal use of the Control Module when the operator is absent, protection must be enabled. This protection is provided by the CMN command.

CMN ␣ < password> <user name>;

This command is the first one to be given to begin a session. All other commands are simply dropped by the Control Module until it has received a correct CMN command.

CMF ␣ < password> <user name > ;

This command stops the accessibility to the Control Module. The only command accepted after a CMF command is a CMN command.
The password and user name are parameters depending on the local implementation.
Only command acceptance is stopped; a started procedure continues to run after this command.

- Termination of a Session

The END command terminates a session of the Control Module.

END < password > <user name>

This command is optional and depends on local implementation. Non mandatory facilities.

Comments:

* < car string > *

Just to put comments on the printed sheet of the terminal.

HLP; help

ACT; display the name of the working procedure.

- On-Line Printing

The PRx commands allow the request of printing of data, making them directly available during the experiments. Some parameters are always given:

1) Local time of the event as recorded by the C. M.
   NM. SS, SS minutes, seconds and hundredths of seconds.

2) Local name of the related events
   AAAAAAA name of a packet, lettergram, liaison.

3) Internal designation
   HHHH : for the packets it is the hexadecimal value of the message identifier in the header, set by the C. M.
   : for the lettergram it corresponds to the number of times the lettergram has been repeated.

4) The type of message XX

5) Values relevant for this type of message ...

Note: The 4 first fields are given an overhead of 25 characters.

- Type of Messages

Code	Meaning	Values	
S (PLT)	send	none	
DC	delivery confirmation	none	
ND	non delivery diagnostic	address of the sending code xxxx	code yyyy
TR	trace	" xxxxxxxx	time of emission
NT	network time	time	time of emission

EC	echo		
LT	letter received on liaison	length	
LG	lettergram	address of the sending code	rightmost 5 bytes
TG	telegram	telegram text	
NS	packet received	header	text

- PRINT Commands

PRT ⌀ <transmission unit name> ⌀ <qualifiers> ;

<transmission unit name> : = <packet name> | <lettergram name> | <liaison name>

<qualifiers> : = <qualifier> <qualifiers>, <qualifier><null>

<qualifier> : = S | DC | ND | TR | NT | EC | LT | LG | TG

If <qualifiers> is not given, all relevant qualifiers are implicitly set.

PRA ⌀ <qualifiers> ;

all <transmission unit name> are implicitly named.

PRM;

to print the contents of the text of the packets not related to a known <transmission unit name>.

PZT same syntax as PRT, the qualifiers are taken as negation

PZA PRA

PZM PRM

to suppress printing selectively.

PZZ; to suppress all printing.

PRZ; to restore printing as it was before PZZ.

XI. 7 Data Acquisition

All the events noticeable from the Control Module are automatically recorded on sequential file which can be processed after the experiments. To allow exchange of such data it is recommended that all centres record all the data related to traffic in the same formats or at least describe accurately these formats to allow conversion.

Moreover, it is possible that other processes are also recording statis-

tics of their activities. It would be appreciated if all these data were available for exchange.

We suggest the following basic format:

< indic> < local time > < ld > < subform > < data >

where

< indic > :< sec. cent. adr. > < port adr > (4 bytes)

< local time > : =< year > < day > < hour > < min > < sec > < mil sec >

< year > : = 2 < decimal digits > (*)
< day > : = 3 " "
< hour > : = 2 " "
< min > : = 2 " "
< sec > : = 2 " "
< mil sec > : = 3 " "
< sub-format > : = 1 byte
< ld > : = length of data binary
< data > : = contents: which is described by < subformat >

The minimum length of such a message is 14 bytes.
The messages may be blocked in fixed length records padded with "high values" (< indic > = 'FFFF').

The Control Module will use the following format type:

Type hex. val.	Meaning
01	packet sent
02	packet received
03	reserved for future use
04	" " " "
05	lettergram sent
06	lettergram received
07	lettergram ACK/NACK
08	reserved for future use
. . .	
0F	" " " "
11	liaison telegram sent
12	liaison telegram received
13	reserved for future use
14	" " " "
15	liaison letter sent
16	" " received

(*) decimal digit : = 4 bits binary value

17	reserved for future use
18	" " " "
19	liaison init sent
1A	" " received
1B	reserved for future use
1C	" " " "
1D	liaison close sent
1E	liaison close received
1F	reserved for future use
.	
.	
.	
FF	reserved for future use.

- Formats for the Control Module

Code	length	
01	12+	<txt l> < header contents> < text>
02	id.	id. id.
05	10	< dist port> <facil> < indic > <letter length >
06	10	< dist port> < facil> < indic> <letter length>
07	10	< dist port> < facil> < indic> <conditions>
11	10	< dist port > <local port> < telg text>
12	10	< dist port > < local port> < telg text >
15	12	< dist port > < local port> < indic> <letter length>
16	12	< dist port > < local port> < indic> < letter length >
19	14	< dist port > < local port> < facil> < buf length >
17	14	< dist port> < local port> < facil> < buf length > < credit>
1D	26	< dist port > < local port> <reason < credit > < number of letter sent> < total volume sent > < number of letter receiv.> < total vol. received> < number of telgs sent > < number of telgs rec.>
1E	10	< dist port> < local port> < reason>

< facil>, < indic>, <letter length>, < telg text>, <credit>,
< reason>, < total volume sent>, <number of ...> 2 bytes
< distant port>, <local port> 4 bytes

< indic>: = progressive binary number.

XII. BASIC TEST SERVICES - ISPRA PROPOSAL

These facilities should be implemented in each centre as subscribers to the local Transport Station. They are the minimum set of facilities necessary to enable the proper integration of the various transport stations existing in the network, and can subsequently be used for making measurements on the performance of the network as a whole.

Both the lettergram and liaison facilities of the Transport Stations will be used, though not necessarily by the same subscriber process.

XII. 1 Lettergram Mode

It is recommended that a port operating in lettergram mode initially sends at least 4 "Receive Lettergram" commands to his T.S., thus allowing each centre to use his service. The statistics which should be kept are:

- distant port,
- text length,
- time of receipt.

The buffer length offered should not be less than 1K characters.

XII. 2 Liaison Mode

It is recommended that a port operating in liaison mode will accept between 4 and 10 liaisons. A credit of 3 should be given on each opened liaison.
The statistics which should be kept are:

- distant port,
- traffic volume,
- time of open,
- time of close.

A maximum letter length of 4K characters is recommended.

XII. 3 List of Ports

- DROP: (PORT NUMBER: 0001)

  This port receives in lettergram mode and in liaison mode; it does not emit and destroys all letters and telegrams received. It closes automatically a liaison on a time-out of about 1 min if no new letter is received.

- ECHO: (PORT NUMBER: 0002)

  This port receives in lettergram mode and in liaison mode; it sends

back all the letters or telegrams it receives. It closes automatically a liaison on a time-out of 1 min. if no new letter or telegram is received.

- REBOUND: (PORT NUMBER: 0003)

This port receives only in lettergram mode; it uses the 40 last bits of the letter received as an address to resend the letter (after having shortened it by 40 bits). It uses the same facility as those used to send him the letter. If error control is requested and transmission error is detected it uses the first 40 bits of the letter as an alternate address.

- DEBUG-COMMUNICATION: (PORT NUMBER: 0004)

This port receives only in lettergram mode. It prints on a local printer, the full address of the sending port and the contents of the letter; it sends back to the sender the message "ACK" when the message has been printed successfully, and "NACK" if the message cannot be printed correctly.

- EMIT: (PORT NUMBER: 0005)

This port works exclusively in liaison mode. When a liaison is established with this port, it emits letters of the length specified in the LI-INIT. The text of the letter is composed of the repetition of a field of 4 characters which are the sequence number of the letter (in ASCII code).
It sends a maximum of 9999 letters and then terminates automatically the liaison.

This port receives only telegrams. A telegram text of FF00 (Hex) stops smoothly the transmission (when the proposed letters have all been transmitted).

A telegram text of FFFF (Hex) stops abruptly the transmission (forces immediate closing of liaison) at the initiative of EMIT. All other telegram text values limit the number of letters which will be transmitted, the value is interpreted as a positive binary value.

If received in time, the termination will be smooth; if too late, the termination will be abrupt.
Values different from FF00, FFFF and greater than 270F (Hex) = 9999 (Dec) are considered error and cause abrupt termination.

ACKNOWLEDGEMENTS

Most of the ideas presented in this paper reflect the results of the discussions held within the EIN-CCG meetings. The authors gratefully acknowledge the contributions of:

M. Gien, H. Zimmermann, L. Pouzin of l'Institut de Recherche d'Informatique et d'Automatique, Paris;

A. Duenki, P. Schicker, W. Baechi of the Eidgenossische Technische Hochschule, Zurich;

J. Laws, R. Scantlebury of the National Physical Laboratory, Teddington, United Kingdom;

A. Le Moli, E. Repossi of the Centro Rete Europeo d'Informatica, Milan;

D. Barber, M. Bolognani, M. Deparis of EIN-Executive Body, London.

Grateful thanks are also addressed to G. Nocera for the assistance in installing and maintaining the equipment.
The present report was carefully typed by Mrs. A. Dorpema.

REFERENCES

- EIN Documentation; D. L. A. Barber, EIN-Executive Body, NPL, Teddington, TW11 OLW, United Kingdom
- IFIP INWG Documentation; D. L. A. Barber, EIN-Executive Body, NPL, Teddington, TW11 OLW, United Kingdom
- European User's Workshop on Protocols; R. Scantlebury, NPL, Teddington, TW11 OLW, United Kingdom,
- The 31st International Conference on Computer Communication, August 1976, Toronto, Canada.

European Communities — Commission

The connection to EIN packet switched network

by S. Amic, F. Sorel, A. Termanini (Dept. B — Electronics Division) and W. Boettcher, A. Endrizzi, P. Moinil, J. Pire, K. Weaving (Dept. A. — Informatics Division)

Joint Research Centre, Ispra Establishment, Italy

Luxembourg : Office for Official Publications of the European Communities

1977 — 71 pages — 21.0 x 29.7 cm

EUR 5718, 'Information Management' series

EN

Catalogue number : CD-NU-77-009-EN-C

FB	310	DKr	50,80	DM	20,15	FF	41,60
Lit	7 300	Fl	21,10	£	5	US	$ 8.50

This report presents the design and the implementation of the JRC-Ispra connection to the European Informatics Network (COST 11 Project).

Problems raised by the physical and logical links, necessary to connect a large data-processing installation to a network are analysed.

Implementation details are well separated from more general topics, so that the document can be considered as a valid contribution to the present discussion on computer networks design.

SALES OFFICES

BELGIQUE — BELGIË

Moniteur belge — Belgisch Staatsblad

Rue de Louvain 40-42 — Leuvenseweg 40-42
1000 Bruxelles — 1000 Brussel — Tél. 512 00 26
CCP 000-2005502-27 — Postrekening 000-2005502-27

Sous-dépôts — Agentschappen :

Librairie européenne — Europese Boekhandel
Rue de la Loi 244 — Wetstraat 244
1040 Bruxelles — 1040 Brussel

CREDOC
Rue de la Montagne 34 — Bte. 11
Bergstraat 34 — Bus 11
1000 Bruxelles — 1000 Brussel

DANMARK

J.H. Schultz — Boghandel

Møntergade 19
1116 København K — Tel. 14 11 95
Girokonto 1195

BR DEUTSCHLAND

Verlag Bundesanzeiger

Breite Straße
Postfach 10 80 06
5000 Köln 1 — Tel. (0221) 21 03 48
(Fernschreiber : Anzeiger Bonn 08 882 595)
Postscheckkonto 834 00 Köln

FRANCE

Service de vente en France des publications des Communautés européennes — Journal officiel

26, rue Desaix — 75 732 Paris - Cedex 15
Tél. (1) 578 61 39 — CCP Paris 23-96

IRELAND

Government Publications

Sales Office
G.P.O. Arcade
Dublin

or by post from

Stationery Office

Beggar's Bush — Dublin 4
Tel. 68 84 33

ITALIA

Libreria dello Stato

Piazza G. Verdi 10
00 198 Roma — Tel. (6) 85 08
CCP 1/2640 Telex 62008

Agenzia :

Via XX Settembre
(Palazzo Ministero del tesoro)
00 187 Roma

GRAND-DUCHÉ DE LUXEMBOURG

Office des publications officielles des Communautés européennes

5, rue du Commerce
Boîte postale 1003 — Luxembourg
Tél. 49 00 81 — CCP 19 190 81
Compte courant bancaire : BIL 8-109/6003/300

NEDERLAND

Staatsdrukkerij- en uitgeverijbedrijf

Christoffel Plantijnstraat, 's-Gravenhage
Tel. (070) 81 45 11 — Postgiro 42 53 00

UNITED KINGDOM

H.M. Stationery Office

P.O. Box 569
London SE1 9NH — Tel. 01 - 928 6977 ext. 365
National Giro Account 582-1002

UNITED STATES OF AMERICA

European Community Information Service

2100 M Street, N.W. — Suite 707
Washington, D.C. 20 037 — Tel. (202) 872 8350

SCHWEIZ—SUISSE—SVIZZERA

Librairie Payot

6, rue Grenus — 1211 Genève
CCP 12-236 Genève - Tél. 31 89 50

SVERIGE

Librairie C.E. Fritze

2, Fredsgatan — Stockholm 16
Postgiro 193, Bankgiro 73/4015

ESPAÑA

Libreria Mundi-Prensa

Castelló 37 — Madrid 1
Tel. 275 46 55

OTHER COUNTRIES

Office for Official publications of the European Communities

5, rue du Commerce
Boîte postale 1003 — Luxembourg
Tél. 49 00 81 — CCP 19 190 81
Compte courant bancaire : BIL 8-109/6003/300

NOTICE TO THE READER

All scientific and technical reports published by the Commission of the European Communities are announced in the monthly periodical '**euro - abstracts**'. For subscription (1 year : FB 1 200) please write to the address below.

FB 310 DKr 50,80 DM 20,15 FF 41,60 Lit 7 300 Fl 21,10 £ 5 US $ 8.50

OFFICE FOR OFFICIAL PUBLICATIONS
OF THE EUROPEAN COMMUNITIES
Boîte postale 1003 - Luxembourg

Catalogue number : CD-NU-77-009-EN-C