sparekassernes
datacenter

august 1983

TOWARDS

A SYSTEM ARCHITECTURE

Project report of the

EF-57

industrial research education program

Tue Bertelsen

TABLE OF CONTENTS

Towards a system architecture

Final report of the
EF-57
industrial research education program

This report is part of the fulfillment of the conditions for completion of the industrial research education program denoted 'EF-57 Microprocessor'.

The EF-57 project has been sponsored by the Danish Academy of Technical Sciences and Sparekassernes Datacenter.

Tue Bertelsen
Sparekassernes Datacenter

August 19, 1983

This page intentionally left blank

THE EF-57 INDUSTRIAL RESEARCH EDUCATION PROGRAM

1.0 Acknowledgements

The EF-57 project has been supervised by a supervisory group, consisting of

Per Gert Jensen	Department of Computer Science Technical University of Denmark
Jan Bech	Sparekassernes Datacenter
Birger Nielsen	Sparekassernes Datacenter

I will express my thanks to these three people, as they have continuously contributed to the project with many good advises and discussions.

Likewise I will thank the employees at the Department of Computer Science, who participated in the discussions around the TUBUS design.

Furthermore will I thank my colleagues in the department for decentralized equipment within SDC, TEKII, where I am employed, and especially the Microlab group, which have contributed with good discussions and criticism on the project, as it evolved.

Also the different vendors, with whom I have had contact, should be acknowledged, as they have provided the different contacts around the world. Especially Olivetti should be ack-

nowledged for providing me access to their Advanced Technology Center in Cupertino, which I used as the basis for many of the activities and visits, while I was in Silicon Valley.

Finally I will acknowledge Sparekassernes Datacenter, the Department of Computer Science and ATV for giving me the possibility to carry this project through to its final stage.

1.1 Synopsis

The EF-57 project has been concentrated around defining a model for a new system architecture of the branch office computer systems in the Danish savings banks, which eventually could replace the systems currently in delivery. At the same time, the study has provided myself with a useful know-how within microprocessor and microcomputer technologies, and this know-how has been the basis of the actual architectural proposal.

A system architecture model has been devised as a part of the project. My method has been to design a model, which may take benefit of the on-going developments and standardization efforts in the microcomputer market, given the assumption that this market will dominate the type of systems, which are being designed for business usage.

As a result, the proposed architecture will allow a step-wise migration from the current branch office system to a new system following the concepts within the architecture. Finally, the architecture will be prepared for change, if technology or products make this necessary.

The practical case of the EF-57 project has been the design of a microcomputer system, able to operate as a gateway between systems designed for the Danish public data network and the old loop line data communications network between the savings banks and SDC.

Other practical issues have been the design of a microcomputer backplane bus and the participation in the IEEE P896 'future bus' standards committee.

1.2 Conclusions

The goals and objectives of the EF-57 industrial research education program have been:

a) **to design and implement the architecture of a savings banks branch office terminal system based on state-of-the-art microprocessor technology which can act as a model for future evolution of the terminal system currently in delivery.**

b) **to document and communicate leading edge know-how within the fields of microprocessors and VLSI-systems, including both hardware and software aspects.**

c) **to indicate the direction and activities necessary for SDC to impact its suppliers of data processing equipment to migrate towards the concept of a system architecture developed during the Industrial Research Education program.**

Due to an interruption of the project in 1981, the practical implementation case has been changed to include the McTan X.21 network interface project, described in chapter 10.

The rest of the project has concentrated on providing the background for a new system architecture for the decentralized branch office systems in the Danish savings banks. This has been done by introducing the concepts for a system architecture, called ASA, which takes into account the trends within the microelectronics area and provides a leveled architecture, that may be step-wise introduced. ASA is primarily a architec-

ture addressing the technological problems of systems design.

This architecture will allow the migration to a new type of branch office systems, that are prepared for change.

ASA is a new type of system architecture, that through its levels will accomodate different types of system configuration, while keeping a uniform interface at the appropriate points. ASA also rely on industry standards for system interconnection, which means that future systems within the savings banks can be designed using individual components a multi-vendor environment.

As such, ASA is a model concept, that is able to follow the progress in technology during the years to come.

ASA is hence a fulfillment of objective a) stated above.

Objective b) has been covered by the practical case of this project and by the activities on the design of the TUBUS microcomputer backplane bus and the participation in the IEEE P896 future bus standard committee.

Objective c) is fulfilled by the concepts introduced in the ASA architectural model, as well as the project proposals made in chapter 9 of this report.

As such, the results in this report will form the technologically basis for future migration activities within SDC.

1.3 Summary

The rest of this report is organized as follows:

Chapter II describes the EF-57 project background and the goals and motives of the project. The chapter also contains a description of the old TP1 decentralized terminal system in the savings banks branch offices and the new TP2 system currently under installation, as well as it indicates the problems inherent in the new TP2 system.

Chapter III is focused on the evolution of microelectronics technology, which is impacting the current and future state of computer design. The conclusion of this chapter is, that through the use of cheap microprocessors, several processors will exist within any system or system component in the future branch systems.

Chapter IV discusses the general concepts involved in linking together several processors, as well as it contains a brief overview of the current market of microprocessor based computers.

Chapter V concentrates on the methods existing for interconnecting processors in a loosely coupled fashion, either through nation-wide data communication networks or through geographically restricted local area networks.

Chapter VI describes the interconnecting of processors within the nodes. Interconnection is performed through computer backplane buses, and the chapter describes the current standards in this area. Also, it discuss the design of SDC's own backplane bus TUBUS, which was designed during the first part of the EF-57 project. Finally, it describes the work within the IEEE P896 future bus committee, where I participated as a part of this

project.

Chapter VII relates to software issues, and the interrelation ship between the hardware modules and the system software. It discusses the trends within standardization of operating systems for microcomputers, and arrives to the concept of a message bus, which must be implemented to ease the communication between application software modules in future systems based on multiple processors.

Chapter VIII summarizes the preceding three chapters into a system architecture proposal, called ASA. ASA is a leveled architecture, which specifies the concepts for four levels, that cover the branch office systems:

- the ASA nation-wide level
- the ASA branch office level
- the ASA user node level
- the ASA system software level

Chapter IX presents some technical steps, that SDC may take to migrate from the current TP2 system to the new ASA proposal. It discusses two potential project: one is the design of a network communications board in accordance with the ASA user node level. The other is the design of a new high-performance processor board for the terminal computers in TP2, which eventually may migrate to the ASA branch office level.

Chapter X describes the practical part of the EF-57 project. This has been the design of a microcomputer system, that has allowed the TP2 Olivetti systems to connect to the old loop line network in the savings banks by simulation a DCE from the Danish public data network together with front-end functions within the microcomputer.

Included in the back of the report are a bibliography list of references and other sources of information, which has been the basis of the project. Also a subject index is included.

1.4 Project History

The EF-57 project was started in November, 1979, with the purpose to design a multiprocessor system, based on the then new 16-bit microprocessor technology, which should give Sparekassernes Datacenter (SDC) know-how on this technology. During 1980, the goals and objectives of the project was changed to be oriented towards using this know-how to define the architecture of a new branch office system for the Danish savings banks. This of course changed the way, the work was carried out, and changed the final result to a product, which would bring more value to SDC. However, this change in goals caused the emphasis to be placed on a larger subject, where the original goal only was a minor part.

As such, there are so many subjects and factors, that are important for defining a system architecture, that it would normally not be assigned to only one person to carry through. Hence, the project mainly concentrates on the technologically issues in architectural design, leaving some areas uncovered.

In October and November 1980, I spend two months visiting major microprocessor and computer companies in the United States and attended conferences at the same time.

In the summer 1981, the project was interrupted due to the practical X.21 project described in chapter 10. This project went on until the end of 1982. As the EF-57 project then was restarted, it was decided to use the X.21 project as the practical case of this thesis and instead keep the architectural part of the EF-57 project on a theoretical basis without any practical implementations.

I have participated in several small projects within SDC during the period and must say, that it sometimes is difficult to completely concentrate on the research program, when urgent pro-

blems arrive. However, these are the conditions of working in an industrial enterprise, and it brings the needed practical experience along.

As part of the study, I have had contact with various companies and institutions abroad, to get a feeling of what was happening on the microcomputer market.

Educational activities has mainly been concentrated on the ESF-50 course, a 30 days course, which concentrates on the relationships between industry, businesses and society. Likewise, I have participated in various conferences on electronics and communication. A presentation of some of the ideas behind our view on use of the new technology was given at the NORDDATA'81 conference, and a presentation of the practical project was given at the Programmering'82 conference in Copenhagen.

1.5 Dansk summary

Denne rapport beskriver en model for system arkitekturen for sparekassernes fremtidige terminal systemer.

Kapitel II beskriver projektets baggrund og målsætningen for EF-57 projektet. Kapitlet indeholder også en beskrivelse af det tidligere TP1 decentraliserede terminal system i sparekassekontorerne og det nye TP2 system som i øjeblikket er under installation. Det beskriver også nogle af de problemer, som eksisterer i TP2.

Kapitel III omhandler udviklingen indenfor mikroelektronikken, med speciel vægt lagt på microprocessor udviklingen. Konklusionen er, at på grund af mikroernes billige pris, vil fremtidige systemer være baseret på anvendelse af multiple mikroer adskillige steder i systemet.

Kapitel IV diskuterer de generelle begreber ved sammenkobling af flere processorer, ligesom det indeholder en kort beskrivelse af mikrodatamat markedet.

Kapitel V omhandler data kommunikationsnet, både det landsdækkende offentlige datanet som SDC benytter sig af, samt lokale datanet.

Kapitel VI omhandler, hvorledes processorer kobles sammen internt i datamatsystemer ved hjælp af computer busser. Det beskriver de gældende standarder på dette område, samt min egen udvikling af TUBUS. Desuden beskrives min deltagelse i IEEE P896 standard kommitteen, hvor der er gjort bestræbelser på at definere en fremtidig 32-bits bus standard.

Kapitel VII omhandler programmel siden, hvor der lægges vægt på anvendelsen af standardiserede operativ systemer på mikrodatamat markedet. Der beskrives en logisk besked-udvekslings bus, som vil være nødvendig når applikationer skal kommunikere sammen.

Kapitel VIII summerer de tre tidligere kapitler i et forslag til en model for en system arkitektur, der kaldes ASA. ASA er en niveau-delt arkitektur med fire lag, som dækker de decentrale sparekassesystemer:

- ASA landdækkende lag
- ASA kontor lag
- ASA internt system lag
- ASA system software lag

Kapitel IX beskriver de tiltag, som SDC kan gøre i første omgang for at komme fra TP2 til den nye ASA struktur. Der fremføres to projekt forslag: et der omhandler udvikling af et standard kredsløbskort, der indeholder SDC's kommunikationsstruktur, og et, der omhandler udvikling af et nyt ydedygtigt datamat-kort til den nuværende terminal datamat i TP2.

Kapitel X beskriver den praktiske del af projektet, hvor der er blevet udviklet en mikrodatamat, der på den ene side simulerer en DCE på det offentlige datanet samt SDC's front-end og på den anden side har grænsesnit til SDC's gamle ringnet. Projektet har gjort det muligt at tilkoble TP2 systemerne til de gamle ringnet, da leveringen af SDC's nye front-end anlæg blev forsinket.

Bagerst i rapporten findes en bibliografisk fortegnelse over kilde- og studie-materiale til projektet, samt et index-register.

CHAPTER TWO

PROJECT BACKGROUND

2.0 Introduction

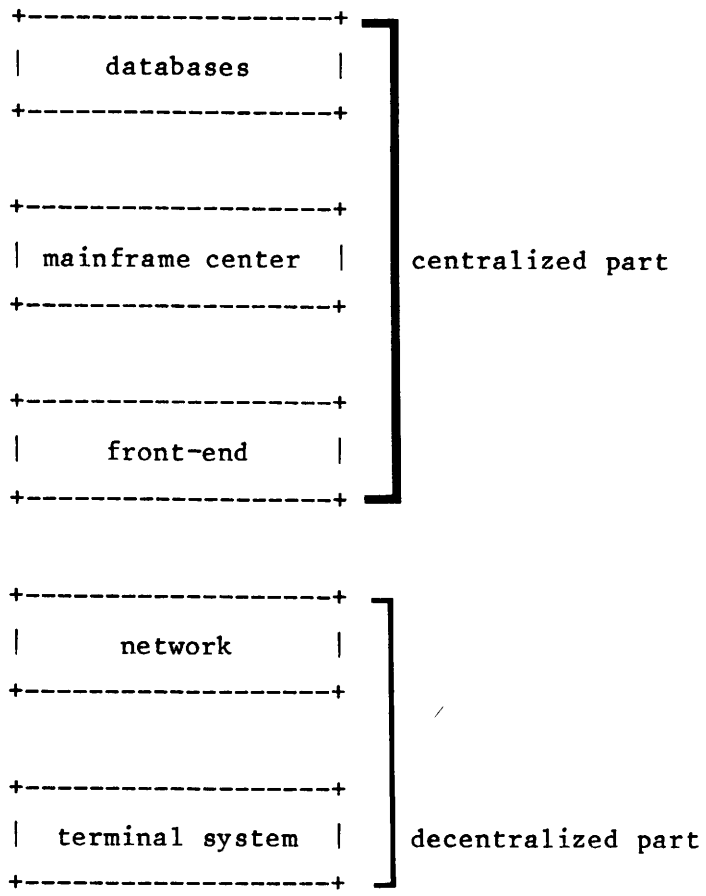
Sparekassernes Datacentraler (SDC) was established in 1963 as an independent organization by the Danish Savings Banks Association. Its purpose is to promote the most efficient use of information processing in the Danish savings banks, primarily by developing and operating automatic data processing systems.

In the late 1960'ies, the importance of collecting data at the origin, rather than through some intermediate way like punched cards or tapes, was realized by the Danish Savings Banks. At that point in time, SDC initiated the first terminal project, which resulted in the installation of Datasaab Minicomputers and teller terminals in the branch offices of the savings banks during the first half decade of the seventies.

This terminal system was the first step towards todays highly automated banking environment, based on computer systems placed locally in each branch office with a data transmission network to a central common data processing facility, shared by all the savings banks, for transaction processing and database administration.

As a result of the concentration of the common data processing services into one center in the seventies, SDC's name changed to **Sparekassernes Datacenter**, keeping the same abbreviation. Although being a datacenter, SDC's system responsibility covers

not only the part within the center, but also the decentralized part. In fact, the entire savings bank system supported by SDC can be segmented into five parts:



The first terminal system, which is still running today, is normally called Terminal Project number 1, or TP1. Strictly speaking, TP1 in this report only refers to the actual terminal project, although the normal use in SDC of the concept TP1 includes both the terminal system, the network, the front-end and in some cases the corresponding application programs on the central mainframes.

2.1 TP2

The seventies unveiled the rapid growth in data processing technology and an increasingly use of and dependency on the installed TP1 banking system. New requirements came along together with new activities in the savings banks sector, some of these caused by the new Danish bank act in 1975, which almost equalled the savings banks with the commercial banks. Nevertheless, the technological elements of the data processing system in the savings banks branch offices remained unchanged throughout the seventies. All application development needed to support new requirements from the savings banks were done centrally on the central mainframes.

The missing ability of the TP1 system to respond to the change caused SDC to initiate the Terminal Project 2, or TP2, in 1977. TP2 should be a new terminal system to replace TP1 and 11 suppliers of data processing equipment were invited to make proposals for the new system. In 1978, five of these suppliers were selected for further evaluation. This eventually resulted in the selection of Olivetti as the supplier of the new equipment in 1979.

Whereas TP1 supported data collection in the branch offices, TP2 also supports data processing in the branch offices and offers the users an extensive set of banking-oriented functions, radically different from those in the TP1 system. The TP2 system has been one of the largest orders for data processing equipment ever made by a banking community, with a system cost around 500 million dKr.

TP2 introduces the concept of decentralized applications development for the savings banks, i.e. applications software, that runs directly on the computers within the branch offices. One reason for doing this is that TP2 uses screen terminals and

user dialog programs, which were not present in TP1. This type of processing can be carried out locally, thus restricting the centralized part of the system to transaction processing and database handling.

It is important to separate the functionality of the TP2 system from the technology in the system. Functionality is concerned with the logical functions, that the user experiences in contact with the system, i.e. the different types of banking-oriented applications implemented in the system. Technology has to do with the physical design of the system, i.e. the hardware components around which the functionalities are built.

TP2 was supposed to provide the technological foundation for all new functionalities in the savings banks throughout this decade. However, the technology avalanche has made the TP2 system technologically outdated even before it is installed, and the increasing complexity and requirements of applications now cause the performance limits of the TP2 system to be a significant problem. This report will show how and why.

That is why, SDC now needs to establish a long-term target of a future terminal system, let it be called TP3, before even TP2 is fully installed.

Several reasons cause this approach to be necessary:

- a major replacement, as with TP2, will no longer be economically feasible. TP3 must migrate from TP2 or live together with TP2 over an extended period of time.

- performance problems are sneaking in on TP2, and TP2 is not prepared for easy upgrade with new standard products.

- TP2 introduces entirely new concepts and application areas totally different from the old transaction/inquiry oriented TP1. The high development costs involved herein must be justified beyond the TP2 life cycle.

- the impact from technology is significant increasing, and TP2 cannot easily use progress in technology. This makes it hard for SDC to remain competitive as the sole provider of data processing systems to the savings banks in relation to outside vendors, who might offer more cost/effective solutions to the savings banks, but not compatible with the TP2 concept. Also, if the technology within the system cannot keep pace with the new requirements of functionalities, the savings banks may not retain the competitive edge in relation to the commercial banks, who will introduce new branch office systems during the next decade, based on a more sophisticated technology.

- finally, but maybe most important, the long-term goals for data processing use in the savings banks must be stated to define the medium-term actions, SDC must perform in the years to come. TP2 was conceived and designed within a few years time span, necessitating a short organizational process to convert the development activities within SDC to the target TP2 system without amble room for reviewing, testing or prototyping new ideas.

The target of the EF-57 industrial research education program is to address some of these reasons and to suggest how SDC may migrate to a new, future terminal system, TP3, by taking into account the continuing developments in technology.

2.2 EF-57 Project Goal

The goals and objectives of the project as stated in the final EF-57 educational program was summarized to the following:

a) **to design and implement the architecture of a savings banks branch office terminal system based on state-of-the-art microprocessor technology which can act as a model for future evolution of the terminal system currently in delivery.**

b) **to document and communicate leading edge know-how within the fields of microprocessors and VLSI-systems, including both hardware and software aspects.**

c) **to indicate the direction and activities necessary for SDC to impact its suppliers of data processing equipment to migrate towards the concept of a systems architecture developed during the Industrial Research Education program.**

Note, that the first objective involves not only a design of what can be made today, but actually a forecast of the data processing senario in five to ten years. Also, it includes the utilization of off-the-shelf technology, in stead of dedicated hardware design. The main purpose is to integrate available

building blocks into a product, which serves the banking requirements of the savings banks. Finally, the product need not to be a revolution in systems design, necessitating complete abandonment of the existing system, but a evolutionary concept allowing the easy and economic migration from what we have now.

Also, the project is technology oriented rather than functionality oriented, and concentrates on providing the technology background needed to specify and evaluate future products, on which the functionalities have to be implemented.

Second objective is the statement of the educational responsibility of the project in acquiring technological know-how, and to involve other people in SDC in the assimilation of this knowledge to be used in current design. As such, this report constitutes a partial fulfillment of that objective.

Finally, the project is not self-contained, but a step in the large task to impact our suppliers to consider SDC as a qualified design partner in future systems design. This means, that the systems architecture derived in this report may not be the eventual requirement of SDC, but our proposal to the suppliers - a fact, which is revolutionary enough, as most users contend themselves with a manufacturer's catalog products, which in some cases effectively ties them to a manufacturer.

It should be noted, that due to the interruption of the EF-57 project in 1981, the original practical case of the project - the actual implementation of a new architecture as stated in objective a) above - was completely dropped and subsequent replaced by the McTan X.21 project (see chapter 10). As such, the major part of this report will contain only a so to say theoretical or conceptual design of the new branch office system of the future.

2.3 Long Term Goals and Motives.

EF-57 main goal is to define the technology view of the next generation of terminal systems in the Danish Savings Banks. Throughout the report, this concept of the new system will be named Terminal Project #3, or TP3 for short.

Primarily, EF-57 is an education in industrial research, development and management. Severals parts have interest in the project of various reasons, and it might be fruitful to shortly analyze the motives of these parties to understand why and how this project has been carried out.

ATV: The motives of ATV's participation in the management of the industrial research education program are to provide the Danish Industry with qualified people able to manage R&D divisions and secondly providing an opportunity for companies to enter a promising or untraditional R&D project with less economically risk, while at the same time obtaining a goal-oriented education of an employee.

SDC: SDC has many motives to support this project. As stated previously, the primary reasons of getting a clear view of the next generation terminal system and of solving the current performance problems are obvious.

Secondly, SDC has engaged itself much more aggressively in the cooperation with the new supplier, Olivetti, and required a joint-venture agreement on future banking modules as part of the TP2 project; a joint-venture agreement, which eventually should descent from the management level of informative discussion to the creative level of active codesign and mutual exchange

and utilization of information and ideas. Thirdly, SDC has always tried to acquire and use state-of-the-art know-how within the areas of data processing concerning banking applications, and the almost exponential increasing impact from technology, especially from the area of microelectronics, has necessitated the need for qualified people on these subjects within SDC.

Nevertheless, steps to support this need for know-how have been slow, although the microprocessor development group, initiated by Anders Nielsen and me in 1978, was the first radical move in SDC to get in-house applicable know-how on microelectronics. Still the concept of a data service and software company deeply involved in hardware design lacks acceptance to cost justification at some parties within and outside SDC, making the EF-57 project a means to justify both the investments already made and also further research in this area, as the true benefits and impact are now beginning to show up. Also this project should indicate the need for having a qualified staff of people within SDC, capable of following and evaluating hardware design together with technology progress in the market area of interest for the savings banks.

ID: The motives of the Department of Computer Science at the Technical University of Denmark, who is the third official part in the project, are different from those of SDC. The Institute, with its background in pure research and educational responsibilities, tends to consider the project as a normal licentiate project, much more theoretically oriented. Hopefully, the Institute expects to see some results from the project, which can be primarily used in the education of M.Sc.'s in data science, and secondly can be used to qualify the name of the Institute as a source of advanced

degrees in data processing. As a sideeffect of the project, the Institute should benefit from the industry contact, by experiencing the turbulent way projects are carried out within an industrial enterprise, and by seeing how much or how little of the theoretical work is actually used and considered practically important by industry.

My Own: My own motives, more or less matured through the period of the project, sometimes correspond and in other cases conflict with those stated above. My prime motive is to show, that an organization like SDC must employ a broad and market-oriented know-how on hardware and computer technology when implementing large systems like TP2.

My view of SDC is, that it is a company, who primarily buys hardware and systems software from outside vendors. If standard offerings do not meet our requirements, SDC may eventually enter a cooperate development with vendors as a source of ideas, constraints, inspiration and evaluation. SDC expects the vendors to have the needed high-level, detailed expertise in specific areas, whereas the general overview of what the users need and the high-level knowledge of future requirements and applications is the responsibility of SDC. Thus, my intention is not to do my own private system design, but to devise a system concept, that eventually will be supported and produced by the market. This however requires, that we know where the market is going and when.

Hence, another of my motives is to provide SDC (and myself) with a level of know-how in microelectronics, enabling SDC to participate in the discussing of computer design with the vendors. Besides the work done within the microlab group of SDC (see chapter 10), SDC

has never before been involved in the microelectronics field and has no know-how within this area. As microelectronics will be the basic hardware and firmware building blocks of future banking systems, such a know-how must be acquired other places in SDC.

Also, despite the fact that many Danish companies have expressed a troublesome view on the ability to compete in computer manufacturing, I will try to show, that a Danish industrial enterprise (if SDC can be characterized as such) in a few years period can attain a level of design know-how starting basically from scratch, which is recognised as important by other parties in the microelectronics field, and which can be used as a tool for clever utilization of the technology instead of resignation.

Finally, I must stress that I consider this project to be an education of my personal abilities to go into a complete new field of concepts, to apply these concepts together with previous experience in SDC to forecast the impact on SDC and savings banks, being able to cope with the steady change in technology and applications in a manageable way, rather than knowing the answer at the beginning of the project, and then just maturing details underways.

As such, I do not consider the project to be an exercise in theoretical equilibrism. Likewise, I do not consider it important to spend two years going into details of a very specific remote corner of data processing technology.

It is important to recognize, that all the fields covered in this report has been primarily new and unknown to me. As such, I do not pretend, that this report will uncover all the answers for SDC in the years to come, and from that background, the following confinements have been made.

2.4 Confinements:

A full TP3 concept involves many factors, similar to TP2. They include the kind of modules needed in a branch office system, the applications which will be performed, the access of information, ergonomic issues, the type of software development and eventually the cost of the system.

My basic confinement of the project has been to concentrate on the technological rather than the functional aspects of the branch office systems design, both because this is the area, where my own prime interest lies and it has the closest relations to the initial preliminary objectives of the project, and because this is the area, where SDC currently has its largest weakness in know-how and evaluation capabilities.

To confine the technological part of this project, I have selected some important issues which I consider to play the key role in TP3 design. Likewise, I have made default assumptions on issues, which are not covered by EF-57, but affecting TP3 anyhow.

The basic view, that I will advocate in this report, is:

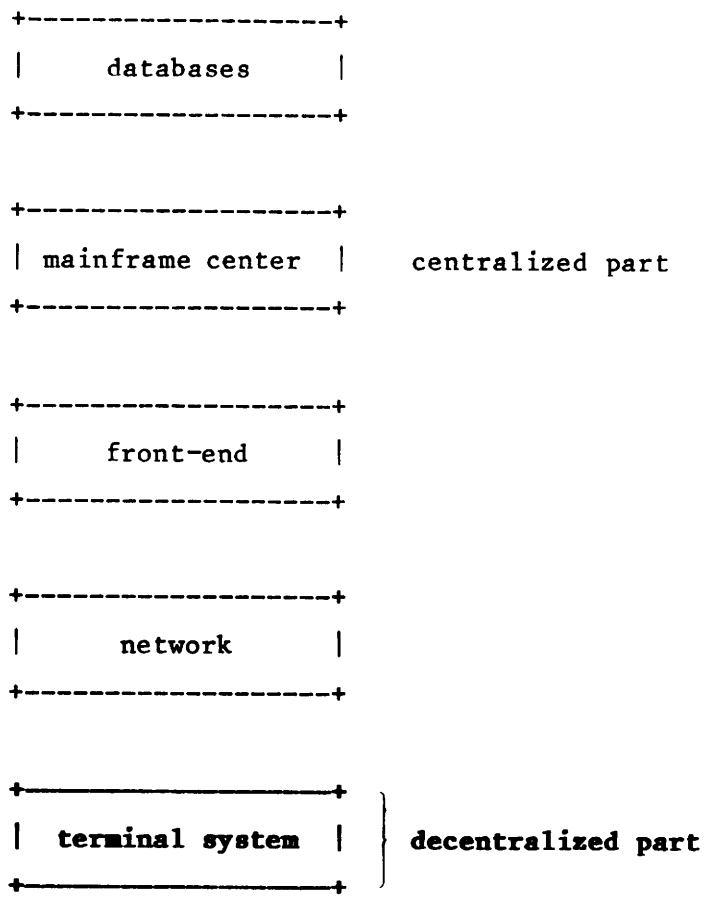
TP3 must be designed for change!

I consider it impossible for anyone to exactly describe the type of applications or systems, that the savings banks will need ten years from now. Needs will change, technology will change, people will change, SDC will change. If SDC cannot respond fast to changes, SDC will loose control of the utilization of data processing services within the savings banks, and then SDC may loose its current justification.

This concept of adaptation to change is also the reason, why certain fields are not covered down to specific details. I have tried to cover the basic areas, where technology change will occur. All over the world, people and companies are trying to solve problems within these areas, and my intention with this project is not to supersede that work, but to take advantage from it.

As such, the general approach in my work will be from the high level and downwards, not the opposite way. The opposite way is covered in the practical part of the EF-57 project, where a concrete design is done, based on the new technology.

The technological elements of the entire system architecture of SDC, that I will cover, are primarily the decentralized part and partly the network:



Within that confinement, the following items determines the success of a future TP3 system:

1) How the decentralized parts of the entire savings banks systems are connected with the central processing services of SDC.

2) the general system architecture of the entire computer system installed within a branch office.

Because: this is the basic view, the savings banks have of the system, and it defines the possibilities of changing the characteristics of the system seen from the user level, i.e. number of workstations, disk units, printers, word processing units etc. Also, it defines the framework for any kind of applications development and enhancement. Finally, one must expect the logical parts, which constitute the general system architecture, to be identifiable as replaceable units, which can be exchanged as time goes on, and hence must possess a certain degree of compatibility between different vendors.

3) the internal node architecture of units within the systems architecture.

Because: This is the area, that determines how a single node is constructed from smaller electronic modules, like circuit boards. For some type of systems, the internal node architecture may not be applicable, but for more complex system nodes, like those found in TP2, this area will determine, whether SDC becomes technologically locked to a vendor. Also, this is an area, where microelectronics technology will have a most profound impact.

4) System software architecture, which links together the different parts of the entire information processing concept in the savings banks.

Because: the general system architecture and node architecture will impact the method of software design in a way probably different from TP2, and since the several types of processor modules will be located different places in the system, an unified approach must be established to guarantee, that application software modules will work together. Another factor of the system software architecture will be to enable the use of standard software offerings on the market, which eventually may necessitate SDC to change its policy of acquiring software because of this.

2.5 Default assumptions

1) the Danish public data network will be the nationwide carrier of digital information exchange between branch offices and banking centers throughout TP2 and TP3 life cycle. Thus, I will not discuss e.g. the use of satellite communication nor new proposed projects as the Danish broadband network. I will assume, that SDC will always use the available public data network technology, but this will require, that the branch systems can adapt to changes in that technology.

2) the need for applications development will be considered to be independent of the TP3 system architecture. The needs for various applications are originated at and defined by the users (i.e. the savings banks), and it is the responsibility of SDC to provide a processing machinery where these applications can be implemented.

3) the functional characteristics of peripheral modules within the system will be considered independent of TP3. By this, I mean that the design of a book printing device or a badge card reader is not the goal of TP3. TP3 should be able to handle any peripheral devices, which can be conceived within banking applications. Likewise, ergonomic issues, however important, are a non-goal of this project.

4) SDC may or may not retain its role as the centralized information handler, providing the current type of centralized transaction processing and database storage. This will be dependent on the functionalities put into the system. Current plans are to keep the storage and control of databases within SDC, due to both security, communication costs and manageability. Because of this, a functional concept like distributed database processing is considered outside the scope of this report, although it may contain some technological implications on data communication strategies.

5) the general characteristics of the savings banks' branch offices will remain unchanged, with the current distribution of small and large branch offices basically intact. Although changes in the organizational structure of the savings banks may be foreseen, this will not significantly change the number of branch offices, which today are served by TP2, nor the size of the branch offices. If it does, then it is just one more reason for the need to design TP3 for change.

6) SDC's role as the only official supplier of data processing know-how and services to the savings banks will continue. Even though some of the larger savings banks potentially could start implementing data processing services or buying computer systems themselves without any support from SDC, I will consider SDC to be capable of convincing them, that this approach cannot be beneficial. However, this role of SDC implies, that SDC can provide the savings banks with the needed processing facilities, either hardware or software, in a prompt way.

2.6 Previous and current system architectures

2.6.1 System architecture confinements

In this section, I will describe the basic components of the previous (TP1) and current (TP2) systems. The description will be partly an overview, partly a detailed explanation in those areas, where the impact from the rest of this report comes in.

2.6.2 Terminology

Branch	the restricted building or area of an individual savings bank office, where work and customer operations take place.
Workstation	the entity of computerized modules and equipment assigned to one individual user, i.e. the user's physical interface to the system.
Terminal	sometimes used as a synonym for workstation (e.g. in Terminal Project 2), but in SDC terminology primarily used to signify a video display terminal with its associated keyboard. A workstation is thus a terminal, plus possibly other devices like badge card reader, printer etc.
Module	a single device, which is part of a workstation.
Node	a device, possibly a workstation, connected to a network within the branch office.
LC	Local Computer, i.e. the main or master computer within the branch office.

TC Terminal Computer, i.e. a computer associated with an individual terminal or workstation.

Teller A person responsible for direct customer operations, and responsible for disposals and withdrawals of real money.

Back Office The area of the branch behind the teller area. Normally used to denote operations or people not involved directly with real money handling, but possibly with customer operations.

Transaction 1) an operation, which involves interaction with the computer system in the branch. 2) A message, which is sent to SDC's central systems for inquiries or database updates, and normally results in a response or answer.

2.6.3 TP1 - the Datasaab System.

The TP1 system was designed 1969-72 by the Swedish Saab-Facit Cooperation from the specifications of Nordisk Spardata (A joint-venture cooperation of the Nordic savings banks). The first series installation in Denmark of 2100 workstations took place from 1973-1975. Since then, additional 400 workstations have been installed.

The heart of the TP1 system is the Datasaab D5/20 minicomputer, a 16-bit processor with 26 instructions. Instruction execution time is 5-10 microseconds. Maximum addressing capability is 64 Kbytes, but no installation use more than 16 Kbytes. Memory technology is ferrite core. There is no interrupt system in the D5/20, so software timing is necessary. The D5/20 can control up to 6 non-intelligent workstations. A version D5/10 exists

for those systems with only one terminal.

Each workstation is made up of several individual modules, including key unit, transaction keyboard, indicator light panel, passbook printer, journal printer and a tally roll printer. There are no intelligence within workstations: all modules are controlled by the D5/20 through a parallel bus.

The applications software within the D5/20s are used for data entry, simple transaction and data verification, output editing, checksum computations. These are all simple functions, and all the real data processing is carried out in the central part of the on-line system, installed at SDC.

Software is written in the DIL5 macro language, which is interpreted.

Communication to SDC is done through leased loop lines from the Danish PTT. The loop line protocol is a BSC-like protocol, simple and effective, but non-standard. Datasaab is the only supplier of products for this protocol, besides SDC's McTan processor system. The Danish telephone company JTAS has also designed a concentrator system, based on PDP/11 minicomputers, for the more than 800 telephone terminals used as simple inquiry terminals in the savings banks, but these concentrators make use of the original Datasaab circuit board for the loop line connection through a specially designed bus-adapter board.

Up to 1050 D5 systems has been installed in 1020 branch offices throughout Denmark. The number of workstations has been up to 2500 in the fully operational TP1 system, which is currently being removed as TP2 moves in. Tables 2.1 and 2.2 show the pre-TP2 distribution of systems and terminals on the various system sizes (see SDC, 1978):

system size	# of systems	percentage	cumulative percentage
1 workstation	**338**	32.2%	32.2%
2 workstations	**309**	29.4%	61.6%
3 workstations	**190**	18.1%	79.7%
4 workstations	**120**	11.4%	91.1%
5 workstations	**64**	6.1%	97.2%
6 workstations	**29**	2.8%	100.0%
total:	**1050** systems		

Table 2.1: Distribution of TP1 systems

system size	# of systems	# of workstations	percentage	cumulative percentage
1 workstation	338	**338**	13.5%	13.5%
2 workstation	309	**618**	24.7%	38.2%
3 workstation	190	**570**	22.8%	61.0%
4 workstation	120	**480**	19.2%	80.2%
5 workstation	64	**320**	12.8%	93.0%
6 workstation	29	**174**	7.0%	100.0%
total:		**2500** workstations		

Table 2.2: Distribution of TP1 workstations

Figure 2.1 shows a block diagram of the branch office systems architecture based on the D5 system. Futhermore, figure 2.2 shows the distribution of the different D5 configurations using 1,2,..,6 terminals, according to table 2.1. The bar chart shows, that more than 50% of installations have 2 workstations or less and more than 75% of installations have 3 or less workstations. These figures will later be compared to TP2 numbers.

Figure 2.3 shows the total number of terminals according to the configuration size, as described in table 2.2. It shows, that over 60% of all TP1 workstations belong to the installations with 3 workstations or less, and 80% belongs to installations with 4 workstations or less.

As such, even though the total numbers may seem high, the TP1 system supports locally fairly small systems, where a complete second system is necessary, if a branch needs more than 6 workstations.

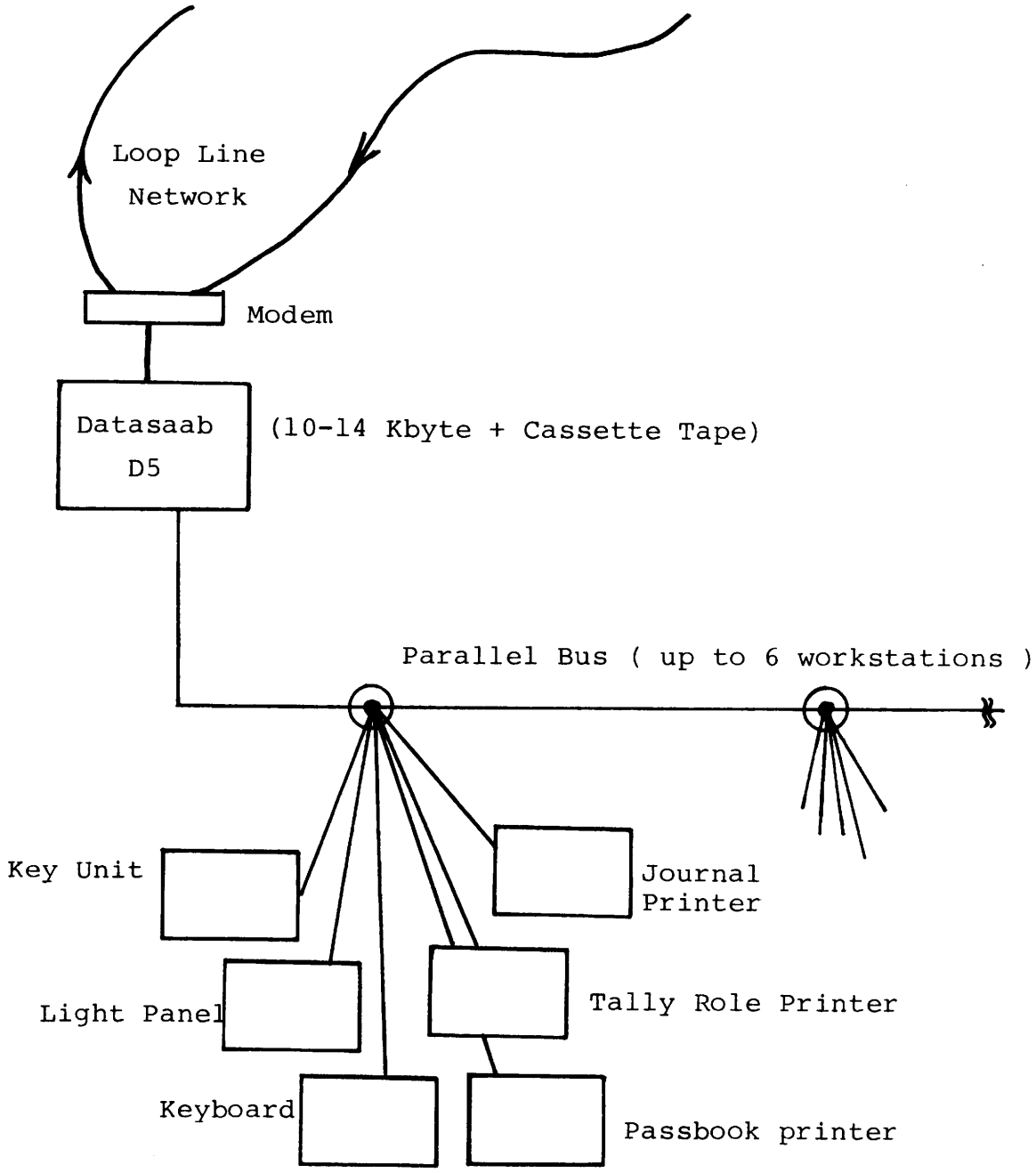


Figure 2.1: Configuration of the TP1 Datasaab system

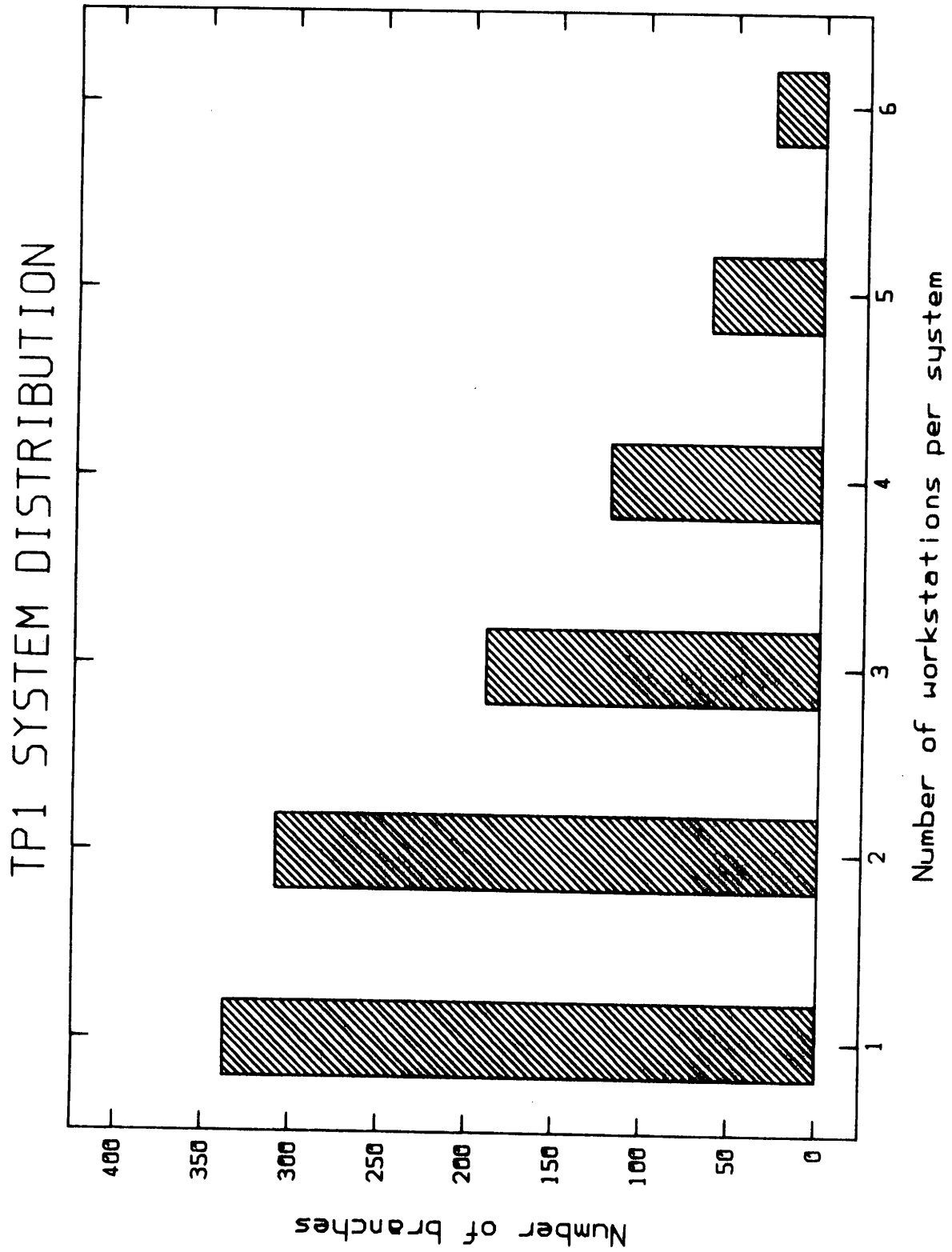


Figure 2.2: Distribution of TP1 systems

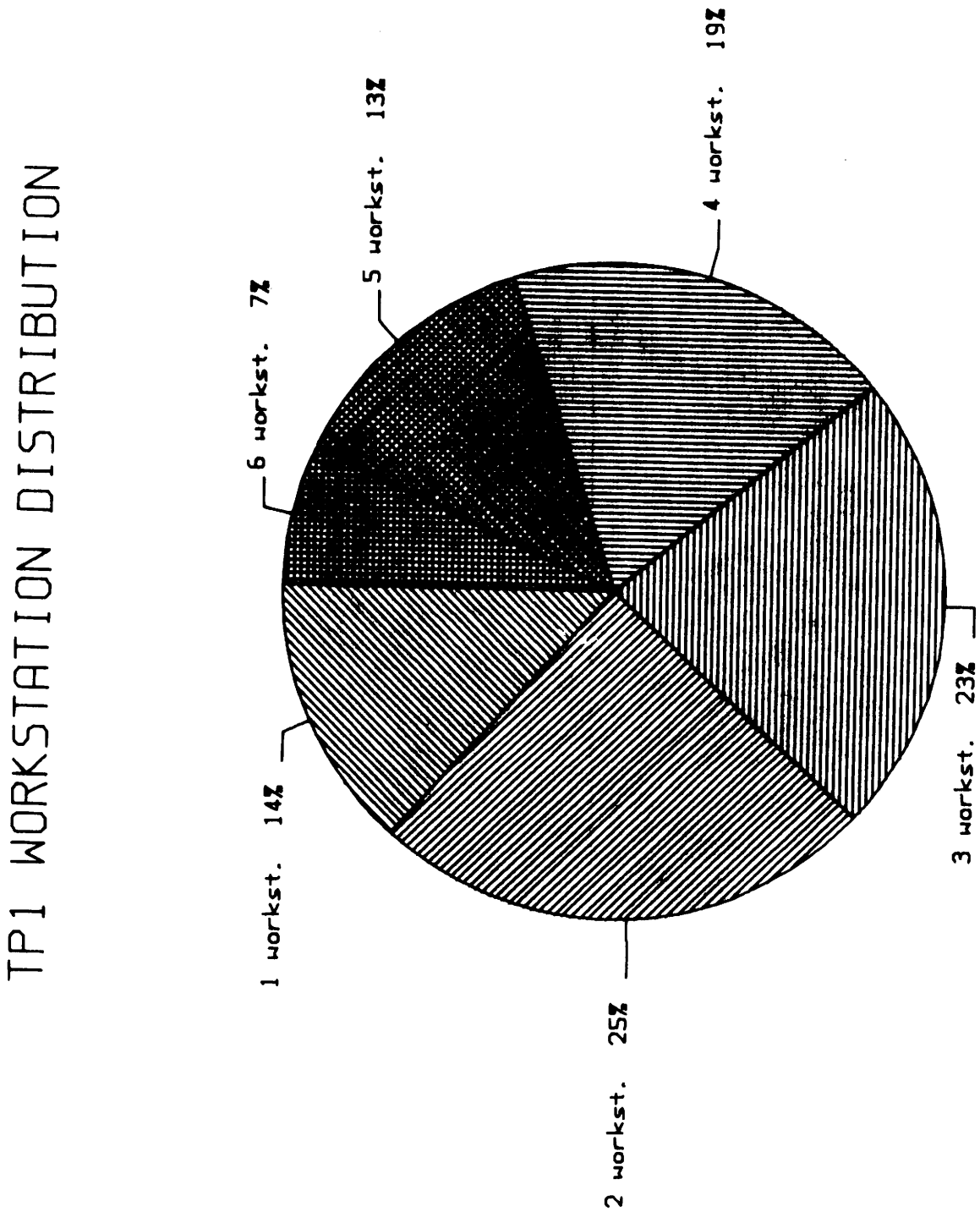


Figure 2.3: Distribution of TP1 workstations

2.6.4 TP2 - the Olivetti branch office system.

The TP2 project started in 1977 and resulted in the selection of Olivetti as the TP2 decentralized system supplier and Collins Radio Cooperation (now owned by Rockwell International) as the front-end supplier in 1979. The TP2 system is scheduled for installation 1982-85.

The branch office part of the TP2 system consists of a local computer (LC) and terminal computers (TCs). The local computer carries out functions common to all the attached terminals and controls common external units like disks, network interface and sometimes printers. At each workstation, the terminal configuration is built around an intelligent terminal, called the terminal computer or TC. The TC controls the connected terminal modules, and executes an application program capable of carrying out simple and frequently used functions. Two logical forms of the TC exists: a teller system, functionally much equivalent to the Datasaab system; and a back office system intended for more complex applications and office automation functions.

The local computer and the terminal computers are connected through the so-called high speed link within the branch office, and the entire system is connected to SDC through the Danish public data network. This basic configuration is shown is figure 2.4.

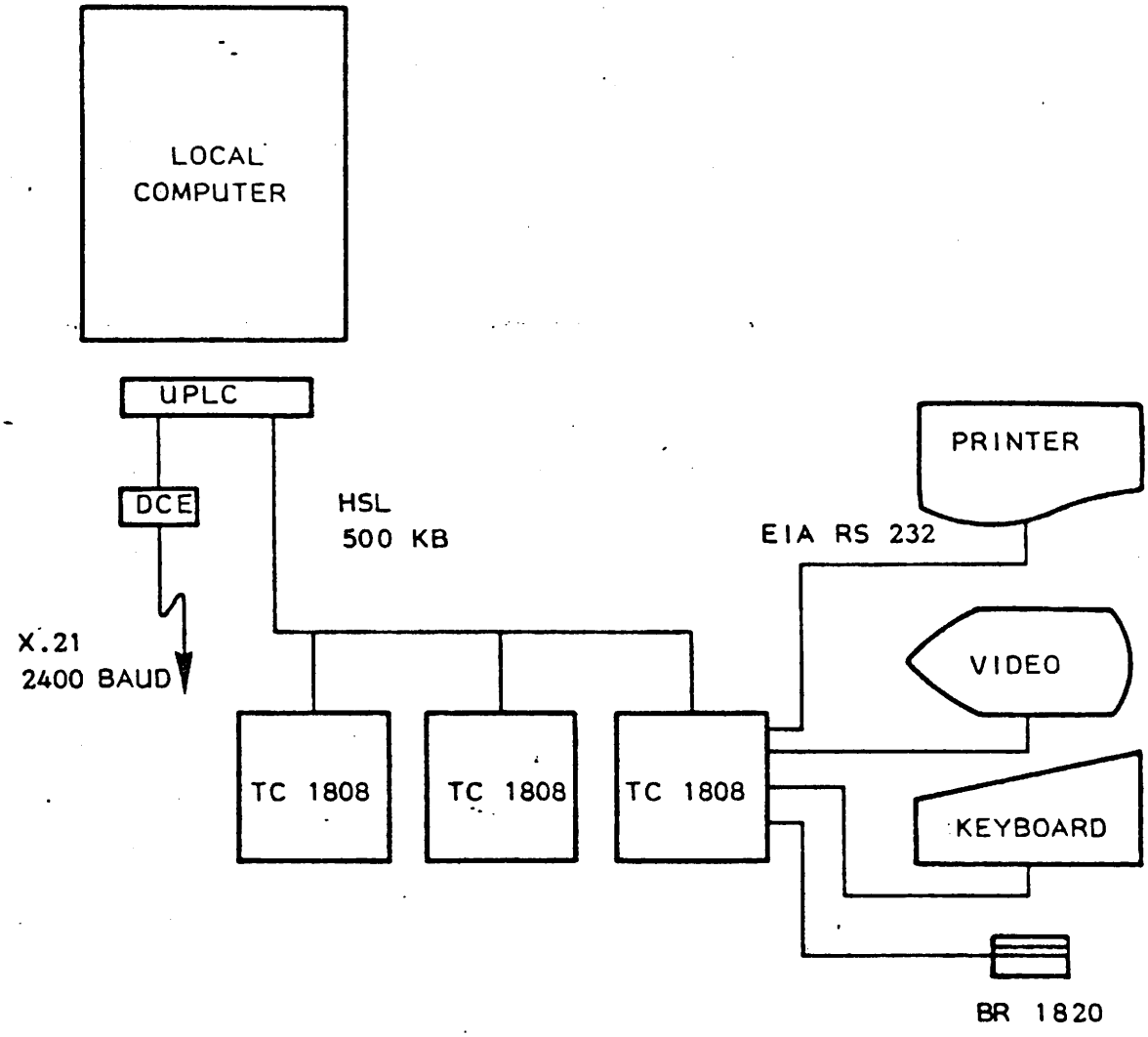


Figure 2.4: TP2 configuration

Compared to TP1, TP2 is a large system comprised of more local computers and more workstations. Also, the number of workstations per system are in general larger. Table 2.3 shows the figures for the ordered TP2 systems, taken from SDC's installation register in May 1983:

system size	# of systems	percentage	cumulative percentage
1 workstation	**160**	13.0%	13.0%
2 workstations	**366**	29.7%	42.7%
3 workstations	**249**	20.2%	63.0%
4 workstations	**159**	12.9%	75.9%
5 workstations	**93**	7.6%	83.4%
6 workstations	**60**	4.9%	88.3%
7 workstations	**40**	3.2%	91.6%
8 workstations	**25**	2.0%	93.6%
9 workstations	**16**	1.3%	94.9%
10 workstations	**19**	1.5%	96.4%
11 workstations	**11**	0.9%	97.3%
12 workstations	**11**	0.9%	98.2%
13 workstations	**4**	0.3%	98.5%
14 workstations	**4**	0.3%	98.9%
15 workstations	**3**	0.2%	99.1%
16 workstations	**3**	0.2%	99.4%
17 workstations	**3**	0.2%	99.6%
18 workstations	**0**	0.0%	99.6%
19 workstations	**0**	0.0%	99.6%
20 workstations	**1**	0.1%	99.7%
21-30 workstations	**2**	0.2%	99.8%
>30 workstations	**2**	0.2%	100.0%
total:	**1231** systems		

Table 2.3: TP2 system distribution

2.6.5 The TP2 Terminal Computer

The Olivetti TC-1808 terminal computer (see OL, 1981) is the user interface to the system. All teller transactions and back-office operations concerned with the user of the system are carried out through a TC. The TC is designed around a Z80® 8-bit microprocessor on several boards (see figure 2.5). Memory size is up to 132 Kbytes. Memory technology is mixed of 128 Kbyte standard dynamic RAMs, PROMs for initial program load and autodiagnostics functions, and 4 Kbyte non-volatile CMOS RAM for data, which must be kept during power failures. As the inherent addressing capability of the Z80 is 64 Kbyte, the larger memory is achieved by use of bank switching (see figure 2.7). The TC contains its own display controller for the attached VDU, which is a pure video monitor without any logic. Interfaces to other terminal modules like keyboards, badge card readers or printers are done through parallel or RS-232-C interfaces (see figure 2.6). The TC also contains a high speed link controller for communications to the LC.

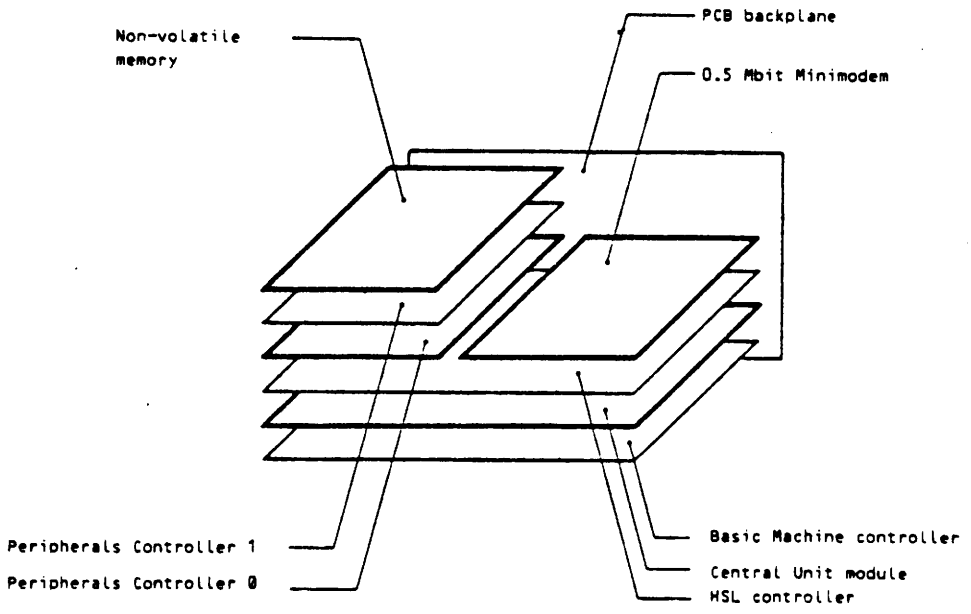


Figure 2.5: TC printed circuit board modules

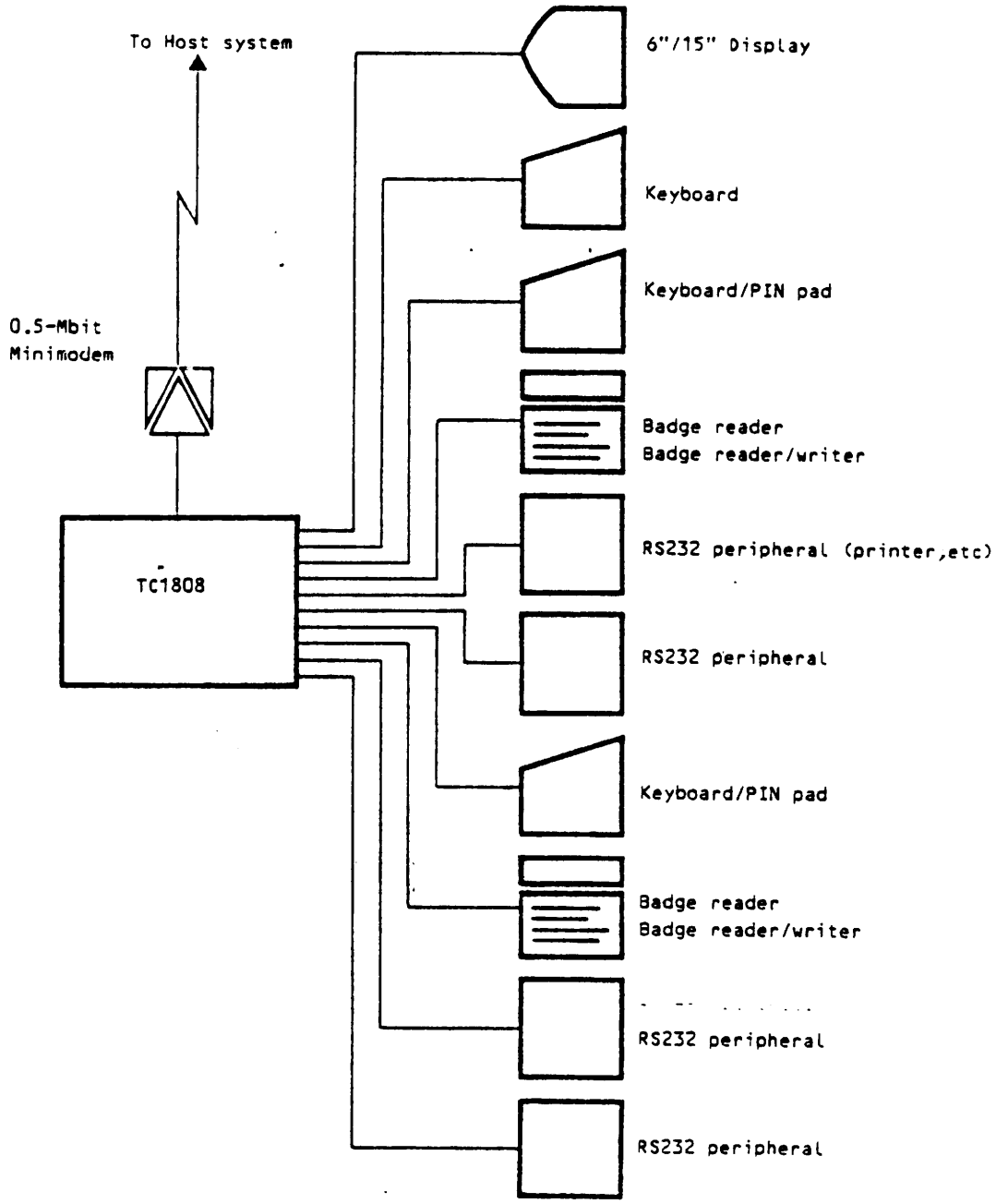


Figure 2.6: TC peripheral configuration possibilities

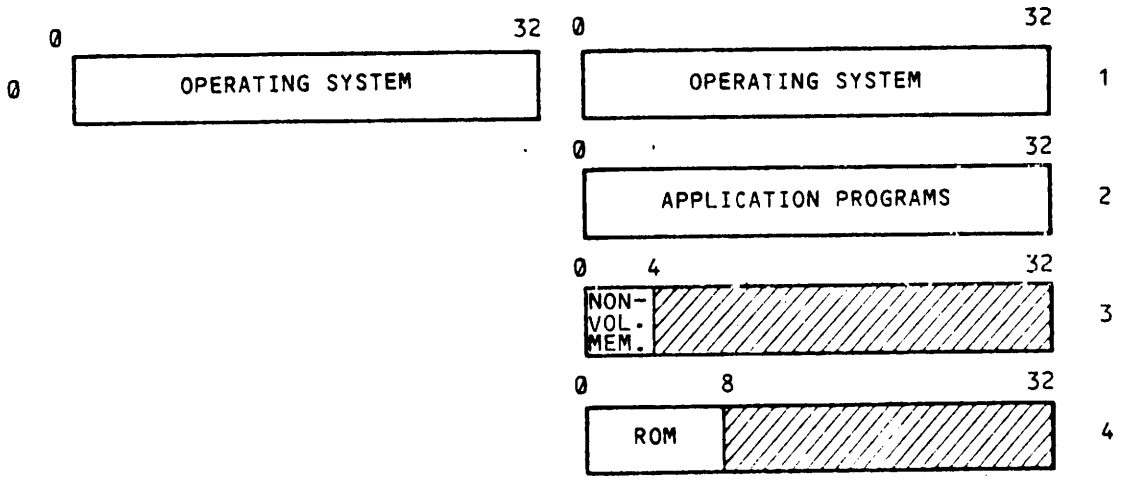


Figure 2.7: TC memory organization

2.6.6 The TP2 Local Computer

The Olivetti S1000 local computer (LC) has two basic roles: it acts as a terminal concentrator for the terminal computer and it is a communications device for the network.

The LC equipment is housed in one cabinet, and consists of central processing unit, memory, disk system, power supplies and the Universal Programmable Line Controller (see OL, 1981.2; OL, 1981.6).

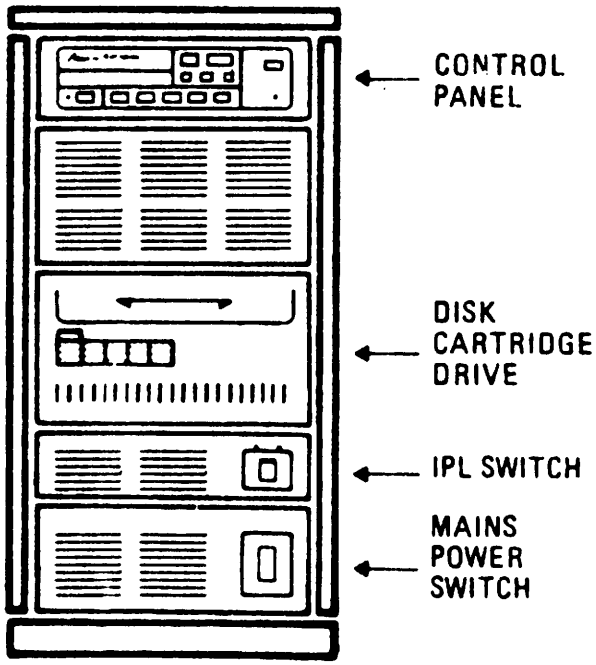


Figure 2.8: S1000 LC

The central processing unit is a three board processor from Microdata in USA. It is a 16-bit, byte-addressable, microprogrammed machine. Its internal architecture is stack-oriented. The three PC boards are mutually interconnected to form the CPU. They communicate with other boards (like memory and I/O) through a bus architecture called the Monobus (see OL, 1977). Bus communication on Monobus is based on a master/slave relationship in which the requesting unit is master and the addressed unit is slave. The CPU is master by default and does not perform bus requests. The CPU has the lowest priority on the bus.

The following table shows some of the performance characteristics of the CPU, memory, and bus:

- write cycle time:	675 ns
- read cycle time:	750 ns
- read access time:	525 ns
- firmware cycle time:	150 ns
- CPU/memory transfer rate:	1.25 Mword/sec
- DMA transfer rate:	1.25 Mword/sec
- concurrent i/o transfer rate:	50 Kwords/sec
- data lines:	16
- address lines:	20
- control lines:	19
- Monobus addressing range	1 Mbyte

2.6.7 S1000 typical instruction timing

Instruction timing is dependent on the stack mechanism in the CPU. The data stack is the area of memory allocated for an individual S1000 user's data. It is a push down stack which is specified by four registers: Stack Base (SB), Stack Length

(SL), Environmental Pointer (EP) and Stack Pointer (SP). In order to gain a speed advantage, five special hardware registers are provided as a dynamic head to the stack (see Figure 2.9). The data stack can be up to 64 Kbyte long.

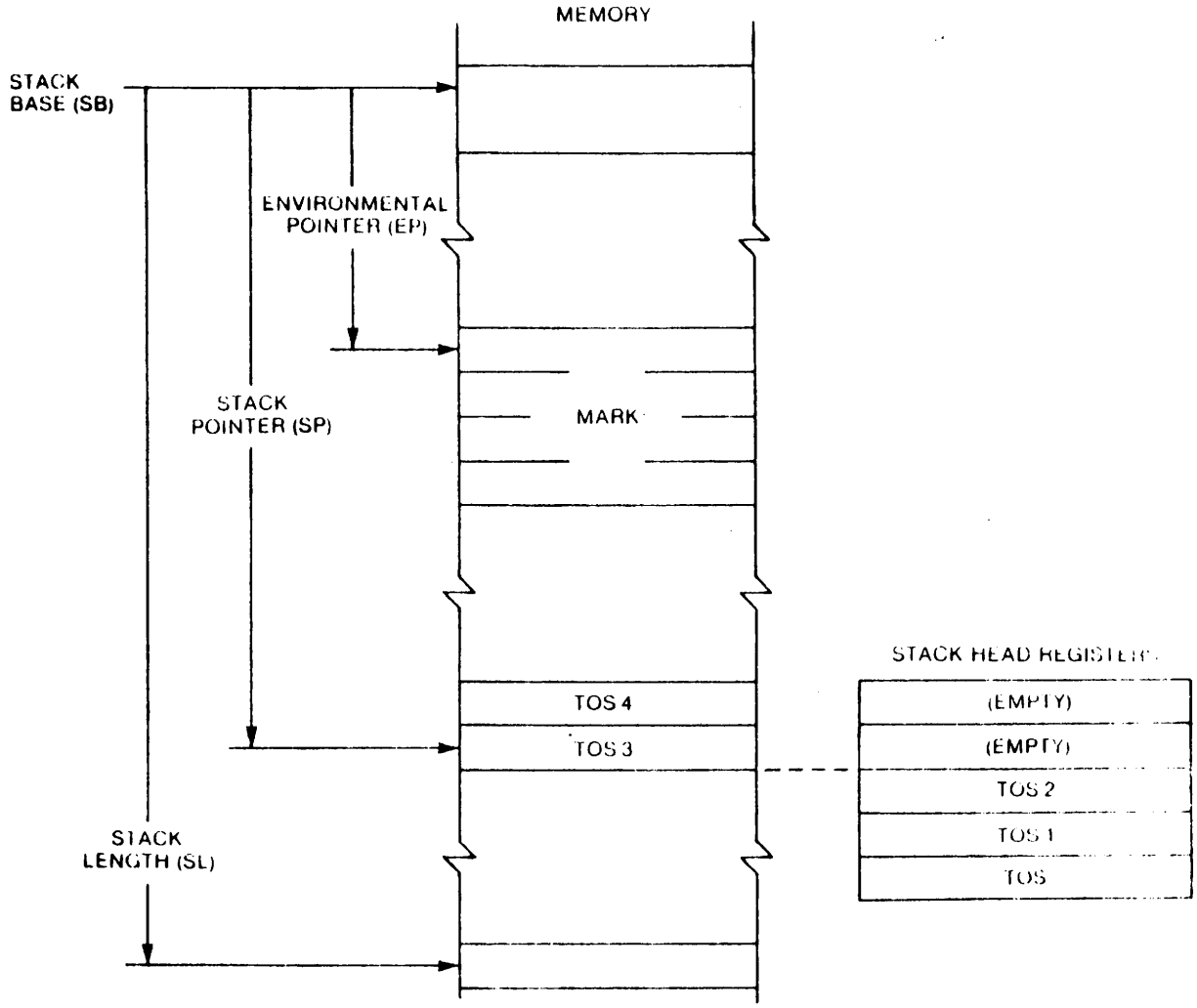


Figure 2.9: S1000 data stack format

Within the data stack, four-word entries called "marks" are installed, that delineate the location in the data stack, where a new procedure block or begin-block began to use the stack. On figure 2.9, the top of stack is denoted TOS. All basic operations works with operands in the top of the stack, and if possible, these operand will be placed in the fast hardware register array, called the Stack Head Register (SHR).

Below are some representative instruction timing values for the S1000 computer (see OL, 1979):

instruction	time (microseconds)			
ADD, SUB	0.75	(1)	1.8 (2)	3.15 (3)
AND,OR,XOR	-------	same	----------	
compares	-------	same	----------	

where (1) = operands in SHR
(2) = 1 operand in SHR
(3) = no operands in SHR

BRANCH	3.45			
LOAD WORD	3.8	(1)	6	(2)
STORE WORD	3.8	(1)	4.7	(2)

where (1) = Top of Stack in SHR
(2) = Top of Stack in main memory
(Local direct addressing mode)

These figures does not give the full view on S1000 performance. Complex arithmetical computations with long equations will fully use the stack structure, but administrative or communications oriented computation using many compare instructions will be slow due to the fact, that a compare instruction replaces the two operands on top of the stack with a true or false result. Thus a conditional branch requires a compare immediately before the branch instructions. In typical cases, where a data value (e.g. an input) has to be compared to some alternatives, this must be implemented as a load of both the data value and the comparison value, then the compare instructions must be executed. If then the data must be

compared with the next alternative, the entire process of loads and compare must be repeated.

The result is, that the LC actually represents a low performance architecture in those applications, where control and not arithmetic operations are predominant.

2.6.8 S1000 circuit boards

The S1000 computer is implemented with printed circuit boards for which the system chassis provides space for either 16 or 22 boards. The board size is 12 inches times 14 inches (30.0 x 35.0 centimeters). The 14" edge plugs into a pair of 50-pin connectors in a multi-layer printed circuit board backplane. This backplane contains the wiring for the Monobus, for control memory buses and for CPU buses.

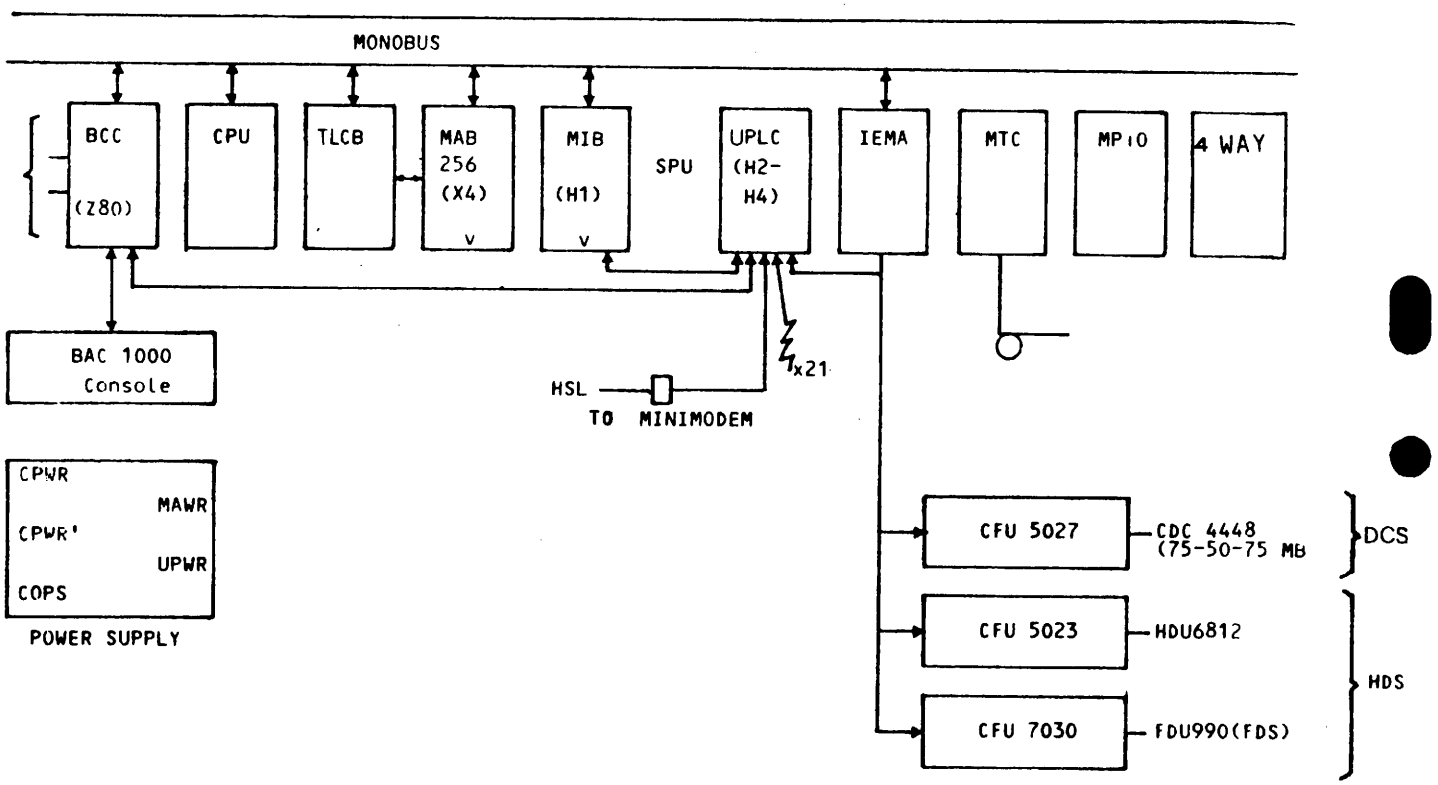


Figure 2.10: System 1000 - block diagram

The S1000 CPU itself, with the S1000 firmware, is contained on three printed circuit boards modules: the Monobus interface board, the data board and the processor control board. The read-only memory chips for the firmware are contained on the processor control board.

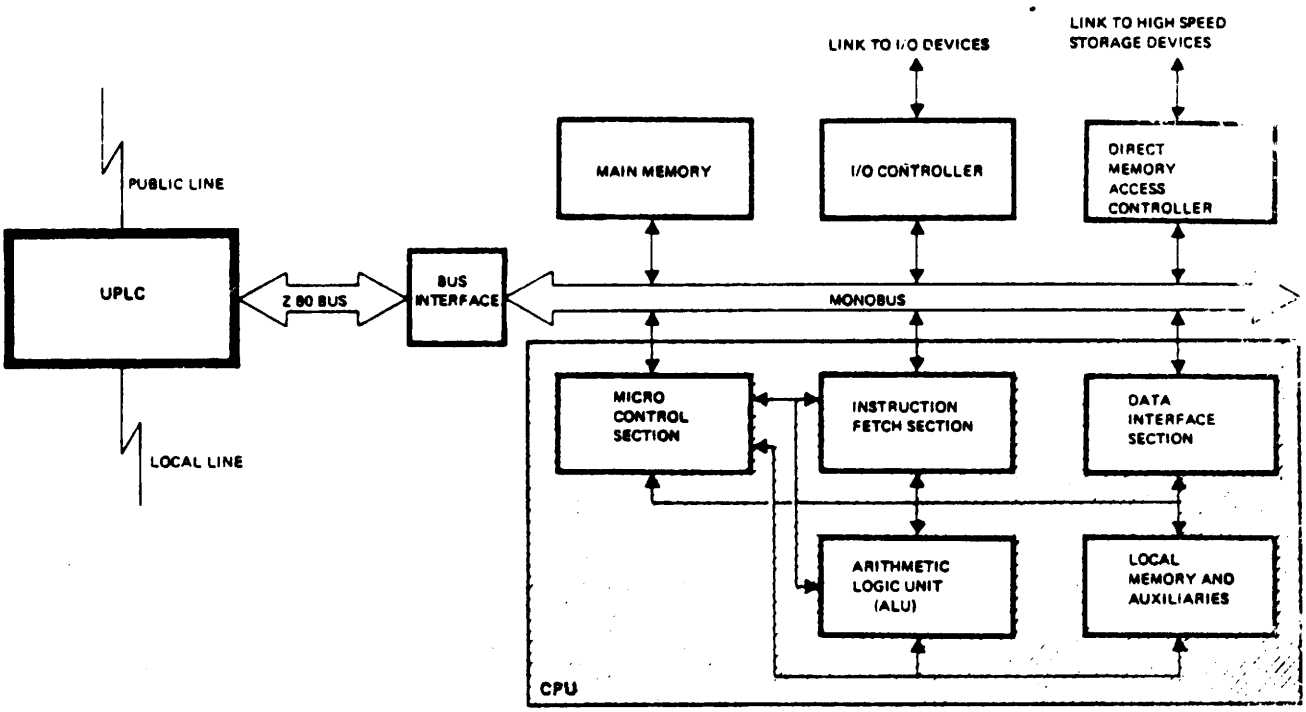


Figure 2.12: Schematic view of LC

Closely associated with the CPU are the BCC board, the basic console and the memory boards.

The Basic Console Controller (BCC) is a microprocessor based controller board that controls the Basic Console indicators and the information coming from the console's keys and switches. It stores diagnostic information for later retrieval and generates system signals. Futhermore, it contains a serial input/output controller and a parallel line printer controller.

Three types of boards provide memory and memory control functions: a 128 Kbyte memory board, a 256 Kbyte memory board and a timing/logging control board. Each memory board contains its own refresh logic. The timing/logging control board (TLCB) performs control and timing functions between the bus and the memory boards. It also handles the error detection/correction.

The Universal Programmable Line Controller (UPLC) is a three board front-end communications processor, designed to handle two types of communications:

a) connection to a public network via 2 X.21 synchronous full-duplex lines, using bit-synchronous HDLC as the data link protocol

b) connection to a local high speed multipoint line.

The UPLC is capable of maintaining communications, even when the local computer fails or is powered down.

An IEEE-488 interface board handles communications to secondary storage. Secondary storage consists of 25 Mbyte hard disk units, 25 or 50 Mbyte disk cartridge units or 1 Mbyte floppy disk units. Later this year, systems will be equipped with a Kennedy start/stop 1/4-" cartridge tape drive for backup.

2.6.9 The high speed link

The high speed link (HSL) is the communications media between the LC and TCs. The speed is 500000 bps, and it uses a bitoriented HDLC protocol. Logically, it is a multipoint line, where the LC is the master and the TC are the slaves, although a TC can initiate a transfer. On the LC, the UPLC is responsible for the high speed link interface. On the TCs, a special high speed link controller is installed. The physical

medium is a twin coaxial cable. A maximum of 20 TCs can be connected to one cable. The LC has provision for installing a second UPLC, so up to 40 workstations can be attached to one LC.

2.6.10 TP2 overall architecture

The TP2 system is a centralized architecture, where the LC minicomputer controls the intelligent terminals, the TCs. The vast majority of applications executing is performed by the LC.

Intentionally, the LC have been selected by Olivetti to be capable of handling the load of a fully configured system, i.e. a system with up to 40 workstations. Table 2.3, visualized in figure 2.12, shows the distribution of systems with different number of workstations. It turns out, that more than 50% of the installations have 3 workstations or less, and that more than 90% of the installations have 7 or less workstations.

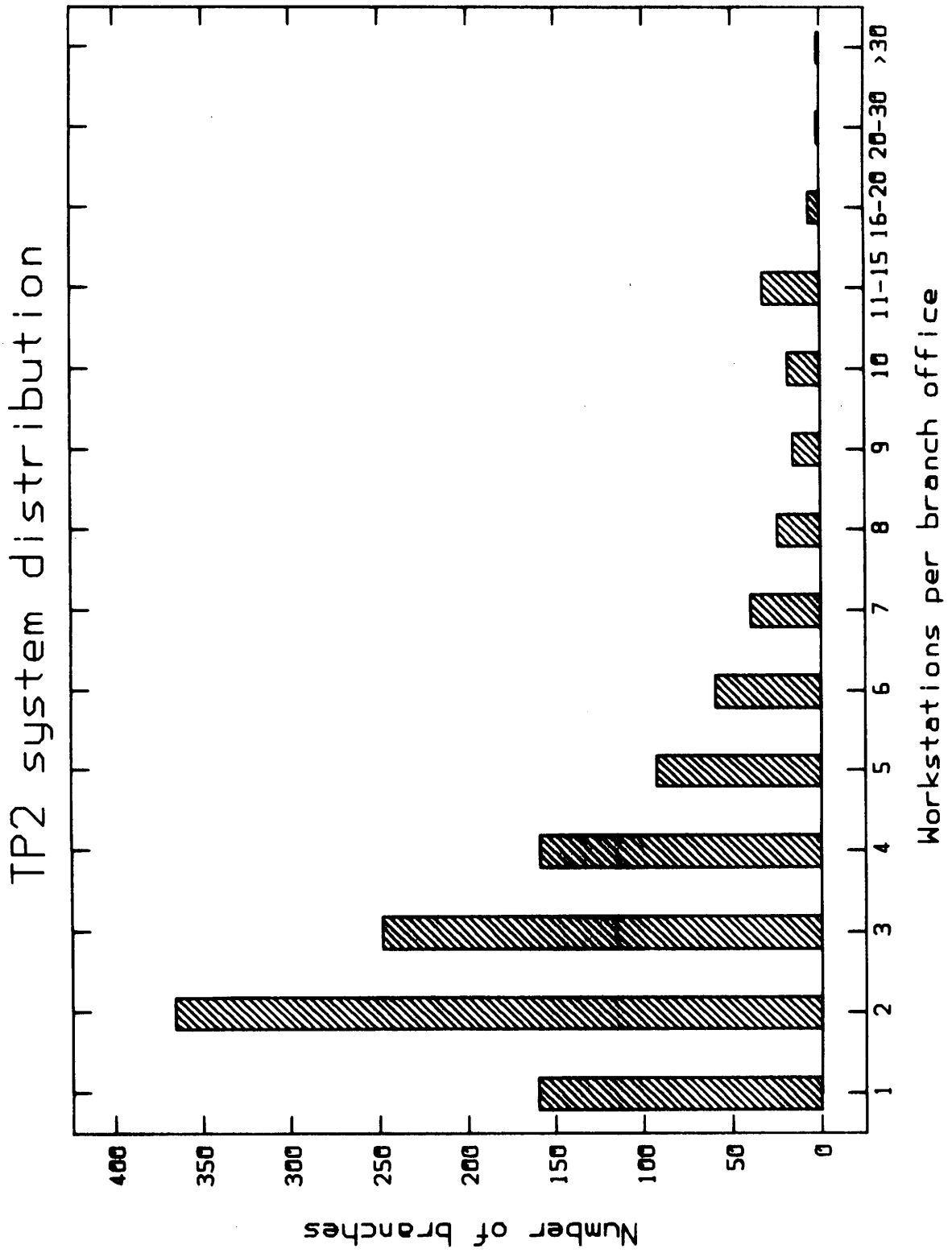


Figure 2.12: TP2 system distribution

However, by looking at the workstation distribution in figure 2.13, showing the numbers from table 2.4, one finds that more than 50% of the total number of workstations actually are connected to systems with 6 workstations or more.

system size	# of systems	# of terminals	percentage	cumulative percentage
1	160	**160**	3.5%	3.5%
2	366	**732**	16.1%	19.6%
3	249	**747**	16.4%	36.0%
4	159	**636**	14.0%	50.0%
5	93	**465**	10.2%	60.2%
6	60	**360**	7.9%	68.1%
7	40	**280**	6.1%	74.2%
8	25	**200**	4.4%	78.6%
9	16	**144**	3.2%	81.8%
10	19	**190**	4.2%	85.9%
11	11	**121**	2.7%	88.6%
12	11	**132**	2.9%	91.5%
13	4	**52**	1.1%	92.6%
14	4	**56**	1.2%	93.9%
15	3	**45**	1.0%	94.9%
16	3	**48**	1.1%	95.9%
17	3	**51**	1.1%	97.0%
18	0	**0**	0.0%	97.0%
19	0	**0**	0.0%	97.0%
20	1	**20**	0.4%	97.5%
20-30	2	**45**	1.0%	98.5%
>30	2	**70**	1.5%	100.0%
total:	1231 systems	**4554** workstations		

Table 2.4: TP2 workstation distribution

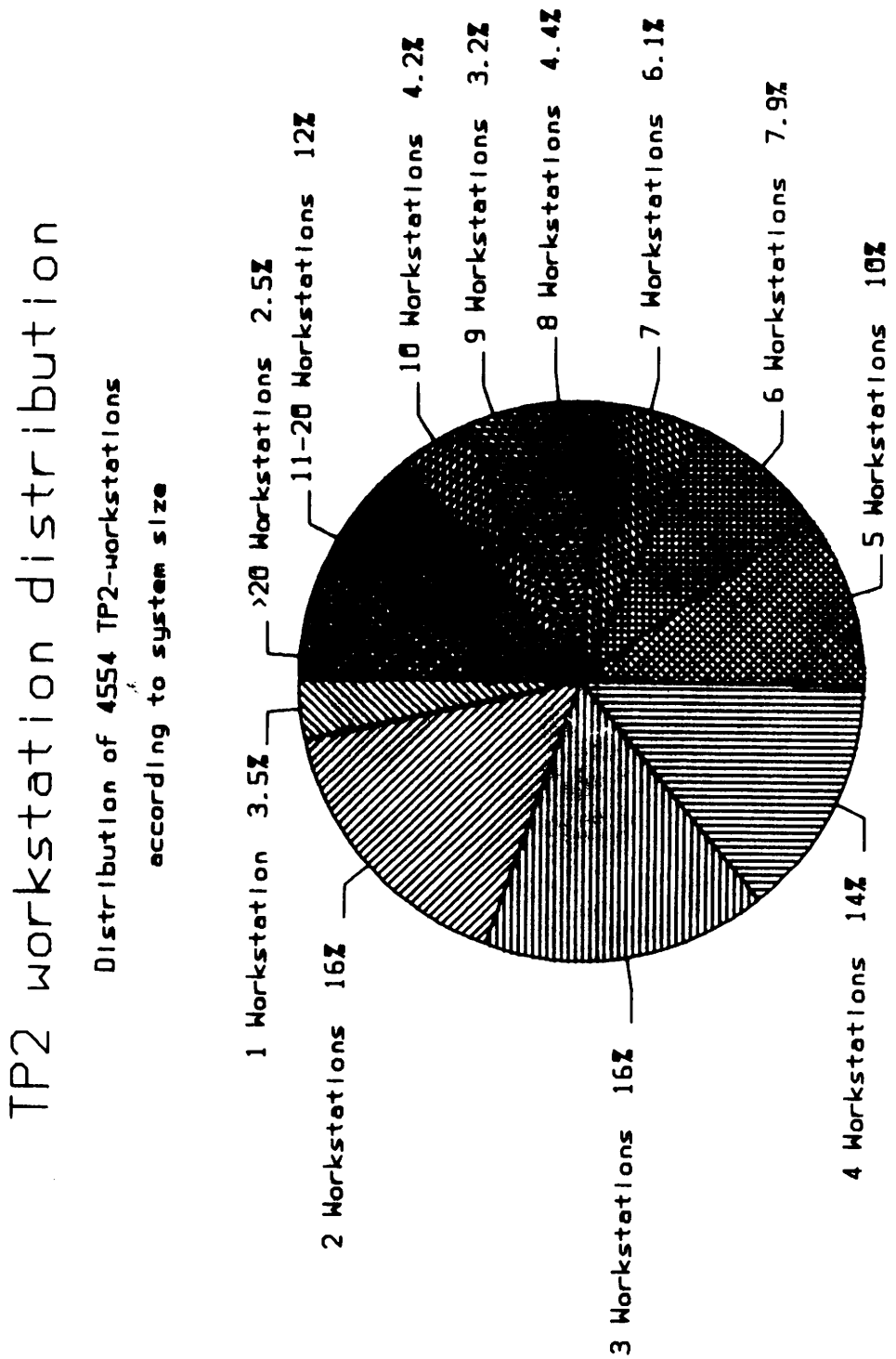


Figure 2.13: TP2 terminal distribution

Table 2.5 shows the relationship between teller terminals and backoffice terminals in the various system sizes.

system size	# of sys	# of workst.	teller workst.	office workst.	teller per sys	% of office
1	160	160	4	156	0.0	97.5%
2	366	732	286	446	0.8	60.9%
3	249	747	315	432	1.3	57.8%
4	159	636	254	382	1.6	60.1%
5	93	465	200	265	2.2	57.0%
6	60	360	137	223	2.3	61.9%
7	40	280	108	172	2.7	61.4%
8	25	200	66	134	2.6	67.0%
9	16	144	46	98	2.9	68.1%
10	19	190	45	145	2.4	76.3%
11	11	121	41	80	3.7	66.1%
12	11	132	32	100	2.9	75.8%
13	4	52	17	35	4.3	67.3%
14	4	56	13	43	3.3	76.8%
15	3	45	13	32	4.3	71.1%
16	3	48	17	31	5.7	64.6%
17	3	51	6	45	2.0	88.2%
18	0	0	0	0	0.0	0.0%
19	0	0	0	0	0.0	0.0%
20	1	20	6	14	6.0	70.0%
20-30	2	45	11	34	5.5	75.6%
>30	2	70	15	55	7.5	78.6%
total:		4554	1632	2922		

Table 2.5: Teller versus backoffice distribution

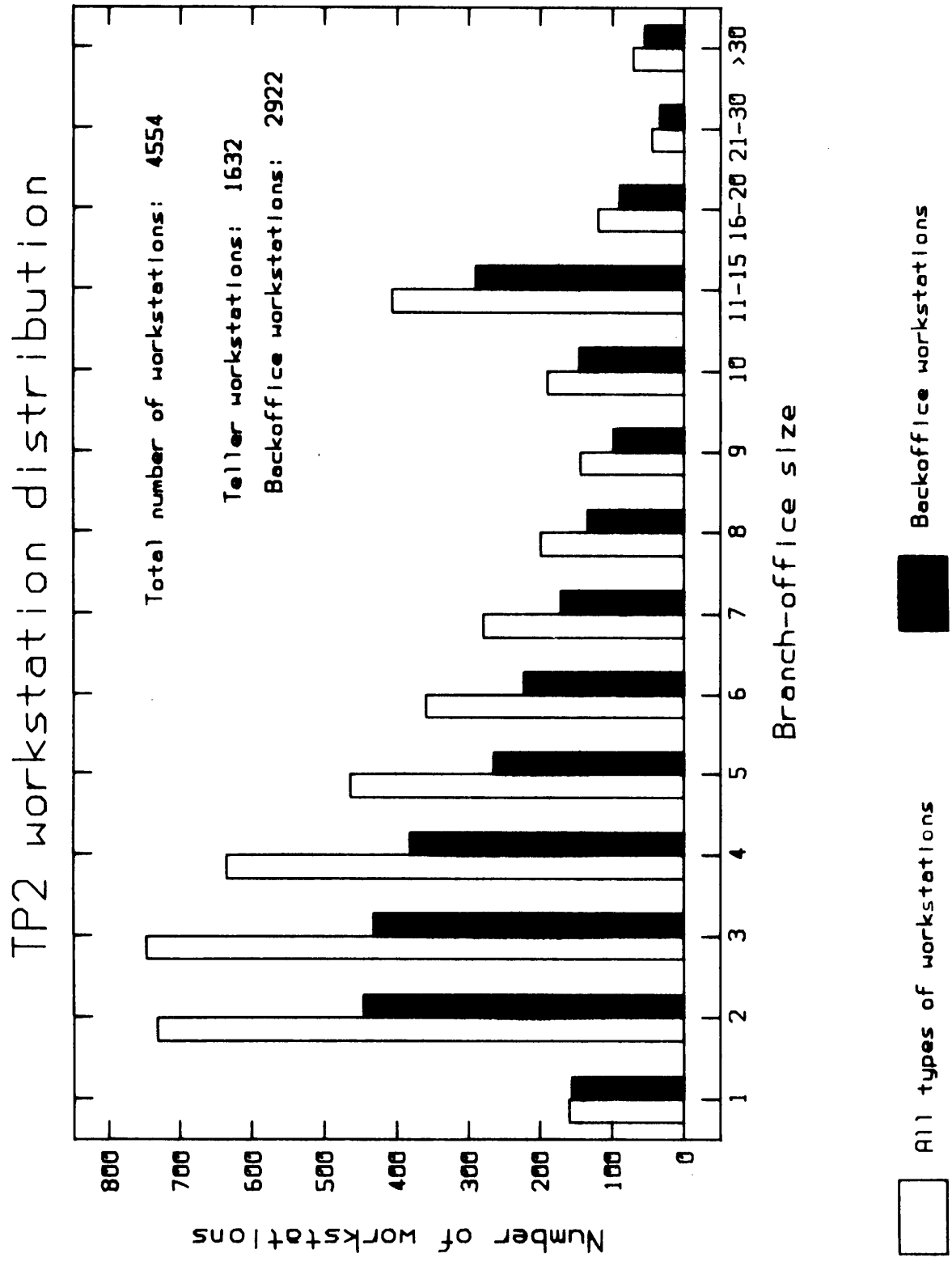


Figure 2.14: TP2 terminal distribution

Compared with the Datasaab D5 figures, it seems that the impact by TP2 on most of the installations has been a shift in the distribution curve to the right by one or two extra workstations per installation. Also, from figure 2.14, it can be seen, that the predominant part of workstations within a given system configuration actual are of the back-office type, especially at the very smallest configuration. Indeed the number of pure teller workstations has decreased compared to TP1.

Taking into account, that the TP2 system concept was designed to eventually support the large installations (of up to 40 TCs) with the same LC hardware as for the smaller installations, the majority of the installations are seemingly focusing a higher cost/performance value, than the system has been designed towards.

Next, there is a paradox of the other end of the scale: the systems with more workstations use the LC as the common processing resource. This means, that the available computing power to a user decreases as the number of workstations increase. This decrease in performance depends on many factors. By the nature of human interaction with a data processing system, all users will not present an equal load constantly to the LC, so based on this fact the decrease will be less than proportional to the number of workstations. On the other hand, the nature of the operating system in the LC using virtual memory and swapping of user areas will cause a decrease in performance more or less exponential to the number of workstations. On the average, we will therefore make the assumption, that the decrease in single-user performance will in fact be proportional to the number of workstations.

The performance ratio between the LC and the TC in the TP2 system is not documented and will be dependent of the type of application programs running on the computer. A fair estimate for the type of applications used by SDC (excluding any concern about memory addressing capability) will however be, that the LC has approximately four times the performance of the TC.

This means, that on a TP2 system with 10 workstations, the performance characteristics of the "logical" LC, the user sees, will be 10 times worse than the performance characteristics of the physical LC. The effect is, that the LC provides the user with a processing power much LESS than the processing power the user has in the TC!

There is no provision for a higher performance LC in these cases. The LC already uses high performance technology. Also, the Monobus bus architecture is inhomogeneous and does not allow multiple processing units within the LC. Similarly, the high speed link does not allow multiple LCs to be attached.

The result is, that TP2 locks SDC into a dilemma, where change means decreased performance, and where change is required for savings banks focusing still higher utilization of TP2.

Also, looking at figure 2.13, we find that the predominant part of the installations have few workstations. This characteristic of the distribution will hold true in the future, even if more workstations are installed in each branch office to support new needs for word processing and office automation. The ultimate need will occur by providing every employee with a personal workstation.

An overview of this situation can be obtained by looking at the current distribution of branch offices versus the number of employees per branch:

number of branch offices with	1)°	2)°	total	# of empl.°	estimated† branches		estimated† employees	
1-2 employees	111				**185**	12.2%	278	3.0%
3 employees	106	374	935	3100	**177**	11.7%	530	5.8%
4-6 employees	344				**573**	37.9%	2867	31.5%
7-8 employees	188				**249**	16.4%	1865	20.5%
9-10 employees	103	94	578	6000	**136**	9.0%	1295	14.2%
more than 10	193				**193**	12.8%	2266	24.9%
totals:	1045	468	1513	9100	1513	100.0%	9100	100.0%

(source: Dfs1,1983)

Notes:

1): 14 largest savings banks (representing 90% of the employees).

2): Branches of remaining savings banks. It is estimated, that 80% of these branches are manned with 6 employees or less.

°: Figures estimated by DfsL.

†: Figures estimated by myself

Table 2.6: Distribution of savings banks employees

As can be seen from these figures, the majority of branches (and hence systems) will have a rather limited need for a high number of workstations to be attached to one branch system. However, even though the number of larger branches are comparatively small, they still represent a larger part of the employees and hence workstations.

The need for a high number of workstations per system or rather per branch office is found in the larger head offices of the savings banks, where teller operations are minor compared to office automation functions. A fully integrated system for these head offices may require the need to interconnect several hundreds of workstation, however probably not more than 500. The TP2 is inadequate for handling these requirements, unless several LC's are installed within an office. This, however, implies a non-homogeneous system.

The following two tables show some of the cost factors for the TP2 system. The figures quoted are from the official Olivetti TP2 price list and represent valid prices of 3'rd quarter, 1983.

It should be noted, that the values shown are those used to invoice the savings banks, based on the expected amount of ordered equipment. For even larger orders, they will be subject for further quantity discount.

Table 2.8 shows the pricing effect on various system sizes. In the table shown, only the costs for the pure workstation or computer have been included. As far as the workstation concerns, it include the terminal computer plus the display and keyboard. As such, it resembles the configuration of many of todays personal computers, although these include mass storage devices as well.

The LC is the old standard version with 1 Mbyte of memory. Cost for the new versions to be installed with a cartridge tape backup has not been calculated. Also, the cost for the high speed link cable and interconnection has not been included.

Hence, the figures represents the effect of having to share the cost of the local computer between the workstations. If the figures from table 2.3 can be remembered, it means, that more than 75% of all installations have a cost overhead per workstation of at least 52500,- Dkr, and more than 90% of all installation must bear an overhead of at least 30000,- Dkr per workstation.

Especially for the smaller branch offices, the workstation price may be prohibitively high.

TP2 single components:

Local computer with 1 Mbyte memory, UPLC, line interface, 25 Mbyte hard disk, 1 Mbyte floppy disk:	209.820,- Dkr.
TC1808 terminal computer, 128 Kbyte memory, 4 Kbyte non-volatile memory:	17.860,- Dkr.
Display monitor, 2000 characters:	11.370,- Dkr.
Display monitor, 520 characters:	8.580,- Dkr.
Alphanumeric keyboard:	4.210,- Dkr.
Numeric keyboard:	3.060,- Dkr.
Badge card reader:	2.070,- Dkr.
Passbook printer:	17.590,- Dkr.
Matrix printer, dual definition:	28.990,- Dkr.

Composite workstation configurations:

TC computer with display and keyboard:	33.440,- Dkr
Teller workstation	49.160,- Dkr
Backoffice workstation	53.100,- Dkr

Note: All prices have been rounded to the nearest 10 Dkr.

Table 2.7: Cost factors for TP2 modules

Number of workstations per system	Total Computer price	Price per workstation	LC part per workstation
1	243300	243300	209800
2	276700	138400	104900
3	310100	103400	69900
4	343600	85900	52500
5	377000	75400	42000
6	410400	68400	35000
7	443900	63400	30000
8	477300	59700	26200
9	510700	56700	23300
10	544200	54400	21000
15	711400	47400	14000
20	878500	43900	10500
30	1212900	40400	7000
40	1547300	38700	5200

Note: Workstation price includes a terminal computer with 2000 characters display monitor and alphanumeric keyboard, but no other peripherals. LC price includes an 1 Mbyte system, with UPLC and 25 Mbyte harddisk plus one floppy disk, but otherwise no other peripherals. No costs have been estimated for high speed link cabling and connectors, nor have other installation costs been taken into account.

All prices has been rounded to the nearest 100 Dkr.

Table 2.8: LC impact on workstation cost

2.7 TP2 performance problems.

The specification above of the TP2 system shows the system components, which are going to be installed. Throughout the TP2 project, it was thought, that this system was able to meet the requirements from SDC. I think it is worthwhile to state some of the initial experiences with the system, which have shown up. These may or may not be present in the final installation of the system, but they indicate some of the factors influencing TP3 design.

* Originally, a memory size of 128 Kbyte on the LC was thought to be sufficient for many installations. It soon turned out to be to little, so memory was increased just to hold the amount of applications software needed. When applications was implemented on the system, response time turned out to be too large. One reason was a trashing condition due to virtual memory swapping (see GSG, 1982). As a result, all system will now have 1 Mbyte of memory, and this figure is already too low to handle many future applications. There is however no provision for adding more memory on the LC.

  The reason for this has been both an underestimation of the needed software on the LC from SDC's side and a corresponding lack of insight in banking software needs from Olivetti's side. One must remember, that the SDC estimates for software was based on the old TP1 system (with max. 16 Kbyte of memory) and the way the centralized applications were build. But the central applications misses any form for user-oriented I/O processing, which is a major part of any decentralized system. The same arguments hold true for the TC memory size, which is also a problem today.

* The high speed link is slow, with an effective data rate of approximately 4 Kbyte/second. This is partly due to the type of protocols used on the line, the organization of the UPLC software and the fact, that I/O operations on the LC are slow. It takes about one minute to load a TC, which can be a problem for the larger installations.

  The slow speed of the high speed line have also necessitated change in some applications, which did extensive message changes between the LC and the TC over the link.

* Still, however, performance problems exists, but these problems now reside in the basic architecture of the LC and its software. One example is the file system, which is comparatively slow, with transfer rates around 10 Kbytes/second. As a result, keyed files are not used due to performance reasons and due to incompatibility with other file types.

* File backup and software distribution have been badgered with floppy disk problems. Disregarding the oddity, that the floppy disk driver is emulating a tape drive (which means, that all reads from the floppy disk must be performed sequentially), this is due to two facts: The Olivetti floppy disk controller uses a higher flux rate than the rest of the entire floppy disk market, which means that not all brands of floppy disks can be used, and secondly, the controller does not retry a read from the floppy disk if an error occurs. The transfer rate to the floppy disk is extremely slow - between 1-3 Kbyte/second (see OL, 1982.3).

* TP2 modules (LC, TC, terminal, keyboard) have been dropped in Olivetti's product plans and the new system line from Olivetti is not compatible with TP2. Likewise, terminals and keyboards in the TP2 system cannot be used with the new Olivetti systems.

* TP2 has suffered from stability problems. These have partly been due to hardware problems, especially in the book printing devices and the winchester hard disks, but mainly due to software problems, as the software is unable to do recovery at almost any level if any irregularities occur from e.g. input or communications lines.

These incidents are to show, that problems exist, even in a well-planned project like TP2. But some of these problems should never have occurred or they should have been foreseen. They also show the problems, that will be present if a system cannot be overviewed clearly or cannot be changed easily.

2.8 Conclusions

As have been illustrated in this chapter, the new terminal project, TP2, for the Danish savings banks is a large project. It continues the route taken by SDC in the seventies, where processing facilities are distributed to the individual branch offices.

In terms of concepts, the TP2 system is probably one of the most advanced banking systems being designed today.

In terms of technology, however, the system is not possessing the needed processing power and the needed flexibility and expansion possibilities, that are necessary to implement the new application concepts, which are needed by the savings banks.

The TP2 system caused a total replacement of the old TP1 system. A total replacement of the TP2 system will be needed faster than anticipated, if nothing is done to bring the technology used within the system up to a state, which is not necessarily better, but at least comparable with the rest of the market.

The following chapters will focus on the related technological areas, which will provide the means to migrate from TP2 to something, that hopefully does not possess the current limitations.

This page intentionally left blank

CHAPTER THREE

TECHNOLOGY IMPACT ON THE SAVINGS BANKS' SYSTEMS

3.0 Introduction

In just one decade, technology progress has changed the computer industry into a state, that could not be foreseen by traditional forecasts and still is not widely accepted by industrial enterprises and educational environments.

This state is that of the invasion of computer power into almost every field of application, arising from the fact, that computer power has become cheap and easy to apply.

One single event in the technological development, that started this change, was the introduction of the low-cost microprocessor - a single integrated, electronic circuit containing the logic and functions of a central processing unit. Basically, the microprocessor is just a product of the progress within microelectronics, where more and more components are integrated on the same chip, thus lowering production cost and enhancing reliability. But the microprocessor is more than that: it is the concept of having a cheap and ready-to-use building block, which enables a designer to implement computer systems in a simple way.

The microprocessor area have cycled the traditional computer development from simple 4-bit controllers to advanced 32-bit architectures in ten years. It has shifted the emphasis from designing computers to using computers in data processing.

This chapter is intended to cover the area of microprocessor technology, its potential usage within SDC and the savings banks and its performance compared to current TP2 technology. In this sense, SDC is comparable to any other advanced industrial enterprise today: SDC is not going to design chips, but to use off-the-shelf chips to design systems.

Also, these chips will be used in almost any decentralized system, that SDC will buy from outside vendors. As such, SDC do have access to the same information as the producers of computerized equipment and can use this information as a planning tool. SDC can furthermore use know-how in this area to evaluate whether new systems from any vendor do fit into the future plans of SDC.

3.1 Basic concepts

In this chapter, we will deal with microelectronics and its impact on systems. It will be a technical chapter, covering specialized topics where some initial knowledge is assumed by the reader. Terms in this area are often misused, so the way I use them are defined below:

Chip: is a synonym for a microelectronic circuit in a packaged form. It is also used to identify a single, separated microelectronic circuit from a wafer, but the former meaning is the most often used, and the one used here.

Component: a self-contained circuit- or system-element, which has a unique functional character, and which can be assigned a part number. As such, the component concept spans everything from a resistor or a chip to a complete computer.

Device: synonym for chip, but also generally used to denote some form of complex component.

Hardware: a general concept covering all "hard" components in a system, both electrical and mechanical. In this chapter, it is essentially components capable of guiding and controlling electric current.

IC: Integrated Circuit - synonym for chip

I/O: Input/Output - the task of transferring data from/to the environment of a computerized system. Note, that I/O is the concept of delivery of data to/from environmental destinations/sources.

LSI: Large Scale Integration - a not uniquely defined term normally identifying the art of integrating between 1000 and 10000 transistors on a single microelectronic circuit.

Microcomputer: A self-contained (i.e. including memory and I/O) data processing system based on a microprocessor as the CPU

Microelectronics: is the concept of integrating electronic circuits and functions within a single, identifiable component which is then called a microelectronic circuit.

Microprocessor: A microelectronic circuit designed to contain the fundamental part of a computer system: the Central Processing Unit (CPU) which are capable of performing control or computations

governed by a set of externally supplied instructions.

VHLSI: Very High Large Scale Integration - as VLSI, but with more than 100000 transistors on a single microelectronic circuit.

VLSI: Very Large Scale Integration - same meaning as LSI, but indicating between 10000 and 100000 transistors on a single microelectronic circuit.

Wafer: the circular block of silicon, on which several microelectronic circuits are implemented

3.2 Items to be covered.

In computer systems, some components play a major role in determining the capabilities, performance, price and size of such systems. These components include:

Single chip devices:

- microprocessors
- microprocessor support circuits
  - memory management circuits
  - arbitration controllers
  - specialized processing
  - buffers
- memories
- I/O circuits
  - data communication controllers
  - video controllers
  - magnetic support controllers
  - voice circuits

mixed or complex components:

  - floppy disk drives
  - hard disk drives
  - tape drives
  - other storage types
  - display technologies
  - printing technologies

I will specifically look at some selected parts of these components, that are of importance for the kind of systems being used (or to be used) in the savings banks. These parts will be:

Microprocessors, because these are now becoming the fundamental building blocks in our systems, and because the progress in this area goes on so rapidly, that forecasts may be difficult.

Memories, because these are the basic containers for the software part of a computer system, and because progress here are more predictable.

Communication circuits, which are needed in a communication oriented environment like the savings banks have. These circuits will however be covered separately in chapter five.

Magnetic storage devices, because these will determine the physical size of many systems.

3.3 Microprocessors

Small computer systems (i.e. excluding mainframe computers and high-end minicomputers) are today being designed around microprocessors. The microprocessor can be considered as the basic work-horse component of such a system. Mostly it will be the factor, which determines the performance limits of the system and sometimes the factor which determines the flexibility of the system.

A microprocessor is normally considered as being a single-chip circuit element, that embodies the functions of a central processing unit. This definition rules out implementations, which relies on multiple chips to obtain a central processing element, but it is the commonly agreed definition in the market. However, as we will see later, some of the newer designs of microprocessors, do actually come as multiple-chip implementations. This definition should be contrasted to the more loose definition of a microcomputer, which sometimes denotes a single-chip processing element with internal memory and I/O and in other cases a large system based on a microprocessor.

Microprocessors can be characterized by a set of certain criterias, that may not be fully unique from a academic point of view, but at least is important in the sense of the industry. These criterias are:

- basic internal word length (8/16/32 bit)
- basic external word length (8/16/32 bit)
- memory addressing capabilities
- memory management capabilities
- instruction set
- I/O structure
- interrupts and system traps
- speed

The basic internal word length is probably the most used marketing factor today and is often misused. The basic word length refers to the width of internal data paths of the microprocessor, i.e. the width of registers, the arithmetic/logic unit and interconnection between these. It should be noted, that many processors do support operations on higher word length than the internal (like 16-bit operations on 8-bit microprocessors), and some system based on 16-bit processors are indeed marketed as 32-bit systems.

As a parallel to the internal word length comes the external word length, which refers to the width of the external data bus. The data bus relationship should be emphasized, as some dp-professionals today still believes, that the word length also refers to the memory address length of a system. Also the external word length need not be the same as the internal word length, as some processors actually come in scaled-down version for a smaller bus.

The memory addressing capabilities refers to the amount of instruction and data memory, that can be physically addressed by the processor itself. This capability is often enhanced in many systems by external memory management circuits and reduced in other systems by memory mapped I/O. Also, the direct memory addressing capability is potentially duplicated in some processors by allowing separate memory spaces for instructions and data, and distinguishing between user and system modes.

Memory management is the concept of controlling and administering the resource, that the external memory constitutes. There are three reasons for memory management: (1) to enhance the direct addressing capability at the instruction level, (2) to ease the programming effort by introducing a one-to-one logical-to-physical address translation and (3) to implement virtual memory schemes, that allow large programs to

run in smaller physical memories. The difference between the first two methods is basically, that in the first the processor is cheated to support more memory, than it was originally designed for, whereas in the second method the programmer is cheated to believe that all the addressing needs starts from location zero.

The instruction set of a microprocessor is one major issue in system design, as the instruction set determines the success of implementing software functionalities. The instruction set is sometimes more or less a consequence of the basic processor architecture, but is tending to become more and more high-level software-oriented. Still the instruction set seen from the normal user is loosing importance, as user-utilization of the processor takes place through high-level languages or software-tools. However, when the processor is to be used in special environments, where the type of application is known, like in data communication or text processing applications, the instruction set can be a major factor in determining the usability of a given processor.

The I/O structure is the way, the processor interfaces to chips accessing the external world. Two basic methods exist for this: one is the availability of separate I/O addressing spaces and a fundamental set of I/O instructions in the processor to access these addressing spaces; the other is the cheap way of using memory-mapped I/O, where I/O chips are accessed as memory-locations within the memory address space. The latter method needs no special I/O instructions within the processor, as all instructions available for memory access can be used. However, this method effectively reduces the effective memory space and worse, it imposes severe reliability and maintainability problems on software drivers, as I/O cannot be distinguished from data manipulation. This is in contrast to the former method, where I/O-instructions can be made privileged operations, thus providing a built-in level of security.

The architecture of how interrupts and system traps are handled can be influencing the design of special software drivers and checks for special software conditions. This is specially important for use in controller applications, where the interrupt architecture can be the key factor of programming efforts (see Bertelsen, 1979; Bertelsen, 1982).

The speed of a processor is mainly a technological issue, being dependent on process technologies, chip sizes, circuit resolution and a lot more. Naturally, as time goes by, higher speed versions tend to come for standard chips, and interesting enough is the raw processor clock speed often a marketing point for many systems without any concern for the effective instruction rate, which is influenced by the internal instruction processing architecture, external memory management and memory technology.

Generally, most processors are available in different speed versions, enabling the designer to make a trade-off between the wanted performance level and the cost of the complete system. In order to benefit from such a speed choice, the rest of the system must however be insensible to changes in processor speed, and this is not always the case.

3.4 Standard Microprocessors

The systems, that SDC will acquire in the future, will be based on standard microprocessors for four reasons:

(1) Such processors will have the needed performance for the type of task to be carried out within the branch offices.

(2) They will be cheap and reliable due to the high volume production.

(3) The vast majority of software will be designed with these processors in mind.

(4) They will enforce a system independence between manufacturers at the basic system level.

The advances in VLSI technology will allow for more components on a single chip, and this will affect the architectural designs of microprocessors in two ways:

- microprocessors will be enhanced with features not directly related to the pure processing, like memory management, peripheral controllers, timers, and system software.

- microprocessors will be enhanced with multiprocessing capabilities at the chip level to allow for tightly coupled processors in the form of additional parallel chips.

In this paragraph, I will take a look on the existing market of microprocessor chips to illustrate the view above and indicate some potential usage within the savings banks systems.

3.5 8-bit microprocessors

The 8-bit market is dominating the micro-world today, as 8-bit microprocessors is the foundation of most embedded controller designs and the entire personal computer industry, an industry not known or anticipated in data processing seven years ago. As a matter of fact, the personal computer (PC) industry is now playing the major role in providing software tools for these processors, and the personal computer application tools will set standards for future implementations of user-oriented software. The PC industry managed to transfer the usage of 8-bit microprocessors from controller applications to business oriented applications, and this industry will also be taking the major part in the usage of the newer 16- and 32-bit designs.

Three processors dominate the 8-bit market:

- the Rockwell 6500 chip, which due to its low cost started the microcomputer "revolution", and which is the basis of the Apple® microcomputer and several small hobby computers.

- the Intel 8080, which was the basis of the CP/M™ market,

- and the Zilog Z80®, which has now taken over the 8-bit CP/M market, due to its better architecture.

Of course, both these and other microprocessors are used extensively in the more controller-oriented applications, but as far as business applications concern, the three above are the de facto standard chips. Even though the Z80 design dates back to 1976, it is today the most sold 8-bit chip available, despite the fact, that university and industry acceptance has

been slow (see Zbits, 1982). This is an example of how slow changes can penetrate an otherwise dynamic market.

With respect to performance, the Z80-based computers are reaching the limits, as new and more powerful standard software packages are coming with need for larger memory addressing. This is a reason why, the PC market is now changing towards 16-bit processors. However, in controller applications, the Z80 is still unsurpassed in most cases.

The 8-bit market will play a role throughout the century, as new versions of the 8-bit chips are coming along. These versions will sometimes compete with the 16-bit processors. The major enhancements in the 8-bit market will be:

- larger memory addressing capability
- removal of instruction set deficiencies
- I/O controllers on the processor chip
- redundant processors on the same chip

Two current examples can be mentioned to illustrate this:

3.5.1 Rockwell 65C00/20

One is the new 65C00/20 dual-processor chip from Rockwell, where two parallel processors are implemented on the same chip, sharing the bus interface (see Bigelow, 1983). These processors run independently in a multi-processor fashion, each with one-half the chips clock speed. The major benefit of this approach comes in software development, where one chip will control for example two conceptually different types of applications, like one controlling a communications line and the other a video display. One benefit is, that the designer is not concerned with problems of multiprocessor design on the circuit boards, as the control of this is carried out within the chip.

3.5.2 Zilog Z800

The other is the Z800™ chip from Zilog, which will have a major impact on the PC-market (see Whitcomb, 1982; Carter, 1983). This chip is first completely instruction compatible with the old Z80 chip, meaning that existing programs can run without recompilation. Secondly it has implemented a set of new instructions and addressing modes, which have long been needed for high-level applications running on the Z80. Finally, the memory addressing capability is enhanced up to 16 Mbyte through an internal memory management unit, and to increase the speed of the microprocessor chip, it will have an internal cache memory of 256 byte for high speed instruction and data fetch in repetitive loops.

Using it with its full capabilities, it runs in system and user states with a range of privileged operations, enhancing software reliability and security. Memory management allows for instruction restart, a feature needed for virtual memory applications. In short, the concepts known from mainframe computers are now being implemented in the 8-bit micro-world.

3.6 16-bit microprocessors

The 16-bit market is currently the most competitive in the sense of announcing new products and new concepts. The availability of 16-bit microprocessors has in the last three years made a major impact on the 16-bit traditional minicomputer market, which although still present is loosing its market share (see Blundell, 1982). This is due to two types of usage of the 16-bit processors: one in the area of single-user personal computers and the second in the area of the so-called "super-micros" - microcomputers, supporting multiple users, with minicomputer performance (see Isaak, 1982; Hui, 1983; Serlin, 1983).

Whereas the 8-bit market is used in a variety of applications, ranging from controller functions to administrative data processing, the 16-bit market is mainly oriented towards what can be called the business oriented type of computing. This trend is also dominating the future development of such processors.

With the advent of 16-bit processors, the industry became aware of the problems of software and hardware migration to new types of processors. This have also driven the industry to pay more attention on high-level languages and operating systems, which again affects the actual processor designs.

The current 16-bit market is dominated by three processor families: the Intel 8086, the Motorola 68000 and the Zilog Z8000. All these families will have interest for SDC, because future systems, that SDC will buy from outside vendors in the next two to four years, will be based on these types of processors.

3.6.1 Intel 8086

The 8086 was the first "real" 16-bit microprocessor to become available. By "real" is meant, that some 16-bit designs were available earlier, but these were based on old-fashioned architectures or replica of minicomputer designs, and have never played any significant role in the business oriented area. Examples of these are the 9900 from Texas Instruments and the 9450 from Fairchild.

8086 is by many considered to be an enhanced 8080 8-bit processor and is the lowest performance 16-bit processor available. Nevertheless, it has been adopted in many design, and is probably the most sold 16-bit processor today. The reason is twofold: First, it was the first and it came from Intel, a fact that gave security to non-experienced designers. Secondly, it allowed the usage of 8080-type of peripheral chips and the usage of the MULTIBUS™ bus standard, meaning that existing controller boards could be utilized in 16-bit systems (see Noyce, 1981). Finally, it allowed existing 8080-software to migrate to the 16-bit world.

Other facts could be, that Intel is one of the few companies that have realized the need for software support at an early stage in the design process, and that most of the industry at the time of introduction were actually users of the Intel 8080 and 8085 8-bit micros.

The 8086 has 4 internal 16-bit registers, which can be mapped as 8080-type registers. It allows addressing up to 1 Mbyte, but since the internal word length is only 16 bits, the 20 bit address is generated by using a set of 16-bit segment registers, which act as pointers to 16-byte boundaries in memory, and which are added with a four-bit offset in address computations. As such, the 8086 resembles the way, the TP2

S1000 local computer accesses memory (see OL, 1981.6).

To increase performance, the 8086 has a pipelined architecture. Actually, there are two processors on the chip, one which handles the execution of instructions and one which takes care of the bus communication to the external components (see Alexandridis, 1980).

The 8086 family has been augmented by the 8-bit bus version 8088, which internally is an 8086, but with an external 8-bit bus. The 8088 has been the entry point in the personal computer market, as it can coexist with 8-bit processors in an 8-bit system.

3.6.2 Intel 80186

Today, the 8086 has no interest in itself for serious system designers. The impact from 8086 comes in the newer members of the family, the 80186 and the 80286. The 80186 is primarily an enhanced 8086, where some of the oddities of 8086 has been removed. The 80186 has a performance improvement of 2-3 times the 8086 at same clock speeds. This is mainly due to improved instruction time, like effective address calculation, which is done by microcode in the 8086, but by a special hardware adder in 80186. 16-bit arithmetic is performed faster, and so is an important thing like block move, which now runs at bus bandwidth. The other major concept in the 80186 design is the inclusion of peripheral functions on the chip. These functions include a clock generator, interrupt controller, DMA controller and a local bus controller, all of which had to be implemented by external chips in previous designs. Perhaps most interesting is the bus controller, which directly enforces the logical architecture of a local on-board bus separated from a global backplane bus (see Shoemaker, 1983).

The 80186 improvements also indicate the directions, that VLSI design are taking: VLSI allows for more gates directly on the chip, but these gates cannot be used just to increase the basic processor with more speed, more registers or more instructions. Instead they are used to augment the basic processor with vital peripheral function, and is thus a step towards the complete computer on one chip.

3.6.3 Intel 80286

The Intel 80286 is the next step in this direction (see Intel, 1982). Where the 8086 and the 80186 are still plain processors, where a given program is capable of executing any instruction in any given memory location, that the processor has access to, the 80286 introduces the concept of memory management and protection. Besides being faster than the 80186, up to a factor of 2, it now allows for larger memory addressing up to 16 Mbytes.

Furthermore, virtual memory and protection is provided by allowing virtual address spaces for individual tasks up to 1 Gigabyte. Within each virtual address space, a four level ring protection scheme is implemented. Also, the processor has integrated support for operating systems, like task switching and operating system calls. As such, it denotes a departure from the original 8080-origin of the 8086 and with these facilities and its performance it will impose a real threat against the Motorola 68000 (see Intel, 1983.2), especially as users begin to place more emphasis on software security (see Schell, 1983).

3.6.4 Motorola 68000

The Motorola 68000 is a different bid in processor design and the name comes from the fact, that it has approximately 68000 transistors on the chip. The 68000 is internally designed with 32-bit registers and 32-bit data paths, and physical addressing goes up to 16 Mbyte. Addressing is uniform in contrast to the segmented type of addressing used on the 8086 and the Z8000. These features together with an intensive marketing approach has caused the 68000 to take the major market share for the high-end microcomputer applications.

The 68000 chip contains a register set of 8 32-bit data registers and 9 32-bit address registers in contrast to other processors like 8086 and Z8000, where registers are general and can be used for both address and data. In 68000, the registers have been split due to performance reasons, since address and data calculations can be overlapped with separate register sets.

As the 68000 internally uses 32-bit registers, it could be considered as a 32-bit processor, in contrast to the 8086 and Z8000, which have 16-bit register sets. It is anyhow commonly categorized in the 16-bit area.

The instruction set has been kept small with 56 basic instructions, expandable through several addressing modes. I/O is memory-mapped as was the case with the 8-bit 6800 microprocessor, although a few special instructions exists for handling I/O towards 6800-oriented peripheral chips.

The address and data buses are separate, with 16 data lines and 24 address lines, causing the need for the large 64-pin dual-in-line package.

Newer members of the 68000 family has come on the market since the first introduction. The 68008 processor has an 8-bit external data bus and 20-bit address bus, restricting its addressability to 1 Mbyte, but with the same instruction set as the 68000 (see Stockton, 1983). Of more interest is the 68010 virtual machine, which is an enhanced 68000 with the ability to continue instruction execution after a memory fault, thus allowing virtual memory (see Stockton, 1982). The 68010 also provides possibility for designing virtual machine system, where several operating systems can operate concurrently. The 68010 works together with the 68451 memory management unit, which allows for demand paging memory systems.

Another enhancement of the 68000 has been the MK68200 processor from Mostek, where an asynchronous serial controller, RAM, ROM, timers, parallel I/O and bus arbitration circuits have been added to the 68000 architecture on the same chip, thus effectively having a single-chip 16-bit computer with internal 32-bit paths (see Folkes, 1983).

3.6.5 Zilog Z8000

The Zilog Z8000™ series was introduced approximately at the same time as the 68000, but has been slower in gaining its market share (see Shima, 1978). This late adaptation was due to many factors, including production and late introduction of the support circuits. Also, at the point of introduction, the Z8000 and the concepts it included, was probably too advanced to get acknowledged in the user community, which then hardly had grasped the impact of the 8080.

The Z8000 series is special in many ways: it introduces the concept of segmented memory addressing in a way different from the 8086. In the 8086, direct addressing is only 64 Kbyte, and the enhanced 1 Mbyte addressing is achieved through the addition of a segment register, which has to be preloaded before the addressing takes place. In the Z8000, instructions can directly address any location by having the segment information resident in the instruction field. This, however, necessitates one extra memory cycle. Also, in the 8086, the external addresses are 20 bits directly, whereas in the Z8000, addresses are 16 bit plus 7 bit segment information, so segments can be protected by external devices, like memory management units.

The most important features of the Z8000 are the concept of a general set of registers, usable for both data and address values, and an entirely new type of instruction set (see Peuto, 1979). As such, it is not compatible with Zilog's previous Z80 line.

The Z8000 have these features:

- System/User modes of operation with privileged instructions in system mode. This is important for making reliable software, where certain parts of the software, e.g. the operating system, must not be accessible to user programs.

- Direct I/O addressing, which are privileged operations, which is needed for making structured and reliable software.

- General register set of 16 16-bit registers, which can be accessed as 8-bit, 16-bit, 32-bit or 64-bit registers. This could be seen in contrast to the 68000 processor, where only the lower part of the 16-bit registers can be accessed as 16-bit or 8-bit. This have significance in many applications, like floating point arithmetic, text processing and communication, where byte swapping often is a necessity.

The Z8000 comes in two version, the non-segmented Z8002 and the segmented Z8001. Memory management is handled by connecting one or more Z8010 memory management units to the Z8001.

Like with Motorola, the Z8000 family has been enhanced with two new processors, the Z8003 and Z8004, which allows for instruction restart in the case of a memory fault. This is needed for implementing virtual memory systems, which are supported by the Z8015 demand-paged memory management unit (see Mateosian, 1983).

Compared to the 8086 and the 68000, the Z8000 is by far the best performer (see Patstone, 1981; Prycker, 1983).

3.7 Comparison with TP2

The following figures show the performance of some of todays microprocessors compared to the current TP2 system:

	TP2 S-1000	8086	Z8000
clock speed		8 MHz	8 MHz
addressing	1 Mbyte	1 Mbyte	8 Mbyte†
(with MMU)	n/a	n/a	16 Mbyte
add	0.75††	0.375	0.5
branch	3.45	0.875	1.0
call	>3.45	2.5	1.875

(†: 4 address spaces of 8 Mbyte)
(††: all figures in microseconds)

These figures does not give the full truth of the performance relationship between these processors. One important factor is the nature of the application. The decentralized SDC applications are not computational oriented, but rather control and data movement oriented. This, together with the fact, that the applications are written in high level language, emphasizes a major use of branching and procedure calls, which are slow on the TP2 S1000 local computer. Also the register-oriented architectures of the 8086, the Z8000 and the 68000 facilitates conditional branching, especially in CASE-statements, where a value has to be compared against several alternatives, where the S1000 has to perform a repeated reload of these values on the stack, as compare-instructions destroys the values being compared. Thus application code will execute significantly faster on the microprocessors than can be anticipated from the figures above.

A loose estimate will judge the 8 MHz 8086 to have about 2-3 times the power of the S1000. The Z8000 is about twice the power of the 8086 and 10-20% faster than the 68000 at the same clock speeds (see Prycker, 1983). One should also note, that all of the microprocessors mentioned are available in higher speed versions than 8 MHz.

The terminal computer of TP2 has the same problem. It is based on the Z80 microprocessor, running with 2.5 MHz clock speed, and with no provisions for enhancing that clock speed. No systems on the market today (or for the last 3 years) using Z80 run with less than 4 MHz. New designs today use the 6 or 8 MHz versions of the Z80.

This is the reason, why the TP2 system is technologically outdated. The sad story is, that these new microprocessors were announced and available long before the first prototype of the TP2 system was available. Today, the TP2 system is an example of the unchangeable design of the 70'ies. As it stands, it cannot survive the needs from the savings banks in the 80'ies, especially not when the commercial banks begin to install new decentralized systems based on these and the coming microprocessors.

3.8 32-bit microprocessors

Even before the 16-bit market has begun to top, the 32-bit microprocessors are now beginning to emerge. Some of these have been available for some time, whereas some still are awaiting the implementation on silicon.

Along with the introduction of 32-bit microprocessors, we also see a trend towards a usage that falls within the traditional mainframe area.

The chips in the 32-bit market will both replace existing 16-bit designs and go into new application fields. This is also reflected in the different trends and designs, that are known today. These trends include:

a) 32 bit extensions of existing architectures with an internal word length of 32 bit, but presently using 16 bit external word length.

b) 32 bit migrations of existing architecture, where current 16-processors are enhanced with new concepts and instructions to support 32-bit operations.

c) 32 bit designs, which implements a traditional processing unit, but without linkage to existing design.

d) 32 bit design, implementing new architectural concepts.

3.8.1 Extensions of existing architectures

As representatives of trend a), we have the Motorola 68000 and the National Semiconductor MS16000 series of microprocessors.

3.8.1.1 Motorola 68020

The Motorola extension will come as the MC68020 chip with an external data bus of 32-bit. The internal architecture of this processor will be primarily the same as the 68000, except that it will include a cache memory for instructions (see Stockton, 1983).

3.8.1.2 National Semiconductor 16032

The National Semiconductor NS16000™ series of microprocessors consists of a family of processors with the same internal architecture and the same instruction set, but with different data bus width (see Lavi, 1980; Ashkenazi, 1981; Bal 1982). Even though the design dates back to 1980, it has just recently been available in quantities and is current subject to aggressive marketing efforts to get a piece of the market.

The NS16032 is the competing device at the moment, which is the 32-bit processor with an external 16-bit bus. Although the NS16032 could be considered as a 16-bit device, like the Motorola 68000, it is normally classified as a 32-bit device, and this is the reason why it is included in this paragraph. In 1984, the NS32032 will come on the market with a 32-bit external bus width, but with 24 bit addressing. It is expected that the NS32132 with full 32-bit addressing will come in 1985.

3.8 32-bit microprocessors

Even before the 16-bit market has begun to top, the 32-bit microprocessors are now beginning to emerge. Some of these have been available for some time, whereas some still are awaiting the implementation on silicon.

Along with the introduction of 32-bit microprocessors, we also see a trend towards a usage that falls within the traditional mainframe area.

The chips in the 32-bit market will both replace existing 16-bit designs and go into new application fields. This is also reflected in the different trends and designs, that are known today. These trends include:

a) 32 bit extensions of existing architectures with an internal word length of 32 bit, but presently using 16 bit external word length.

b) 32 bit migrations of existing architecture, where current 16-processors are enhanced with new concepts and instructions to support 32-bit operations.

c) 32 bit designs, which implements a traditional processing unit, but without linkage to existing design.

d) 32 bit design, implementing new architectural concepts.

3.8.1 Extensions of existing architectures

As representatives of trend a), we have the Motorola 68000 and the National Semiconductor MS16000 series of microprocessors.

3.8.1.1 Motorola 68020

The Motorola extension will come as the MC68020 chip with an external data bus of 32-bit. The internal architecture of this processor will be primarily the same as the 68000, except that it will include a cache memory for instructions (see Stockton, 1983).

3.8.1.2 National Semiconductor 16032

The National Semiconductor NS16000™ series of microprocessors consists of a family of processors with the same internal architecture and the same instruction set, but with different data bus width (see Lavi, 1980; Ashkenazi, 1981; Bal 1982). Even though the design dates back to 1980, it has just recently been available in quantities and is current subject to aggressive marketing efforts to get a piece of the market.

The NS16032 is the competing device at the moment, which is the 32-bit processor with an external 16-bit bus. Although the NS16032 could be considered as a 16-bit device, like the Motorola 68000, it is normally classified as a 32-bit device, and this is the reason why it is included in this paragraph. In 1984, the NS32032 will come on the market with a 32-bit external bus width, but with 24 bit addressing. It is expected that the NS32132 with full 32-bit addressing will come in 1985.

Like the Z8000 and MC68000 processors, the NS16032 is reliant on an external memory management unit to support relocation and virtual memory. This unit is the NS16082 MMU (see Lavi, 1980), which support demand-paged virtual memory management as well as support for virtual machines.

3.8.2 Migrations from 16-bit processors

As exponents for trend b) can be taken the Intel 80386 and the Zilog Z80000. These will be true 32-bit processors, based on existing 16-bit designs.

3.8.2.1 Intel 80386

The Intel 80386 will offer some compatibility at the software level with the 80186 and the 80286 processors. It will also come with a 32-bit external data and address bus. It will include memory management and cache storage.

3.8.2.2 Zilog Z80000

The Zilog Z80000™ will also offer a full external 32-bit bus width for data and addresses (see Alpert, 1983). It will be an enhancement of the Z8000 with more registers and new instructions to support high level languages. Again, this chip will contain on-chip memory management and a 256 bytes cache storage for both data and instructions. Address representation will be programmable to support the problems of managing the large address space (see Bertelsen, 1980), as two forms of segmented addressing is possible, as well as linear addressing (see Zilog, 1983).

It will contain 16 32-bit registers, as well as a virtual memory management system, that uses a paging translation scheme (see Bursky, 1983). Memory management also includes an access protection scheme, that allows access rights to be checked at different levels of the addressing computation (see Zilog, 1983). This will further enforce the trends towards software security.

The initial version of the Z80000 will use a 10 MHz clock, capable of executing more than 1.5 million instructions per second (MIPS), although a 25 MHz version capable of 2.5 to 3.7 MIPS will be available later. The Z80000 chip will also be able to support both 16-bit and 32-bit external buses, thus eliminating the need for several processor versions for different buses. This further makes it possible to design a 32-bit processor board with on-board memory connected through a 32-bit bus, while at the same time interfacing to an older type 16-bit backplane bus.

3.8.3 New 32-bit designs

In the area of completely new designs of true 32-bit traditional microprocessors, three types can be mentioned: The Bellmac-32A, the Hewlett Packard 32-bit chip and the NCR/32-000 chip. These can be called traditional, as they all more or less architecturally resemble the hitherto mentioned microprocessor chips, but they are also new, as they neither do have any 8-bit or 16-bit predecessor nor is compatible with any known design.

Until now, they have been internal designs for specific purposes within the designing companies, and it can be questioned, whether they will have any impact on the market or whether they will be released for OEM usage.

3.8.3.1 Bellmac-32A

The Bellmac-32A has been designed for internal use by Bell Laboratories (see Gupta, 1983). It is a fast processor, designed to provide support for the C language.

3.8.3.2 HP 32-bit chip

The HP 32-bit chip is designed for internal usage in HP-products (see Gupta, 1983). It has been a subject for immense writing, as it currently has the highest integration density of any microprocessor with 450000 transistors on the chip.

3.8.3.3 NCR/32 chip

The NCR/32-000 central processor chip is the newest member of 32-bit chips, designed for internal use in NCR's own systems, of which the first, NCR9300, has already been announced. This chip with only 40000 transistors is microcoded and does even include support for emulation of the IBM System 370. The processor chip works together with the NCR/32-010 Address Translation chip that includes memory management and error detection/correction facilities.

Interface to the external world takes place through the NCR/32-500 system interface controller, which provides a message-based communication concept between the main memory and two 8-bit buses. These two buses connects to the NCR/32-580 system interface transmitter and the NCR/32-590 system interface receiver, which converts the parallel bus to a serial local network, using a CSMA/CD based protocol with a data rate of 24 Mbit/second.

3.8.4 New architectures

By new architectures is considered implementations, that do not adhere to previous concepts of computer design, i.e. the register oriented architectures of the processors described above.

3.8.4.1 Intel iAPX 432

The Intel iAPX432™ represents the last type of trends, i.e. new architectural concept (see Hemenway, 1981). The iAPX 432 does not fall into the previous definition of a microprocessor, as it is a three-chip set. The iAPX 43201 instruction decoding chip and the iAPX 43202 instruction execution chip constitute the 432 General Data Processor (GDP). Another chip, the iAPX I/O interface processor, handles the link to the external environment, as the 432 processor system will only handle pure processing and not I/O. The three chips are linked together through a processor-memory interconnect bus, that allows several GDPs to be attached to the bus (see Jurich, 1981).

The 432 system has been designed to support ADA, using an object-based architecture, and has built-in operating system primitives, that handles process creation and intercommunication (see Intel, 1981; Ziegler, 1981). Also, these primitives includes the ability for a system with several GDPs to automatically dispatch processes for execution on a free processor, and as such, this has been the first design to implement a multiprocessor building block.

The 432 allows a physical address space of 16 Mbyte. Logical addressing, however, allows 1024 Gbytes, with the restriction that the logical addressing capability of a given program "only" is 4 Gigabytes.

Besides this, the 432 has implemented facilities for redundancy, as one GDP can monitor another and take over processing at any time. This facility is aided by two other components, the iAPX 43204 bus interface unit and the iAPX 43205 memory control unit, which include the circuitry needed to detect failures and switch to redundant processors, memory or buses.

3.8.5 32-bit microprocessor applications

The prime usage of these new chips will be in the so called "super-microcomputers", which is business oriented multi-user systems. However, with the decreasing price of components, we will find them in single-user systems and personal computers before the end of this decade.

In terms of processing power, they will be an ideal candidate for systems, that now meet the boundaries of current 16-bit microcomputer designs. Still, performance figures for the newer types of these processors can be hard to get, especially as performance can depend on many factors, like addressing schemes, pipelining and the usage of new features, like operating system primitives.

Anyhow, the performance of these chips go into the area of previous mainframes. The Intel 80286 and the higher speed versions of the Motorola 68000 perform better than the VAX-11/780, one of the high-end performance references in the minicomputer world, the Intel 432 is superior to an IBM 370/148, the Bellmac-32A and the HP 32-bit chip are expected to be superior to an IBM 370/158 (see Gupta, 1983).

The 16-bit microprocessors were found to be better than the local computer in TP2. The new 16-processors will be even better and the 32-bit microprocessors will constitute migrations from these in the years to come. As such, SDC will have a wide range of processing options to choose from, when selecting processing power for the savings banks. Whether we will be able to do that depends on many things: the software, the operating system, the surrounding hardware, the general architecture of the computer. However, as these chips will be cheap, they may penetrate even the smaller systems sooner than anticipated (see Chapter 10).

3.9 Support circuits

Along with the processor development, other components follows the progress in technology. These components comes in two forms: as I/O devices and as coprocessors. I/O devices may contain their own kind of processor and are generally programmable to serve a set of specific I/O requirements. Coprocessors, on the other hand, are designed to work closely together with the microprocessor chip, where they act as an extension of the basic instruction set of the processor.

3.9.1 I/O circuits

Almost any processor family has a set of suitable I/O devices, that can work with that specific processor. But these I/O devices can in fact work with almost any other processor, when the needed interface logic are added. Of special interest here is the various types of communication controllers, which implements the logical link interfaces needed in different communications environments, like X.25 networks and local area networks. Another part of interest for SDC will be the encryption chips, which implements the standard NBS DES

encryption/decryption algorithm.

3.9.2 Coprocessors

Coprocessors will normally only be used with the microprocessor, it has been designed to support. What coprocessors does, is to enhance the instruction set of the processor by doing specialized computations separately in hardware. A system without a coprocessor will normally perform these computations in software, so the processor does not enhance the functionalities of a system, but it increase the performance. Standards are an important issue in coprocessor design, because the concept of coprocessors implies, that any application software should never detect the presence of a coprocessor or not. This means, that software implementing the same operations as the coprocessor, should always give the same results.

The most used candidate for coprocessor design is the ability to support floating-point arithmetic, and most microprocessors do have or will have a floating point chip. Traditionally, floating point operations has been a vendor specific issue, where operations could be implemented differently, dependent on technologies and methods used. For most programs, this will have no effect, but for those applications, where rounding errors and changes in the least significant digits may be of importance, problems may occur (see Cody, 1981). One example could be a formula to iterative compute a rate of interest over a longer period of years, which may have been checked and audited by SDC dp-auditing on one system, but which produces wrong results on another system.

Such phenomena are precluded by the usage of standards and all floating-point arithmetic capabilities in the microcomputer world today follows (or should follow) the IEEE 754 standard (see Stevenson, 1981). Also, most of the floating point

coprocessors do support conversions to/from BCD-arithmetic, which is the prime representation used by SDC. However, these coprocessors uses another size than the one, that SDC has chosen. As long as SDC stays with the current TP2 system, this should be no problem, but it will become a problem as soon as the savings banks start the more intensive use of personal computers and supermicros for numeric analysis or APL-programming. It may be worth considerable in the future for SDC to settle for a standard number representation and computation, at least locally in the new processor systems, both to have control of numeric results and to gain advantage of these new types of coprocessors.

All of the 16-bit microprocessor families are supported by floating-point coprocessors, like the Intel 8087, the Zilog Z8070, the Motorola MC68881 and the National Semiconductor NS16081.

Other coprocessors types will be text coprocessors and graphics coprocessors, although these still are in a premature state.

Another interesting development in this area is the concept of operating system chips, where Intel has announced two coprocessor chips, called Operating System Processors, OSPs (see Evanczuk, 1983.2). They are the 80130 with the Intel iRMX™ operating system (see Heider, 1982) and the 80150 with the CP/M™ operating system implemented on the chip (see Noice, 1983). First these chips ease system design and they (at least yet) free the user from paying any royalties or licensing fees for use of the operating system (see Lettieri, 1982).

3.10 Memories

The progress of technology has also influenced the memory area, where higher and higher densities becomes available. These higher densities facilitates the design of smaller systems and the usage of the memory addressing capabilities in todays microprocessors.

The memory market is large and offers a wide range of different memory types for different applications. What will be of interest here, is the type of memories intended to serve the microprocessor market. Even here, several options are possible, but only a few of interest will mentioned here.

The dynamic RAM memory is the cornerstone of microcomputer design. These chips offer a relative high density together with acceptable high speeds. The commonly used type of chip today is the 64 Kbit dynamic RAM, although the 256 Kbit dynamic RAM is just about to be available in quantities. Use of these circuit densities futhermore facilitates a design concept, where all the memory, a processor needs, are present on the same circuit board. Designing with 256 Kbit dynamic RAMs allows one megabyte of memory on the same space as 8 lines of this reports paper width.

As such, the developments in memory design are foreseeable. What will be of interest for SDC is both the possibility of designing self-contained boards with the needed processor, memory and I/O and also the possibility of using the small space requirements for adding extra chips, so memory error detection and possibly correction can be implemented. Such features are also made possible by the various VLSI-components, which implements these detection/correction circuits. The result will be systems with a larger inherent reliability seen from the user's point of view. Reliability will also be

enhanced as the memory chips eventually will be equipped with redundant cells, that can replace bad cells.

Another type of memories of interest for SDC are the non-volatile memories, that are now available. Non-volatility means, that the memory keeps its content, even if the power is off. This facility is critical for SDC, where teller counters have to be maintained in the case of a power failure. In the current TP2 system, terminal computers have a small area of CMOS memory with battery backup, which can be used for that purpose, but the battery backup is only able to keep the information for a couple of hours. Usually this is no problem, as power will normally be restored within that time.

3.11 Peripheral components

Peripherals are constituting a market of the same dimensions as that of the semiconductors. What I will shortly overview here is the market of storage media, which normally can be considered as being internal to a given computer.

These peripherals come in several types, but the SDC prime interest lies in the magnetic media area, with floppy disks, hard disks and cartridge tape drives.

3.11.1 Floppy disks

The floppy disk market has gained an enormous popularity because of the advent of the personal computer market. Actually, the combination of the first microcomputer (originally using cassette tape as secondary storage) and a floppy disk created the product, which can be characterized as the business personal computer, that started the penetration into the high-volume business market. As floppy disk has primarily been used with small computers, there has been an enforcement within the industry to make the physical size of

these drives smaller, rather than using the technology to obtain more storage on existing drives. The wide-spread use of floppy disks amongst non-computer people has also driven the industry towards designing a more handy format of the disk.

The 8-inch floppy disk market is now beginning loose its market share to the smaller type of drives, although many of these smaller drives still contains less information on each disk. Even though this trend will continue, the 8-inch market will still have a mission for some years, as it is the only place, where a suitable standard currently exists for formatting floppy disks, namely the IBM 3740 format. This format is the standard format for the entire CP/M market on 8" floppies, although it only is able to contain 256 Kbytes. The dual density, dual side IBM system/34 format are normally used for higher densities, with more than 1 Mbyte per diskette, but here standards are spare, as factors like sector offsets and sector sizes vary amongst vendors.

The 5-1/4 inch market is a mess. Almost every vendor uses a proprietary type of format on the disk and not even a sort of de facto standard exists. Lately, however, a standard proposal has come underway for personal computers using the MS-DOS operating system to obtain a standard exchange format for floppies generated by that operating system.

The sub-4 inch floppy market is new, and still standardization is in progress. At least, the major manufactures has agreed on choosing a standard formatting procedure for these small disks, but the fight is still going on to define the ultimate size of the actual floppy. Currently there exists both 3", 3-1/2" (see Jarrett, 1983) and 4" floppies (see Herald, 1983) and corresponding drives. Despite this, the sub-4" market will quickly take over the 5-1/4" market for two reasons: the floppies will be designed to hold the same information as the larger drives, and the small floppies comes in a better package

with a hard envelope and a self-closing door over the read/write-area of the floppy. As such, these floppies are much better suited for informal usage and transportation, and the hard envelope has indeed been designed so it fits into a shirt-pocket without problems.

The current policy in SDC is to keep magnetic media handling away as much as possible from the employees in the savings banks. This is a valid policy, when the larger and more vulnerable floppy disks are used, especially in the case of Olivetti, where the floppy disk drives are extremely sensible to the disk quality (a factor, which is virtually unknown in the rest of the personal computer market, where floppy disks are used and copied by millions every day without problems). With the new sub-4" products, this policy may eventually change, as they will represent a lesser problem to the employees than having the badge card needed to access the TP2 system.

3.11.2 Hard disks

The same trend is valid in the hard disk market, where larger storage sizes are being implemented in smaller physical sizes. Together with the interface standards for such drives, the SMD and ANSI buses (see Parker, 1983), these drives will be the backbone of most 16- and 32-bit microcomputer systems, which will need the storage capacity for the larger application programs and possibly for virtual memory schemes (see Jacob, 1983).

Generally, the hard disk market for these types of applications follows the floppy disk market in form factor, so hard disk will fit into the same size of place as corresponding sizes of floppies (see Warren, 1983). What has lacked behind, has been the size of controller boards for interfacing the drive to the computers backplane bus, but even these boards are now

shrinking due to availability of more integrated components. This again lowers cost, and due to the high volume market in the personal computer area, the total cost of a disk system will decrease. As such, it is expected, that the cost for e.g. a 50 Mbyte hard disk for the PC market will soon come significantly below 500$.

Again, this trend will allow magnetic storage media to be integrated in almost every computer system, that can be anticipated in SDC's branch office systems.

3.11.3 Cartridge tapes

The developments of the smaller hard disks have necessitated a new type of peripherals: the back-up tape. With 5-1/4" and 8" hard disks with capacities between 20 and 100 Mbytes, back-up cannot be done properly in a business environment with floppy disks, and even though the disk drives are becoming more reliable, back-up is needed.

If we take the TP2 system with 1500 installation, each with a hard disk, and estimate the mean-time-between-failure for these hard disks to be 10000 hours, then with a daily operating time of 10 hours, 1.5 disk units will on the average fail per day in that system. Hence, back-up is needed, and this back-up must be fast. Even if only 10 minutes are used per day for back-up, this amounts to 250 hours or more than 1.5 man-month per day, that will be used by personnel in the branch offices.

The standard media for this type of back-up is the 1/4" tape cartridge from 3M. Several types of drives exist for this cartridge, but again standardization lacks. Two basic types of drives exists: those running in streaming mode, where the tape runs at constant speed, while data is being transferred, and those running in start/stop mode, where the tape can be stopped between data transfers. Streaming mode is useful for complete

disk back-ups, where complete tracks are transferred directly to the tape. Start/stop mode has higher performance in selective file back-up, where files dispersed over random disk tracks are to be transferred or where the I/O system of the computer cannot keep pace with the required data rate of a streaming tape.

Still standards are lacking in this field, although serious attempts are now being made to define a standard for drive compatibility, so tapes written on one manufactures drive can be read by another. Two standard proposals exist: the QIC-02 which constitutes the interface towards the drive and the associated commands, and the QIC-24 magnetic format, which defines the physical encoding of the magnetic media. However, the QIC-02 and the QIC-24 proposals allow so many options, that total compatibility between different manufactures drives cannot be guaranteed at the moment (see Aseo, 1983.2).

3.12 Conclusions

This chapter has described some of the evolution with the area of microelectronics. The lesson is, that current microprocessors will provide larger processing performance, than can be found it the current TP2 system. Also, these processors are cheap and surrounding components follow the same trend. Hence, systems with equal processor power tend to be physically smaller, and system retaining their physical size can be equipped with more power. Since the processors are cheap, the traditional viewpoint of having to optimize the use of a single processor disappears. Instead, processors will be used locally at different places within a system, and circuit boards will be designed to embody a self-contained computer.

Hence, the problems in the future will not be how to optimize a single processor, but to figure out, how several processors can work together and intercommunicate. This will be the subject of the following chapters.

This page intentionally left blank

CHAPTER FOUR

ARCHITECTURAL ELEMENTS

4.0 Introduction

This report is a step towards specifying a system architecture for the next generation of the savings banks' branch office systems. The preceding chapter was concentrated around the progress of microelectronics, and we saw how off-the-shelf microprocessors outperform the current TP2-system.

TP2 is an example of a system in which several processors work together and interact. In future systems, we will find significantly more processors, due to their low cost, immediate availability and inherent performance. Although they may carry out self-contained tasks, they need to interact. The way, this interaction is performed, is an important part of the system architecture. A fundamental overview of concepts involved in systems with multiple processors will hence be included.

This chapter will also give a brief overview around what is happening on the related market, i.e. the market producing products similar to those used in the TP2 system. My reason for doing this is, that such market elements will constitute a major part of any future system, which can be conceived in the savings banks. Hence, disregarding the impact of the market may lead to theoretically interesting architectures of probably little interest or benefit for the savings banks and SDC.

Data processing covers an enormous field of products, concepts and disciplines and in some cases, even the simplest product embodies a little grain of each. This also makes it difficult to produce an unbiased view of a product, because most people tend to pay emphasis on the special corners of data processing, that he or she knows best. This, however, is also the reason for the wide variety of products being designed. The same problems have been present with data processing for years: performance, reliability, flexibility, coding, program transportability, quality control, standardization, and thousands of well analyzed solutions have been defined.

Almost any company or expert can provide a suggested solution for TP2 problems and a concept for a TP3 system. Still, one thing is sure: most of these concepts can be used and most will have problems. The same holds true for what is presented in this report. The basic problem for a user will anyway be to select a system giving most benefits and least troubles.

I believe, that the benefits in future systems design for the savings banks will not come from pure technical analysis. They will come from the ease of using and changing the system. This ease and flexibility within the system will be inherently coupled to the way, that system components and processors interact and the following paragraphs will cover this subject.

4.1 System architectures

A system architecture describes a system by defining the individual components within the system and the interconnection between these components. As such, the system architecture can be more or less detailed and it can concentrate on specific parts of an entire system, leaving the other parts free.

System architectures are needed whenever several individual components have to interact. Hence one can say, that almost every known type of computer system does obey some form of concept, that implements a system architecture.

Generally, system architectures come to their virtue in the case, where several processing elements will communicate. Previously, many systems were implemented around one single processor, which carried out the computational tasks for several users. Indeed there has been good reasons for this. This central computer was often expensive, and hence it had to be shared by as many as possible. Secondly, there was some truth in the famous Grosch's Law, that suggested that processor performance was proportional to the square of the cost of the processor. This law directly discouraged any design, in which several processors would cooperate to perform a given task.

The microprocessor technology have changed this. The cost of a processor is now becoming minimal relative to other components of a total system, and hence considerable interest have grown around the concepts embodying the use of multiple processors in one system.

These multiple processor systems have applications in many different areas: they may be used to perform tasks, that are too big for one single processor; they may individually run separate tasks, that previous was performed in a multiprogramming or multitasking environment on one single computer and they may be used to allow more processing power per user for a common type of task, that were previously shared on one single processor.

4.2 Multiple processor classifications

One of the basic classifications of multiple processor system is centered around how instructions and data flow occur within the system, and it embraces almost any kind of conceivable way of designing such systems (see Enslow, 1974; Fathi, 1983).

There are four major classification criterias:

- **SISD Single-Instruction Single-Data** stream. This type is the classical von-Neumann computer, which executes instructions sequentially.

- **MISD Multiple-Instruction Single-Data** stream.

- **SIMD Single-Instruction Multiple-Data** stream, which are typically vector or array processors.

- **MIMD Multiple-Instruction Multiple-Data** stream, which is a very general classification, which can be sub-classified dependent on the type of interconnection method.

The MIMD systems can be divided into:

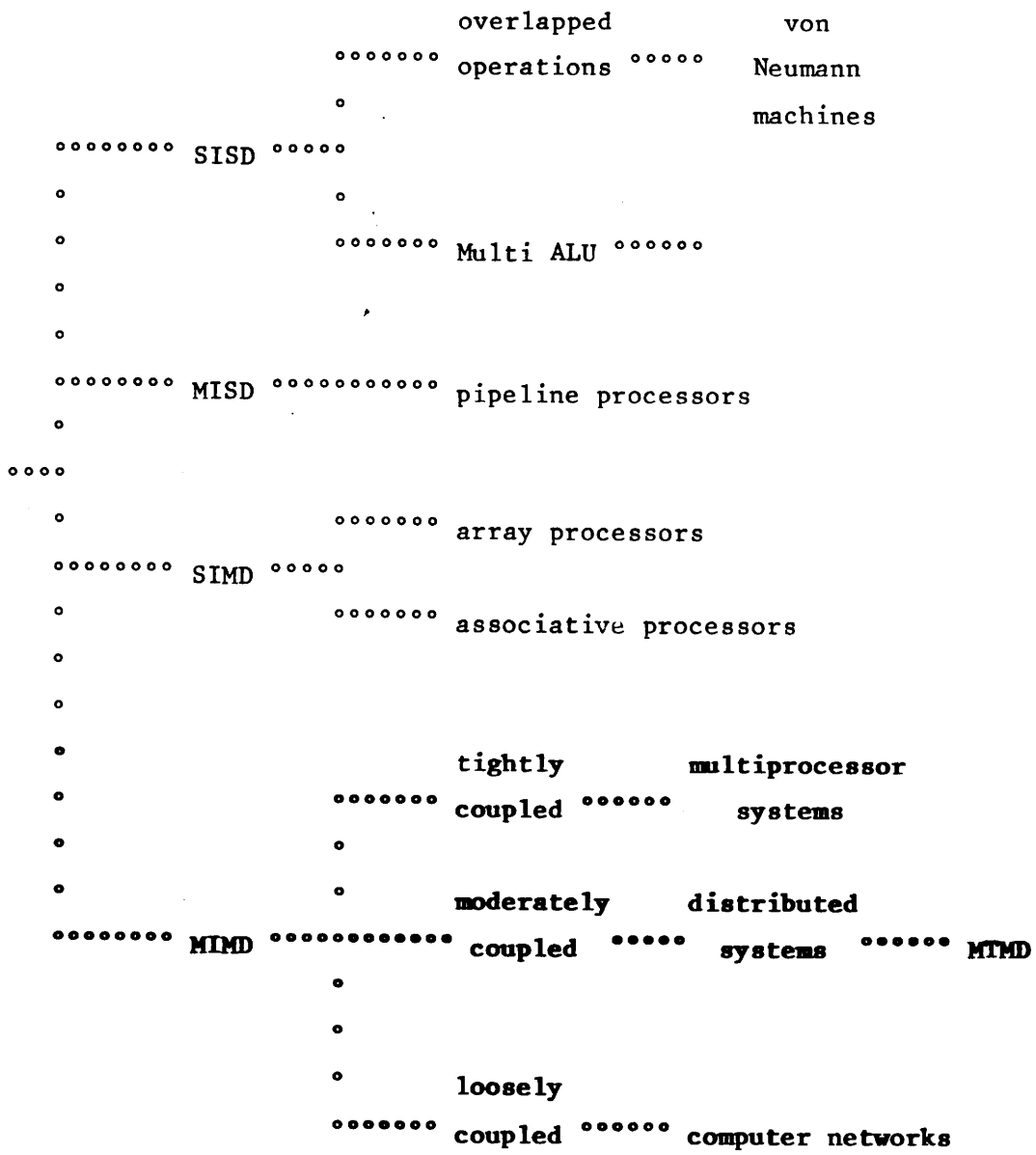
- Loosely coupled systems, also known as computer networks, where the system contains a number of independent, autonomous computers, that can be geographically dispersed. These systems are generally interconnected through data communication interfaces, using serial communication and a well-defined protocol. The network is normally only used for communication, as all computing is done in a single processor or node. In general, loosely coupled systems are characterized by a

low intercommunication speed, which may restrict the type of information being exchanged. Also, loosely coupled systems may inherently be a centralized architecture, when smaller computers are connected to a central mainframe.

- Tightly coupled systems constitute, what is known as "true" multiprocessor systems. These are characterized by having symmetrical processors and shared resources. They normally have access to a common memory, which can be accessed by all processors in the system, although each processor may have its own private memory. They have a common operating system, that controls and coordinates all the interaction between processors and implements a synchronization facility between processors. They may have dynamic load sharing, so all processors are presented with an equal load.

- Moderately coupled systems come between the loosely coupled and the tightly coupled systems and they encompass the concept normally called distributed systems. They consist of autonomous elements with their own processor, operating system and memory, which are normally dedicated to a specific task. As such, elements are not symmetrical, as the hardware and software of each element is tailored to the specific task, it performs. Communication between processors consists generally of data and commands or requests for certain tasks. Moderately coupled systems are characterized by a high speed of intercommunication, and this allows a certain dispersion of different computing facilities within a system.

The following figure show this relationship (see Fathi, 1983):



The MTMD Multiple-Task Multiple-Data classification extension embodies the general form of systems capable of executing multiple task concurrently each with multiple sets of data.

SDC has its prime interest in the MIMD type of systems, and in the case of the decentralized branch office systems, it is the moderately coupled type of MIMD, the MTMDs, that are of importance.

Hence, only MIMD systems will be covered in this report, and the impact will be placed on the moderately coupled type of MIMD systems.

All sub-classifications of the MIMD type of multiple processors may however apply to SDC's environment:

- the entire banking system with computers in the branch offices and computers centrally in SDC is an example of a loosely coupled multiple processor system. Actually this concept dates back to the early 70'ies with the TP1 system and the entire SDC organization is organized to support this type of environment in contrast to many of the commercial banks, that still use a centralized structure of the SISD type.

- within the branch offices, the TP2 system has introduced the concept of having both a local branch computer and individual workstation computers, connected together through the high speed link into a autonomous system. This is an example of the moderately coupled type of MIMD systems, although it can be discussed whether it fits into the MTMD category or not.

- Currently the tightly coupled type of MIMD multiprocessor systems does not exist in the savings banks systems today, except perhaps for the UPLC processor connected to the S1000 local computer. As will be clear from chapter 6, we may however expect

this type of systems to be more common as many of the individual node systems will be designed in this manner in the future. Also microprocessors, like the Intel 432 described in chapter 3, are beginning to support this type of systems.

4.3 Requirements for multiple processor systems use

It makes no sense to use a concept of multiple processors if a given task can be made just as well by one single processor. But "just as well" does not mean, that a task can be done, but rather if it can be done satisfactory, taking into account the different requirements, that may be present.

These requirements are simple in SDC's case, and because they are simple, they may preclude the more theoretical interesting issues in multiple processor design in favor of having a conceivable system:

(1) Throughput is one requirement, that is vital for the savings banks. By throughput is not meant computational throughput alone, but also transactional throughput. Throughput is measured in the savings banks by the response time, when a transaction has to be send to SDC and an answer has to come back. It is also measured by the delays in the system, when a teller has to wait for updates on the screen, or when the teller has to wait, because others are using a shared part of the system. Multiple processor MIMD systems may be used to provide sufficient processing power at the user level to minimize these delays, both by isolating the user's computational need into a private processor and by providing the needed processing power at critical points on the transaction route through the system, so transactions are not delayed by other tasks.

(2) Reliability is another requirement, that is vital. The savings banks are completely reliant on the operability of the branch system. The branch may loose customers or money, if the system is down for a prolonged time or if data are lost. Reliability is an issue for both hardware and software. Multiple processor systems may increase the hardware reliability for the system seen as a whole, as the breakdown of a single processor may not affect the overall operability of the system.

Another little-known factor is the effect on software reliability caused by multiple processor systems. A centralized system, like TP2, where the operating system on one processor has to support many users and hence must make more decisions per time unit, actually increases the probability for software errors, the more users it has to support, while at the same time it decreases the available performance for the user (see Castillo, 1980). Hence, the fewer CPU-cycles per time unit, that the user has, the larger the probability is, that these CPU-cycles will be useless! Multiple processor systems reduces the workload of the single processors within the system, hence increases the software reliability per processor.

(3) Flexibility is yet another requirement, that determines how well new facilities or new components can be introduced into the system. Also flexibility dictates how easy components are removed from the system in case of failure, and thus it impacts the availability of the system. Flexibility will also have effect on the life cycle of the system, as it may allow the system to grow with time or to be upgraded easily. Multiple processor systems will by their very nature be prepared for adding new processors to a system, as well as these

processors may be used for adding new software utilities.

(4) Cost must be reasonable and consequent. Cost is involved when the system is installed, but also as the system is developed, repaired, maintained, expanded, modified and used. Cost is also dependent on the life cycle of the system. Cost must be compared to performance. In single processor systems supporting many users, cost/performance is higher, the less users there are connected to the system. Also, as more users attach to the system, the performance available per user decreases. Even if this effect will straighten out the otherwise irregular cost/performance ratio, the result will be, that the cost/performance ratio alway represents a worst-case value. Multiple processor systems may enforce a concept, where a new user is connected to the system by connecting a new processor to it, hence tending towards a case of constant cost/performance.

All these requirements are not exactly technical. They are common-sense requirements, which cannot be capitalized in a strict manner. They will however be the underlying guidelines for system selections and they should be kept in mind for the rest of this report.

4.4 Interconnection topologies

In all cases of MIMD systems, the general question is how to connect the different processors together. Many solutions exist, but they can be divided into four basic types of interconnection schemes:

1) the common bus topology
2) the star topology
3) the ring topology
4) the fully connected topology

Other topologies can be considered as being combinations of these four or variations hereof.

4.4.1 The common bus topology

In this topology, the processors are connected together via a shared common bus. Communication on this bus are broadcasted to all elements, that connects to it, hence some way of addressing the individual elements may be needed. Reliability of such a system is dependent on the bus reliability. If the bus fails, the system may be inoperable. However most buses of this kind are generally passive. Cost is low for this type of system, as the bus or cable is shared. Throughput will be determined by the bandwidth of the bus.

Common bus topologies are generally found in computer backplane buses and local area networks. Here they are chosen because the topology is simple, flexible and reliable, as the bus itself is usually passive.

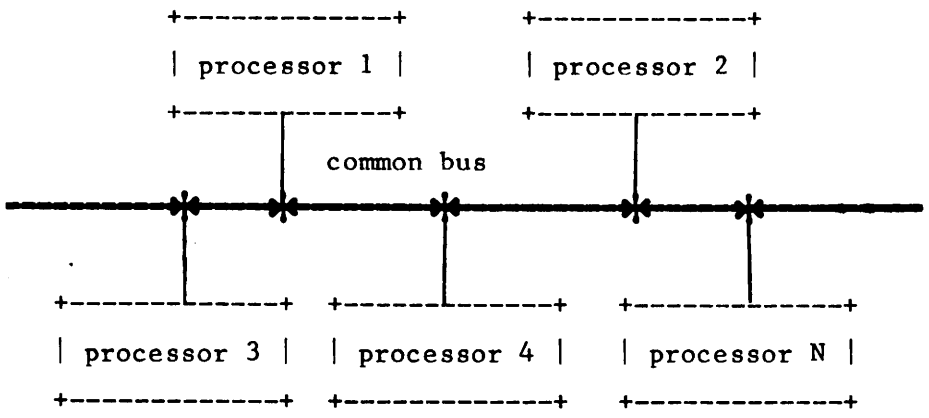


Figure 4.2: The common bus topology

4.4.2 The star topology

In this topology, all processors connect to a central switch. This switch may be passive, but is commonly active and includes the routing logic needed to pass communication from one processor to another. Cabling may be more costly here, but the main cost comes from the switch. Reliability is totally dependent on the reliability of the switch, which usually is lower than any kind of bus or interconnection cable.

Star topologies are usually found in traditionally communication networks or in local network designs based on private branch exchanges (PBXs).

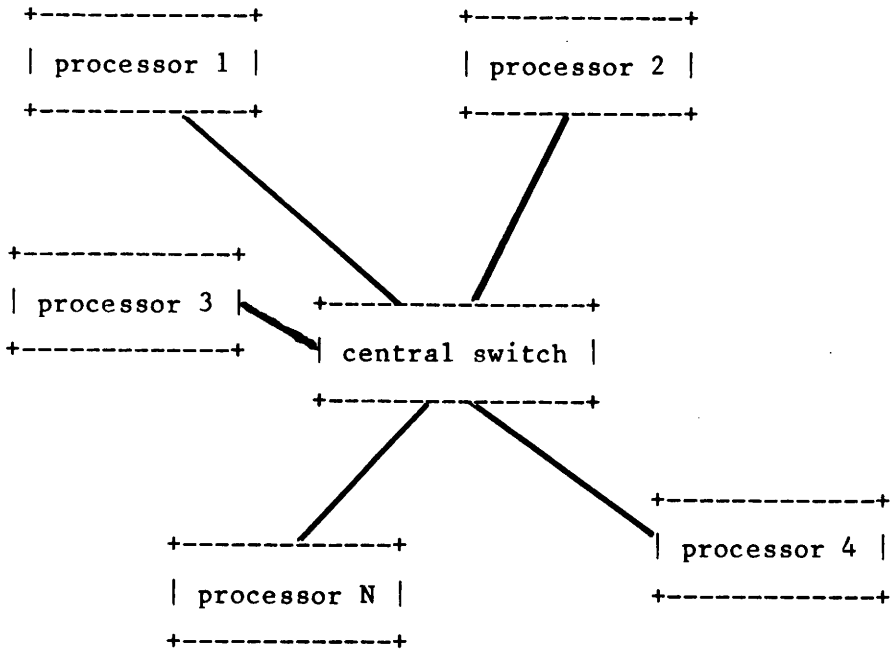


Figure 4.3: The star topology

4.4.3 The loop topology

The loop or ring topology connects each processor to two neighboring processors. Traffic can possibly flow in both directions, but usually circulating traffic in one direction is employed. Reliability is dependent on the interconnection cables and loop interfaces. Flexibility is fair, as the loop has to be cut if new processors must be attached. Cost comes mainly from the loop adaptors.

Loop or ring topologies are found both in traditionally networks, like the TP1 loop lines from SDC to the savings banks, in local area networks or as internal computer buses, like in the Collins C-system (see Weitzman, 1980).

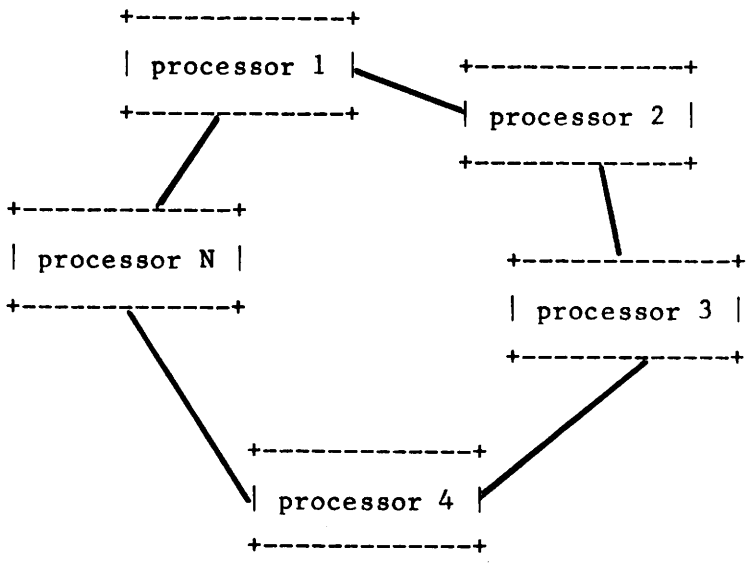


Figure 4.4: The loop topology

4.4.4 The fully connected topology

This topology is probably the conceptually most simple design. Every processor is connected to every other processor through a dedicated path. Costs can be prohibitive as the number of processors increases. Line connection costs can also be high. The flexibility is low in a fully connected system, as a new processors will have to be connected to all others. Reliability of the system can be high, especially is alternate routing is allowed to overcome link failures.

Fully connected topologies are however seldom seen in practice in types of systems comparable to those being used within the branch offices in the savings banks.

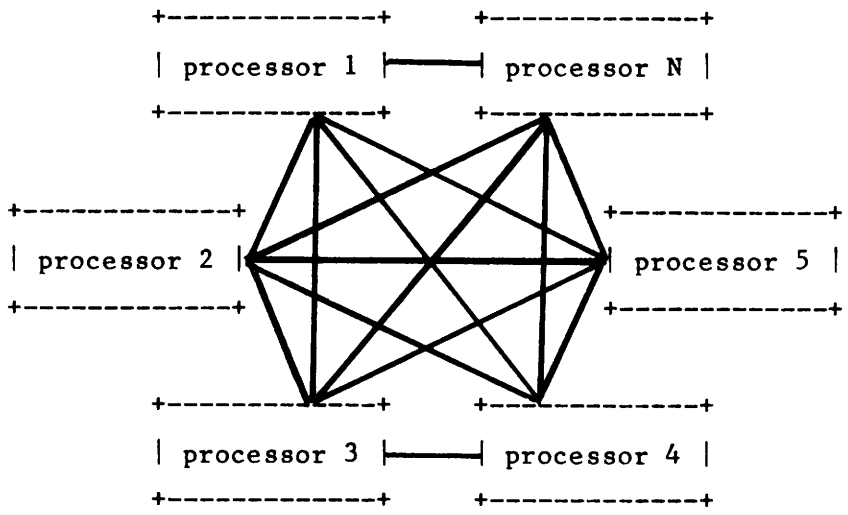


Figure 4.5: The fully connected topology.

Various combinations can exist of these topologies, but those of interest for SDC are the common bus topology and the ring topology. The reason is, that these topologies will be found both in the data communication networks, within the branch offices and within the individual computers, and it is around these topologies, that the general small systems market centers the designs.

4.5 Control, Allocation and Access

The meaning of the topologies above is to provide a channel for data communication between two or more processors. As such, the topologies defines the more physical look of the system.

To make the topology useful for a given application, control, allocation, and access schemes must be implemented.

Control has to manage the use of the channels involved in the topology and to insure, that conflicts between processors, that share the channels, are solved. Control can be centralized, which means that one node or processor controls the access to the network within the topology (which nodes can send messages and when), and the allocation of the channels (how much of a channel a node can use and for how long).

Control can also be distributed, so every node or processor has the ability to attain control of the network by using a common set of rules, that is implemented in and used by every node.

Allocation schemes are implemented in order to optimize the finite capacity of channel.

Access techniques control which node or processor are allowed to use a channel for data communication.

In general, control strategies describe **where** control of access and allocation resides in the topology, access techniques decide **who** gets the channel, and allocation schemes determine **how much** channel capacity a node or processor can have (see Digital, 1982).

Within a specific type of system, there may exist strong relationships between these techniques and schemes, but in general, they are independent and can be combined to provide the needed characteristics of a topology.

SDC's interest will be in those topologies, that implement distributed control. Distributed control implies, that control can adapt to various distributed environments and varying numbers of nodes on the system. In fact, distributed control potentially allows a system to configure itself as changes are being made. Such a feature minimizes the efforts needed to install and change systems, and hence it will lower the total system costs.

4.6 Market

The components used within a given topology are generally processor systems available on the market. Not all these systems will fit into a given topology and some systems have never been designed to interact with other processor systems.

The market of microprocessor-based systems have in the last couple of years grown to an extent, where it today is in the progress of totally dominating the distributed computing market.

Two reasons cause this: the cheapness of microprocessor based systems make them interesting for a new and larger market segment constituted by smaller businesses and individual departments within larger enterprises. The standards, which have gone into the design of both hardware and software, have also caused the design of these systems to become more straight-forward and the implementation of software to become more easy.

A new breed of companies are dominating this market, whereas the older, well-established companies have done nothing, except to follow the newer companies (see Norman, 1983).

In terms or architectural differences, two major segments exist in the microprocessor-based market: the personal computer segment and the supermicro segment.

4.6.1 The personal computer market

The personal computer (PC) segment has been dominated by small companies, which have been very dynamic in adapting the new technologies and utilize the technology in systems. This market arose with the advent of the cheap microprocessors and the

combination of (1) microprocessor-based systems, (2) cheap and fast secondary storage in the form of floppy disks, and (3) the availability of standard easy-to-use software tools, like the BASIC programming language and the VisiCalc™ spread-sheet calculator. This market was until recently totally neglected by the major systems manufactures and is still neglected by most data processing departments. The market is a high volume market with more than 1 million PCs for business use being sold in 1983 in the U.S. alone. IBM, who three years ago were unaware of this market, is shipping 50000 personal computers per month today, with a revenue from this segment of 500 Millions dollars in 1982, starting from zero in 1981. Compared to these figures, the size of the TP2 system becomes infinitesimal, and efforts within SDC to design facilities, that already exist on the PC market, will actually be a waste of money.

Personal computers in this report denote those intended for business oriented use, and not the even bigger plethora of home computers. Whereas software for these systems are rather standardized (due to the high volume and large number of manufacturers), hardware comes in different forms and standardization has been lacking. One reason for this, is that most computers in this area is basically single-board computers, with only spare possibilities for hardware extensions. Most of these processors do however apply an internal multiprocessor type of architecture, either as dual-processor systems with two different types of processors (e.g. 8- and 16-bit), or as distributed function systems, in which specific tasks are assigned a processor each. The typically examples of the latter is having a special screen processor, that unburdens the application processor from screen handling and refreshing.

Interesting enough, the system price for personal computers in single-unit quantities is comparable to the price for the TP2 terminal computer in large quantities. This will enforce a heavy competition on TP2 from the PC market, and it will be even greater, as the market price for PCs will drop to about 50% of its current level within probably the next year.

4.6.2 The super-micro market

The supermicro segment, on the other hand, is paying more attention to hardware standards. First, this market segment normally produces multi-user systems, where several terminals can be connected. This imposes needs for a flexible I/O structure and needs for more memory, than can be implemented on a single board. Such systems typically employs hardware standards at the backplane level and the I/O interface level, as well as they widely disperse processors to handle specific tasks (see Warren, 1983.2; DeJoung, 1983; Geisler, 1981; Huie, 1983; Isaak, 1982; Schultz, 1983; Serlin, 1983).

4.6.3 Future directions

Also the microprocessor evolution has caused renewed interest in practical research of true multiprocessor systems, employing lots of identical processing elements (see Cohen, 1983; Electronics, 1983; Electronics, 1983.2-3; Inselberg, 1983; Manuel, 1983.2).

The impact on SDC will come from the personal computer and supermicro area. As stated before, more than 1 million personal computers will be sold in the United States in 1983, and probably over 30000 supermicros. This evolution make basis for a support industry, which will manage the development of applications and hardware support for these systems.

As such, most systems on the personal computer and supermicro market has been autonomous, stand-alone systems. Communication towards other systems are carried out through old-type standard communication protocols. One problem has been, that the evolution of data communication standards for traditionally loosely coupled systems have never really anticipated the break-through of micros (see McQuillan, 1980). This imposes big problems in todays systems, when applications on microprocessor-based systems need to exchange data with other systems.

Within the personal computer industry, some attempts have been made to implement moderately coupled type of topologies (see Rolander, 1982), which would allow personal computers of similar type to intercommunicate.

On the other hand, most of the personal computer industry is now geared towards designs with moderately coupled systems, based on the new local network standards (see Canning, 1983.2; Ether, 1982; Kayaler, 1983; Manuel, 1983). These designs will allow inhomogeneous computers using different types of standard operating systems to be connected to the system.

Still, the market is in movement and has not stabilized. The last products have not been seen yet and totally other different routes may be taken. However, it is a low-cost market with no place for very theoretical solutions, so changes will occur to make it more easy for users to implement systems based on these products. SDC must be able to cope with these changes.

4.7 Where to concentrate?

As mentioned earlier, the current SDC system is constructed with several processors placed in the branch offices of the savings banks. This trend will be even more visible in the future, where off-the-shelf systems will employ processing power located at the individual user.

Processing power can be conceptualized as being residing in a branch office, in an individual node or computer with a branch office or on an individual board within a node or computer.

In all cases, these processors need to exchange information under certain circumstances, and this exchange will be made by the connection scheme and topology between the computers.

In chapter 5, the interconnection between the individual nodes and the individual branch offices will be discussed. Strategies here will be related to both the loosely coupled and the moderately coupled systems.

In chapter 6, the interconnection between individual processor within a node or stand-alone computer system will be discussed. The strategies employed here will be related to both the moderately coupled and the tightly coupled systems.

In chapter 7, the system software approaches to support these types of interconnection strategies will be discussed. At this point it is important to consider the general impact from the personal computer market, because it will be this market, that will set standards for the future software implementations on systems based on the cheap microprocessors and hence based on multiple microprocessors.

Chapter 8 will provide the general view on how to integrate all these concepts into a system architecture suitable for the requirements of SDC and the savings banks. Again, the fundamental assumption here is, that multiple processor systems will be used and elements in such systems will come from the general marketplace.

Chapter 9 will then discuss some practical steps, that may be taken within SDC to move towards the system architecture proposed in chapter 8, and to make benefit of the actual state of the market of cheap microprocessor-based systems.

this page intentionally left blank

CHAPTER FIVE

DATA COMMUNICATION

5.0 Introduction

Data communication is probably the most fundamental issue in the savings banks data processing system. Data being gathered in the branch offices must be communicated to the central systems in SDC or between individual branch offices in a fast and secure manner.

The purpose of data communication is to transport data from one place to another. A place in this sense can be a branch, a physical node, a computer or an application dependent on the view, that one has of the system.

A secondary purpose of data communication is to ensure, that data is transported without errors and, in fact, provide a certain guarantee, that data will be transported.

Data communication between the branch systems and SDC has been transaction oriented since the introduction of TP1 in the start of the 70'ies. Transaction communication means, that data is formed and collected into a unity, called a transaction, which produce certain predefined operations when received. Also, a transaction is self-contained, meaning that it is independent of other transactions being communicated in the network or being processed at the receiving end.

Transaction communication should be seen in contrast to other types of data communication, like dialog-oriented communication or file transfers. In these types of communication, a history is generally involved in the communication, which may cause each individual unit being communicated to be dependent on previous events and affect coming events.

Designs employing data communication should be accompanied by two thing: first a fundamental understanding of what is going on where in the system and secondly a fundamental documentation of what is being communicated and how, separated from the traditional system or program documentation. These requirements are needed to obtain the best usage of the communication system and to allow for later changes and updates.

Data communication systems have traditionally been surrounded by much respect. Today, this trend is disappearing, as the technology makes it more easy and simple to design and implement data communication features in a system.

This simplification of data communication system design is reliant on standards. Standards enforce the industry to produce components or elements conforming to those standards, and they provide a general reference point, which is needed as data communication always includes at least two participants.

Standardization within data communication is concentrated around the work, that the International Standards Organization (ISO) is performing to standardize a reference model of Open Systems Interconnection (OSI). The OSI model should provide a common basis for the coordination of standards developments for the purpose of systems interconnection, while allowing existing and evolving standards to be placed into perspective with the overall reference model.

OSI is concerned with the exchange of information among systems and not with the internal function of each individual system. Also, OSI is concerned not only with the transfer of information between systems, but also with their capability to interwork to achieve a common (distributed) task, in other words the cooperation between systems.

OSI employs a concept of architectural layering by defining a seven layer reference model, where each layer communicates with the corresponding layer on another system through a peer to peer protocol and interacts with surrounding layers on the same system through a set of interface services.

In a general sense, the OSI model enforces an independence between layers, so different implementations of layers (i.e. at the physical level) can be used freely, as long as the interface services are kept.

A complete description of all OSI layers will not be included here, but it can found in numerous publications and articles, and is (or should be) known by all people involved in the management or design of data communication oriented systems (see Davies, 1979; Schneider, 1979; Walden, 1979; Wecker, 1979; Bass, 1980; Canning, 1980; Cotton, 1980; Gordon, 1980; Hsi, 1980; McQuillan, 1980; Piatkowski, 1980; Warner, 1980; Zimmermann, 1980; Burkhardt, 1981; Canning, 1981; Folts, 1981; Naffah, 1981; IEEE, 1982; Tannenbaum, 1981; Davis, 1982; Mier, 1982; Standards, 1982; Hindin, 1983)

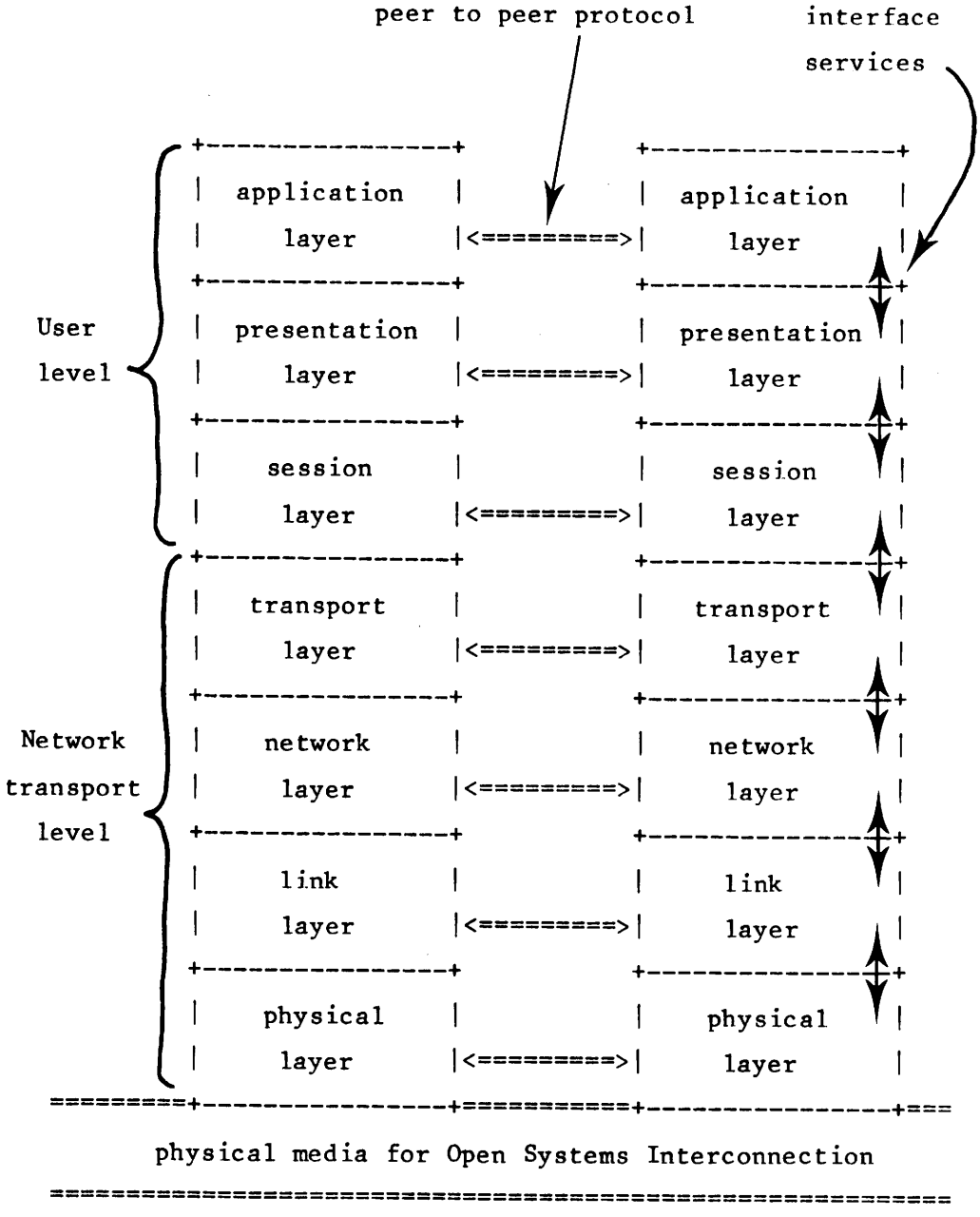


Figure 5.1: **OSI reference model**

The OSI model actually applies to all types of communication between systems. Of interest here are two types of data communication: the long distance data communication, which takes place over a large geographical area and the local area data communication, which takes place within a geographically restricted area.

5.1 Long distance data communication

Long distance data communication are characterized by allowing geographically dispersed systems to intercommunicate, often at limited speeds. In Denmark, this type of communication is controlled and served by the Danish PTT, and systems will have to adapt to the possibilities, that the PTT offers. Naturally, deviations from the standard offerings can be made in certain cases, but in large systems, like SDC's, it will normally be most economically to use the standards.

Data communication systems for long distances are generally called wide-area networks, but I will use the term nation-wide network to describe the SDC network. This is because, the SDC network effectively spans the entire Danish nation, and because it is self-contained in this sense. The SDC nation-wide network is only used for connecting branch offices and the central systems at SDC, although in the future it might be foreseeable, that it will connect to other Danish payment services as well. Communication outside Denmark will generally be accomplished by having gateways between the nation-wide network and foreign networks, e.g. the S.W.I.F.T interbank network.

Several standards exist for nation-wide types of communication, ranging from ISO standards to vendor-specific standards, like SNA and DECnet. Most of these vendor standards will probably in the long run loose importance, as public data networks conforming to the ISO standards will be more common.

A contributing factor is the progress within the microelectronic area, where communication chips offer a significant help in communication systems design. Most standard logical link level protocols are now being supported by these chips, and in the case of the emerging ISO standards, like X.25, these are now fully controlled by chips. This will

enforce the industry to move to these standards, as using non-standards or standards, which require a significant amount of software development, will simply not be economic.

SDC is today using standards for communication between the branch systems and the central systems in TP2. These standards (see chapter 10) include the X.21 network connection and the HDLC link protocol. A correct use of such standards will ensure, that no data can be lost without notice and recorrection.

In some sense, the TP2 communication system, tp-monitor (see OL, 1982.2), can be considered as a partly open system according to the OSI terminology. The SDC model, as shown on figure 5.2, has only 5 layers in contrast to the seven layers of the ISO model. This has significance related to the SDC requirements of short response times (see chapter 10). Actually, the SDC tp-monitor concept is fairly consistent up to and including the end-to-end layer.

The mistake of SDC has been to include the OSI presentation layer within the TP2 application layer, and, by separating the organizational responsibility for these layers into different departments, allowing ad hoc solutions for the application layer being implemented by people not familiar with the fundamental concepts of data communication methods. This means, that the TP2 application layer and its interface to the TP2 end-to-end layer does not conform with the OSI requirements for layer design, and as such, the TP2 communication system cannot interact with other open systems. The fundamental problem is that of data representation, where data transactions and answers in TP2 has been defined in a rather mainframe-like ad hoc style, with a special record layout for each answer, which has to be known at both ends. The result is, that no general processing of answers can exist, as all answers rely on an application program, that knows its internal structure. Also,

the missing awareness of data representation has caused an increase in response times, as answers will contain superfluous information. One example is all text data, which is send in its full length, justified to fit into predefined fields on the TP2 screen, even though most of the transmitted data is empty spaces. This can amount to an increase in response times of 0.5 - 1 second.

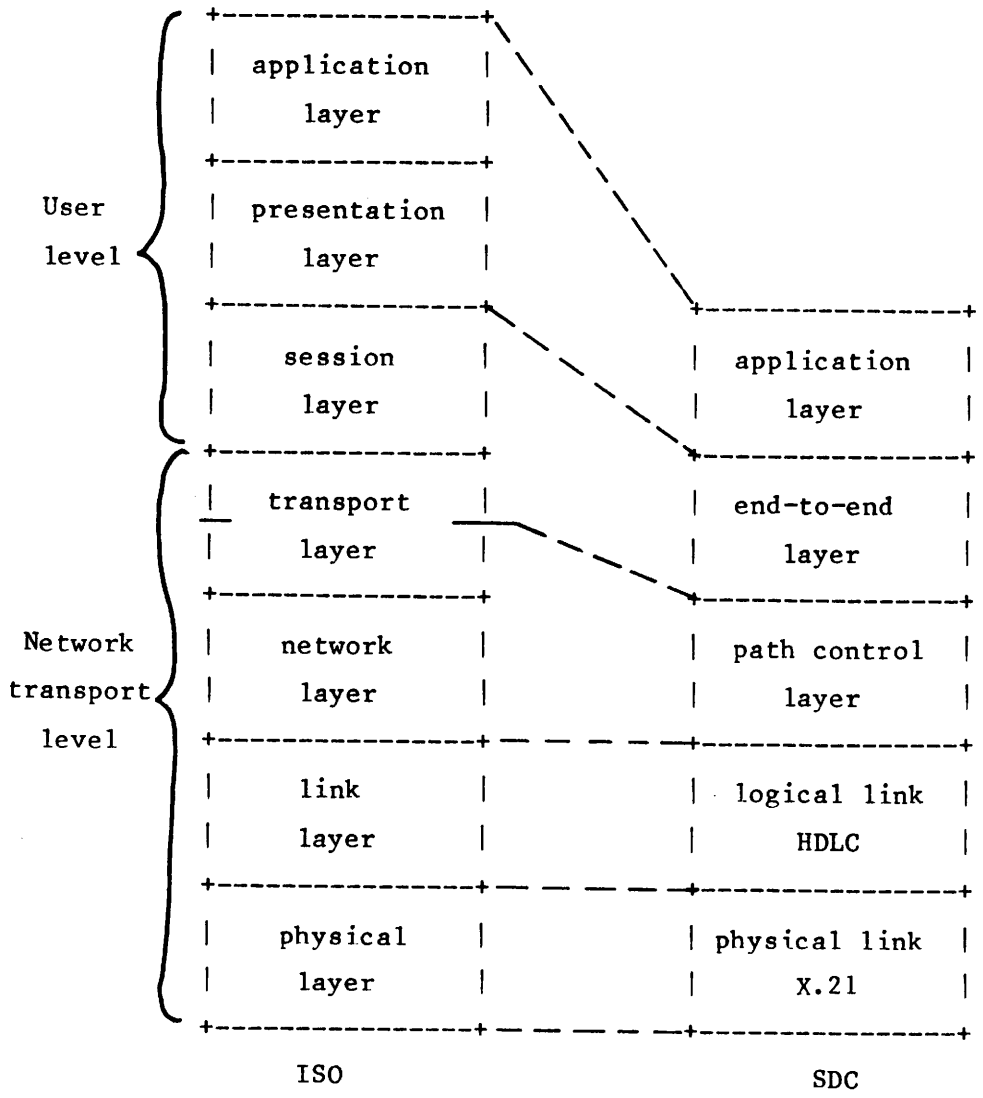


Figure 5.2: **SDC network layers**

Another impact of this strategy is that it burdens the design of new systems to be connected to the network. One example is the use of personal computers or customer computers, which will impose problems due to the requirements of large disk space for holding all descriptors of the individual answers.

Other problem areas will be in the editing of output formats, which now are dependent of the structure of the answer. Also, when any modifications of answers are made within SDC, corresponding modifications must be made in the decentralized software. Perhaps the worst problem lies in the future integration with other data centers and information services, where a large amount of manpower must be allocated just to keep track of conversion problems.

As a consequence of having detected these problems as a part of the practical case of this project (see chapter 10), a pilot project has been initiated within SDC with the purpose of defining and evaluating such a presentation standard (see Bertelsen, 1982.3). However, this is a low-priority project due to the other problems in the TP2 system and other projects. One fundamental problem is, that once accepted, it will require application changes both in the mainframe applications and the TP2 applications if not a suitable migration strategy is implemented, like being able to support both the old and the new format simultaneously in a transfer period.

The benefits involved are however significant on the long term scale. Again one should realize, that the project proposal covers an internal SDC standard, which in the long run may not be compatible with on-coming international standards. It will however be far more easy to convert from one standard to a new one, rather than converting from nothing. The sooner, SDC takes the first step, the better.

In the future, however, the utilization of OSI-related standards will go within the branch offices to connect the individual computers in such an office, and hence this trend may eventually also necessitate the need for applications modifications.

It is therefore important, that SDC follow up on the progress within the standardization areas within the nation-wide networks. If we have problems here, we will certainly get more problems going to the concept of local area networks.

5.2 Local area networks

Traditionally, data communication was oriented towards transporting data over longer distances, with small speeds and often through connection to public carriers. The small speed of this type of communication enforced the development of ingenious protocols and concepts, which would optimize the use of a given physical link, and minimize the delays, that might occur for data being transported through this link. Often it could be beneficial to go into research, looking for new methods to exchange data. Such approaches could also be justified, as most projects would connect a rather limited number of computers into a fixed and static network.

The advent of microcomputers and microprocessor-based systems has in the recent years caused a need for a new type of data communication network, the local area network. The need has arisen due to the fact, that the emphasis is shifting from the traditional type of centralized computer structure, with a single computer serving many terminals, to a more distributed type of structure, where each terminal has its own processing power.

Local area networks are, as the name indicates, networks deigned for use within a limited area, typically a building, a campus of buildings or a branch office. It should be noted, that networks of this type have always existed, and the connection between a dump terminal and a centralized computer could well be described as a local network.

However, in the industry, the term "local area networks" have attained a special meaning, as they denote a type of networks, which offer high speed, low cost, flexibility, simplicity and compatibility. As such, traditional connection networks between terminals and computers are not considered being a local

network by the industry, and a network like the high speed link in the TP2 system is just on the border of the industry definition.

Local area networks have been underway for many years, but it was not until the late 1970'ies, that the computer industry suddenly became aware of the impact, that local area networks would have on future system design (see Hunt, 1979). Since then, attempts have been made to define standards, and several non-standard local networks have arisen. Today, around 80 different local area network concepts exist from different manufactures (see LOCALnetter, 1982), each trying to get their share of the market. However, only two major attempts are underway for setting a standard.

The local area network market is dominated by a large fight over various topics in network design. These topics vary from the type of transmission medium being used, to the way control is implemented in the network, the speed needed, the accessing scheme, the topology of the network and the applications of the network.

The truth is, that no single local network approach exists, that will satisfy every user. This makes it even harder for a user to select a local area network, as almost every manufacturer of networks claim that particular network to be capable of everything.

In the following, some of the basic issues in local network design will be covered. After that, the two standard approaches will be examined.

5.3 Local area network concepts

Local area networks can be described by many different characteristics. The most important of these are the properties of the physical transmission medium, the access method and the topology of the network.

5.3.1 Physical media technology

The transmission medium is still an issue for numerous discussions as the choice of medium often impacts the costs and possibilities of the network. Several approaches exist, each having different characteristics. The types of physical media include the twisted pair cable, the coax cable used in baseband networks, coax in broadband networks and fiber optic cables.

Bandwidth is a measure of the capacity of the medium to transport data. The twisted pair type of cable offers relatively low bandwidth, less than 1 Mbit/sec. Coaxial cables used in baseband modes offers a moderate bandwidth, typically 5-50 Mbit/sec. **Baseband technology** signifies the fact, that only one channel is responsible for using the bandwidth of the cable. This channel can either transmit at the highest speed possible or it can be quiet. Hence the data rate on the cable can vary from 0 bit/sec to the maximum rate. Baseband technology can be compared to an old telegraph Morse cable. Only one can morse on the cable at a given moment, and the signalling speed is dependent on the capability of the keying person and the length of the cable, even though the cable potentially could support a higher speed, using other methods.

On the other hand, broadband coaxial cables offer high bandwidth of several hundred Mbit/sec, but generally this high bandwidth is divided in numerous channels, each with a low bandwidth (50 Kbit/sec up to 1 Mbit/sec). **Broadband technology**

is actually the same used for cable television, where several channels can be transferred simultaneously over the same cable. However, like a TV-set having a receiver, which can be tuned to receive a specific channel, data communication over broadband networks must include the same type of selective receivers and transmitters, thus making the actual interface costs high, especially when multiple services has to be used by a single network station.

Fiber optic cables are still not commonly used in local area network designs (see Sauer, 1980; Bliss, 1983), although they potentially allow for very high bandwidths. The high bandwidth of fiber optic cables (as well as broadband cables) also offers the possibility of transferring video and voice simultaneously with data.

Another factor is the ability to handle many nodes or connections to the media. This can be dependent of the electrical characteristics of the network and the possibility of having 'open' connectors, i.e. plugs with no computer attached. Twisted pair technology offers a limited number of connections, typically less than 100. Baseband coax offers typically between 256 up to 1000 connections per network, whereas broadband technology can accommodate several thousands connections within a single network. Fiber optic is still a problem, as current designs only allows point-to-point connections, which means that the cable must be broken and interspersed with active repeaters, when a new node has to be attached. Some research has however been done to investigate the possibilities for a shared use of a fiber optic cable, but this have still a long way to go.

The distance or length of the local area network is dependent on the speed and the electrical characteristics of the network. Incidentally, a provision for supporting a long distance may be of little use in a local area network, which usually are

geographically restricted.

Another issue is how easy various network topologies, like common buses, rings, stars etc. can be implemented. The electrical cable media allow for potentially any topology, whereas fiber optic is restricted and is mostly a candidate for ring or star topologies.

Ease of installation signifies partly the ease of wiring a building, which is almost the same independent of cable technology, but also the ease of connecting nodes to the cable. Generally, coax cables offer the easiest type of connection here due to the advent of intrusion-type connectors, that will attach to the cable without having to break or deisolate it. Hence, with these types of connectors, it is only a matter of seconds to connect a new system to the network, and it can ideally be done, while the network is running.

Noise immunity is an important factor in local area network design. Generally, noise will be detected and corrected by the various error detection schemes used in the logical link protocols on the network. However, noise may be more critical for local area networks, due to the higher speeds. A noise spike of 100 microseconds will hardly be noticed on a traditional communication network. On a public network line, running 2400 bps, it will affect less than a quarter of a bit period, and normally nothing will be lost. On a local network running 10 Mbit/sec, this noise pulse corresponds to 1000 bits or a complete transaction. Hence, even small noise level on a local network can cause excessive retransmission.

Cost of the various media does not include installation cost, but the cost of the media itself.

Media	Bandwidth	Ability to handle many nodes	Distance	Topological versatility
Twisted pair	low	low	low	high
Baseband coax	low to moderate	moderate	moderate	high
Broadband coax	high	high	high	high
Fiber optic	very high	low	very high	moderate to low

Media	Installation ease	Noise immunity	Cost
Twisted pair	moderate	low to moderate	low
Baseband coax	high	moderate to high	moderate
Broadband coax	high	high	moderate to high
Fiber optic	moderate	very high	very high

5.3.2 Local network accessing schemes

Accessing schemes determine how nodes gets permission to use the network medium. Several different forms of access exist, and these are subject for immense discussion in the local area network world. The two most common access schemes are the random access scheme represented by the CSMA/CD, and the deterministic token passing.

Random access is provided by the Carrier Sense Multiple Access with Collision Detection (**CSMA/CD**) protocol (see Metcalfe, 1976; Ether, 1980; IEEE 1982.3; Shoch, 1982; Stuck, 1983). This method allows any node to access the network in a random fashion. When a node wants to access the line, it first monitors the line to see if any other node is using it (Carrier Sense). In the absence of any signal, the node (and possibly any other node monitoring at the same moment) is free to use the line (Multiple Access). The node then starts transmitting the data on the line. Sometimes, however, several nodes may start using the line at the same moment, which means that the signals will be intermixed. This effect is called a collision.

A node is however able to monitor the line while it transmits, thus being able to detect, whether the signals on the line correspond with the data, that it should send. If they do not, a collision (or noise) is present on the line, but as the node can detect this (Collision Detection), it will immediately stop further operation and release the line. This is called back-off. All nodes, that detect a collision will back-off and wait for a random amount of time, before they try to access the line again. In this way, most conflicts are solved, because nodes are unlikely to try to access the line again at the same moment. Theoretically, a node can however wait forever under worst-case conditions, if a CSMA/CD network is heavily loaded. Also the network works with a first-come-first-served

philosophy, which means that there is no priority involved.

The CSMA/CD method is conceptually simple. Being simple, it obviously lacks some features known from traditional communication networks, like deterministic access. One must here consider the type of environment, that the local network has to work within. Office automation applications are typically not dependent on the non-deterministic delays involved in CSMA/CD, but rather focused on low cost and ease of use (see Dahod, 1983). Industrial control applications do however need a certain guarantee, that urgent messages can be transmitted without delay or within a certain timeframe. Hence, industrial application need a method with deterministic and prioritized access.

The most commonly used approach in local area network design to implement deterministic access is **token passing.** Token passing is a method, where each node in the network receives and passes the right to use the channel in turn and in a predetermined order (see Jones, 1982.2; Myers, 1982; IEEE, 1982.4; Miller, 1982; Stieglitz, 1982).

Tokens are special bit patterns or packets, that circulate in the network from node to node, when there is no other traffic. When a node possess the token, it has exclusive access to the network for transmitting data, thus avoiding conflicts with other nodes. When a node has finished transmitting the message, it puts the token back in circulation.

Token passing works as a type of distributed polling, and as all other polling systems, delays will be present due to the polling mechanism. Hence, token passing will show its major benefits in networks with heavy load, whereas in networks with a small load, it will introduce delays compared to CSMA/CD. However, these delays can be computed for worst-case conditions.

Token passing can be used in various topologies, including both the common bus and ring networks. For ring networks, another form of token passing exists, called the slotted ring. This is implemented in a ring topology, where a number of slots or frames of fixed sizes circulate around the ring. Each frame will generally contain fields for source and destination addresses, control information and data. When a node wants to transmit data, it waits for an empty slot to come by. It then inserts the data and marks the slot as being full. As the frame circulate further along the ring, the other nodes will check it to see if the information was addressed for them. If so, they will copy the data and mark the slot as being free.

Slotted rings do provide a simple access mechanism, but they also have some inefficiencies. Because of the high transmission speed on the ring, only few slots can be placed on the ring and these slots are normally of limited size. Hence the amount of control information in a slot becomes high in proportion to the amount of data, and messages must be divided into several very small pieces, that fit into a slot.

Other access methods use a dedicated approach in which the capacity of the network is subdivided into smaller pieces. Frequency division multiplexing (FDM) divides the available bandwidth into smaller, independent frequency channels, each of which can be use as a separate channel for transmission. Time division multiplexing (TDM) use another method in which each node is allowed to use a small time interval in turns, during which it may send data.

The following tables show an overview of the different access methods. The characteristics are generally dependent on the various implementation as well as the speed. Both CSMA/CD and token passing networks facilitate a distributed implementation of control, whereas methods as FDM and TDM require centralized

control, which constitute a single point of failure. Bandwidth seen from the node is dependent on the type of physical medium being used, but here CSMA/CD and token passing are generally superior to TDM and FDM.

The ability to handle many nodes is less for CSMA/CD due to the contention problems, that may occur in large networks with heavy load. The maximum distance between nodes is dependent on the media being used and the transmission speed. In general, the length of a frame in the CSMA/CD type of transmission must be twice the as long as the propagation time to the most distant node in order guarantee collision detection before the transmission is completed. Token passing on the other hand requires smaller distances to minimize the delays involved in circulating the token. Flexibility determines how easy new nodes can be attached to the network. Here CSMA/CD has the fewest restrictions, whereas token passing may involve reestablishing the token circulation and allocation of priorities.

Accessing method	**Control**	**Bandwidth**	**Ability to handle many nodes**	**Transmission distances**
Random (CSMA/CD)	distributed	high	moderate	moderate to high
Polling (token passing)	distributed	moderate to high	high	low
Dedicated (TDM, FDM)	central	low	high	moderate

Accessing method	Contention	Determi-nistic?	Flexibility	End to end message delay
Random (CSMA/CD)	high	no	high	unknown
Polling (token passing)	low	yes	moderate	low to moderate
Dedicated (TDM, FDM)	low	yes	low	low

5.3.3 Local network topologies

Network topologies (see chapter 4) for local area networks are normally linked together with the type of access method used. CSMA/CD are used with the common bus topology. Token passing are normally used in ring or loop networks, although it can be found in bus networks. Star networks are seldom found in local area application, whereas tree networks are implemented in more traditional concepts using centralized control. Generally, the choice of a certain topology is governed by a mixture of requirements and vendor specific possibilities.

Topology	interface complexity	flexibi-lity	expanda-bility	reliabi-lity	cost
Bus	low to moderate	high	high	high	low
Ring	low	moderate	moderate	high	moderate
Star	low	low	low	moderate	high
Tree	moderate	high	high	high	low
Varied	high	very high	very high	moderate	high

5.3.4 Alternative types of local networks

It should be noted, that some proposals still exist on using private computerized branch exchanges (PBX, PABX, CBX or DBX) as a basis for local area networks. I do not believe these approaches to be of significance for the savings banks environment and have chosen not to include a more complete discussion here (see Coleman, 1982; Digital, 1982; Phister, 1982; Wiley, 1983).

5.4 Local area network standards

Two standard proposals exist for local area networks, which will determine the future developments in this area. The standards are the Ethernet and the IEEE 802.

5.4.1 Ethernet

Ethernet was originally designed by Xerox to be used in office automation applications (see Metcalfe, 1976). The first version was called experimental Ethernet, and was a 3 Mbit/second local network, using the CSMA/CD access method on a common bus topology. Experimental Ethernet dates back to 1972 and provided the background and research that eventually directed the design of the second-generation of Ethernet. This was done in 1980, as Xerox together with Intel and Digital Equipment published a refined specification, the "blue book", which today constitutes the de facto Ethernet standard. This specification describes a 10 Mbit/second local network, using the CSMA/CD access method on a common bus topology, using a baseband coaxial cable medium (see Ether, 1980; Ryan, 1981; Shoch, 1982).

Ethernet defines the network interface as two parts: a controller and a transceiver. The controller manages the interface towards the computer system and performs the encoding/decoding of frames, the basic CSMA/CD protocol, error detection, buffering and address recognition. The transceiver contains the electronics needed to interface the coaxial cable for transmitting and receiving data, for carrier sensing and collision detection. Also it include an electrical isolation between the cable and the controller electronics.

As such, two points of compatibility exist: the interface between the coaxial cable and the transceiver and the interface between the controller and the transceiver. Typically, the controller will reside on a single circuit board within a computer or workstation, whereas the transceiver will be placed in close proximity to the cable. Also, the controller and transceiver will typically come from different vendors.

Ethernet supports up to 1024 nodes on a single network. The maximum cable length is 500 meters, but through the use of repeaters a single Ethernet can extend up to 2.8 kilometers.

A minimum has been imposed on the packet length to guarantee collision detection on long networks. Minimum packet length is 64 bytes, of which 46 can be used for data. Maximum packet length is 1518 bytes, of which 1500 can be used for data. This means, that all frames will be at least 64 bytes long, but normally this does not impose any performance restrictions, as most frames will contain a certain amount of header and control information used by higher layers.

Addresses in Ethernet is 48 bits long. This has been a point of arguing in local network design, as most networks traditionally has only 8 or 16 bits for addressing. The reason, however, has been that of simplicity. The 48 bit address allows for assigning a unique address to every node, that will ever be produced for connection to Ethernet. Addresses are assigned to users and manufacturers by a special service office at Xerox, and this address administration ensures, that any Ethernet devices can connect to any Ethernet network at any time, without requiring, that the user should control and maintain his own administration of network addresses. Also, the long address field allows for enough free addresses to support broadcasting and multicasting, in which all or a group of devices can be addressed with one frame.

One should note, that Ethernet only defines the lowest layers in reference to the ISO Open Systems Interconnect model. This means, that an Ethernet capability alone is not sufficient to guarantee compatibility between different vendors' equipment. Definition of the higher layers is still in progress, although standard proposals have been published by both Xerox, Intel and the U.S. Defense Applied Research Projects Agency (see Mier, 1982; Announce, 1983; Aseo, 1983; Hindin, 1983; Parker, 1983.2).

The prime advantage of Ethernet is, that it implementable in hardware in contrast to the higher layers, which are more software-oriented. Several manufactures offer integrated Ethernet VLSI chips today, which implements the functions of the Ethernet controller directly in hardware. The transceiver is still mainly made up by discrete components.

The chips are now available for Ethernet like the Intel 82586 LAN controller chip (see Arst, 1982; Beach, 1982; Intel 1982.3; Intel, 1983; Intel 1983.3). Another company, Seeq, is already producing and selling them in quantities for about $135 in 100-unit quantities (see Kopec, 1982; Parker 1983.2). Later this year , Fujitsu U.S. in joint venture with Ungermann-Bass Inc. (see Goodrich, 1982) Xerox, Digital, AMD, Mostek and Rockwell will have chips available (see Coleman, 1982.2; Coleman, 1982.3; Hindin, 1982).

In the meantime, controllers based on off-the-shelf logic, are available and can be directly installed in systems using standard backplane buses, like MULTIBUS, S-100 and VME-BUS (see Harakal, 1981; Ether, 1982; Killen, 1983; Manuel, 1983). The cost for these controllers is now dropping to 2000$ in small quantities and $1000 in large quantities. The other needed part for Ethernet is the transceiver, which is the electronics directly attached to the cable. Transceivers has been on the

market for a long time, and the price is dropping. Current prices is less than $300.

Standard software for Ethernet is available, like Unet from 3Com Corp. which covers the top five layers of the ISO model and runs on UNIX and UNIX derivatives (see Aseo, 1983; Parker, 1983.2).

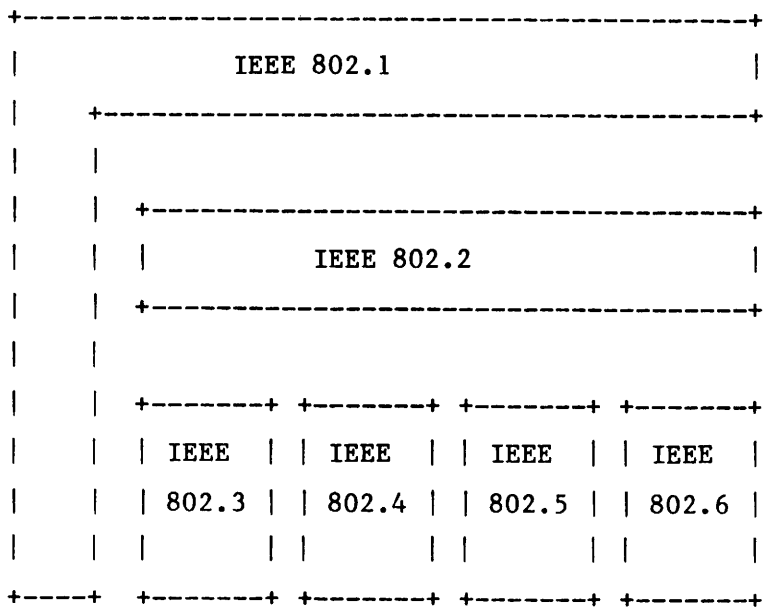
The Ethernet chips both allow for a very dense design of Ethernet controllers and a cheap implementation. The latter factor is important, as the nodes, which will be connected to an Ethernet, will be cheap microprocessor-based systems (see Killen, 1983). Still the cost for an Ethernet connection is moving around the 1000$ line, although it is expected that this cost will drop to 500$ by the end of 1983. As such, the price for an Ethernet controller will be in the same range as a traditional serial interface.

5.4.2 IEEE 802

The IEEE 802 local network committee was established in february 1980 with the purpose to establish a vendor-independent local area network standard before the Ethernet specification would be adopted by the industry as a de facto standard (see IEEE, 1982). Originally, IEEE 802 was the fastest working committee in IEEE's history with the objective to define the standard before Xerox, Intel and Digital officially announced the Ethernet specification (see Ether, 1980). The committee did not succeed in this task, and very soon the conflicting manufacturer, designer and user requirements made it obvious, that one standard could not solve all needs (see Datacomm, 1982).

The committee has then taken the approach, that it will standardize more than one type of local area network in a manner, so these standards are structured, so that the drawbacks in having more than one standard are minimized.

Hence, the IEEE 802 standard proposals embodies three media accessing schemes: CSMA/CD, token passing for bus topologies and token passing for ring topologies. This is done by defining a set of sub-standards, which are interfaced by a common standard, called the logical link control, LLC (see IEEE, 1982.2). The standards are organized in the following manner:



IEEE 802.1 is the guideline, describing the relationship among the other standards and their relationship to the ISO Open System Interconnect reference model.

IEEE 802.2 standard describes the logical link control, which constitutes the common interface between higher layers and the various interface standards for the physical layers.

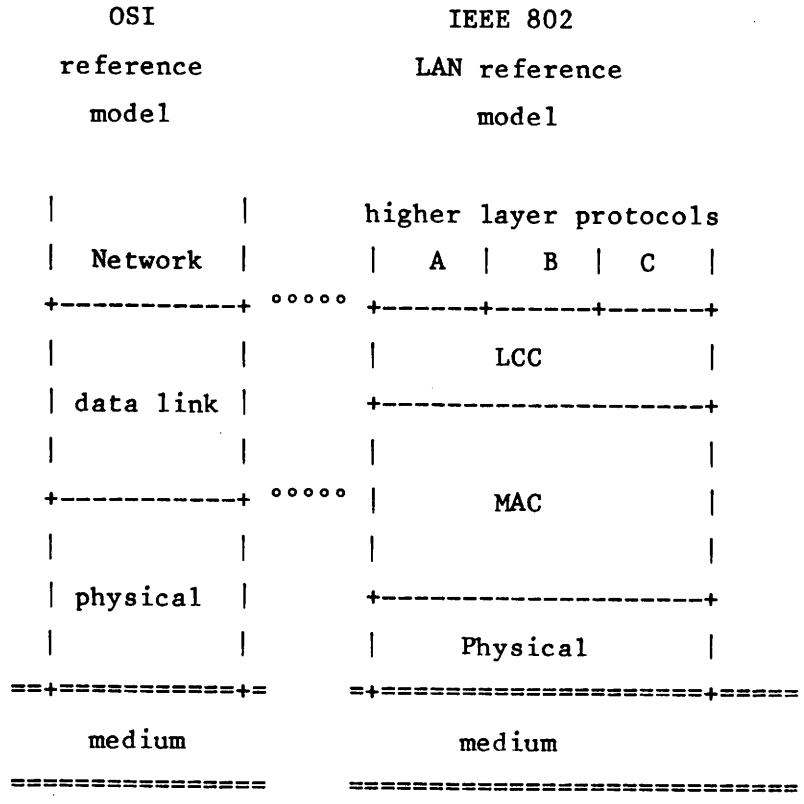
IEEE 802.3 standard describes a bus utilizing CSMA/CD as the access method.

IEEE 802.4 standard describes a bus utilizing token passing as the access method.

IEEE 802.5 standard describes a ring utilizing token passing as the access method.

IEEE 802.6 standard is currently under study and describes a metropolitan area network access method.

Seen in relation to the OSI model, the IEEE 802 standard has changed the organization of the lower layers. This has been needed to solve the problems of allowing different access schemes and different physical media:



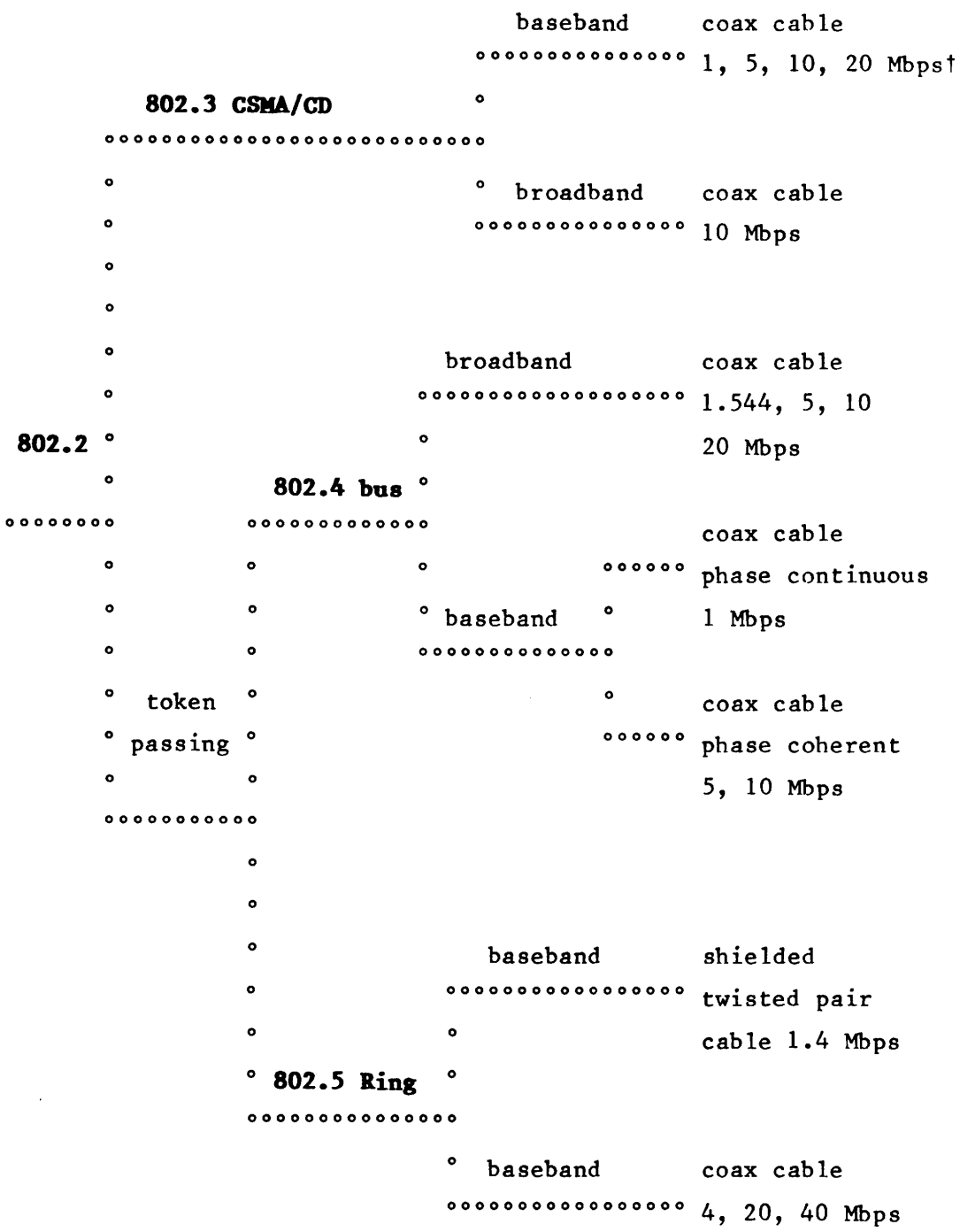
The IEEE 802 logical link control (LCC) provides the common interface to higher layers in the network. The LCC protocol between two LCC layers is very similar to the HDLC protocol known from the traditional OSI model. However, where the OSI model is sufficient for describing traditional type of network interfaces, a new layer has been introduced in the IEEE 802 standard to overlap both the ISO link control and physical layer.

This new layer is the Media Access Control layer (MAC). It supports the CSMA/CD and token passing access methods towards the IEEE physical layer. Also, MAC performs destination and source addressing of frames and control of frame check sequences.

The IEEE physical layer allows different types of media to be used under one of access protocols defined by the Media Access Control layer.

The sub-division of layers allows for a large range of different physical media and a large variety of local network implementations, which can address the various needs and requirements of local network applications, however still being in accordance with the standard specification.

This organization of the standards allow a user to define and implement applications which are independent of the actual physical media used for the local network. Each of the sub-standards IEEE 802.3-5 also allows different types of physical media implementation and speeds, according to the following figure:



[†: Mbps = Megabit/second]

IEEE 802 substandards and physical media combinations

5.5 Pure data networks ?

Most local area network implementations are exclusively designed to support data communication. This has led to a fraction within the local network market, which have started to implement other type of services on the network, like voice or video.

Generally, these new types of networks are based on broadband technology, where the broadband cable can support several independent network channels, which can be allocated for data transmission, voice transmission or video signals. The reason for doing this is to minimize the internal cabling of a building, as all signals in these networks share the same cable.

Basically, these networks offer a transport possibility for voice and video. What they do not offer, is the integration of voice and video with the data processing functions (see Wiley, 1983). On these networks, data transmission and voice transmission takes place on logically and physically different networks, which through the use of frequency multiplexing however use the same cable.

The need for voice and/or video transmission on the same network is still basis for discussion and research. In general, it is a question, whether one wants a network optimized for voice/video with occasionally data, or a network optimized for data with occasionally voice (see Rauch-Hindin, 1982.2).

The first place, where voice will be a significant factor, will however be on the pure data networks, like Ethernet and IEEE 802. This is due to the fact, that voice can and will be digitized. Digitization of voice means, that it can be handled in the same way as pure data, and hence a pure data network can

be used. Transmitting voice does however impose other requirements on the transmission protocols, than data exchange does. It makes no sense to have extensive error checking and retransmission facilities for voice data, as voice by its very nature is a here-and-now event.

Using a data network for voice transmission gives other benefits. As voice is being digitized and transmitted by a computer system, it can also be stored in a database system and retrieved for later use.

Digitized voice, however, is characterized by a large volume of information needed to reproduce the voice with sufficient quality. The most popular method for digitizing voice, the pulse code modulation, uses a bit rate of 64000 bits per second. To maintain a two-way conversation, a data network must be able to keep pace with a continuous data rate of 16 Kbytes per second, not including overhead. This can easily be maintained by the Ethernet and the IEEE 802 types of networks. Analysis has shown, that the Ethernet network can support up to 68 simultaneous conversations, using digitized voice by the PCM method (see Coleman, 1982). Using the compression techniques developed for modern voice response systems, the bit rate will be further reduced and hence the number of conversations increased.

As such, the local area networks designed for data transmission can support voice transmission for smaller systems, as well as providing an integration possibility, where voice can be treated as data.

5.6 SDC future use of local networks

Local area networks will be needed in future branch office systems to interconnect nodes with internal processing power. The requirements that SDC must impose on such networks will be:

- low cost, to allow any component to be connectable to the network, without requiring the use of centralized systems

- flexibility, to be able to continuously upgrading the branch office system without the need closing down or modifying the rest of the system, when new components are added.

- simplicity, to minimize the administration and installation effort.

The use of a local area network standard cannot be discussed, as this will be the only way to maintain compatibility over a certain period of years and to ensure, that the network will not be dependent on one manufacturer.

Most non-standard local area networks embodies facilities, that cannot be used by the savings banks. As an example, a facility which allows for several thousand terminals to be connected is of no use. Likewise, networks needing a central controller is also a bad way to go, as the individual network within a single branch is actually small, making a central controller to expensive, as the cost must be divided over a small number of workstations.

The following table shows the current distribution of branch offices as a function of the number of employees. As can be seen, more than 60% of the branches have less than 7 employees. Only 13% has over 10 employees. The number of employees can be taken as a coarse estimate of the number of workstations needed in a branch office. Most branch offices will have a workstation per employee, plus some teller workstations. Besides this, a branch may contain some network un-manned components like file servers, print servers and communication servers towards the public data network.

As a result, the number of devices connected to a local area network within the branch will be small. Less than 20 devices will be connected in the majority or cases.

number of branch offices with	1)†	2)†	total empl.	# of	estimated branches	
1-2 employees	111				185	12.2%
3 employees	106	374	935	3100	177	11.7%
4-6 employees	344				573	37.9%
7-8 employees	188				249	16.4%
9-10 employees	103	94	578	6000	136	9.0%
more than 10	193				193	12.8%
totals:	1045	468	1513	9100	1513	100.0%

(† source: Dfs1,1983; see chapter 2 for notes)

Then, will a local area network be sufficient to serve the needs of a branch office system? If we suppose, that a branch office system will support a number of users, each with their own workstation with the required processing power, we can make the following assumptions:

Each workstation will need the load of a small application program of 512 Kbyte through the network every third minute. Transactions and responses over an interface to the public data network will be generated every 10 seconds by every user. Different file updates on a local network file server will happen every 5 seconds per user. Small message exchanges will be generated every 100 milliseconds by applications.

The following table shows the network utilization under the circumstances above for a 10 Mbit/second network (like Ethernet), where it is expected, that the network can be effectively used 75 %.

Bits per second	10000000			
efficiency	75%			
	SDC transact.	file updates	program load	message exchange
access interval	10	5	180	0.1 seconds
access length	500	2048	524288	100 bytes
number of users	50	50	50	100
header overhead	35	100	35	35 bytes
link capacity used	0.29%	2.29%	15.54%	14.40%
accesses/second	5.0	10.0	0.3	1000.0

Obviously, the largest utilization comes from the program load, whereas the transaction exchange and small file updates hardly affects the network. Message exchange between application will hardly ever occur with the rate shown, but even those do not impact the network significantly.

The figures above were for a large system with 50 nodes. For smaller systems, more representative for the majority of branch offices, the figures are like this:

Bits per second	10000000			
efficiency	75%			
	SDC transact.	file updates	program load	message exchange
access interval	10	5	180	0.1 seconds
access length	500	2048	524288	100 bytes
number of users	10	10	10	25
header overhead	35	100	35	35 bytes
link capacity used	0.06%	0.46%	3.11%	3.60%
accesses/second	1.0	2.0	0.1	250.0

Ironically, by choosing a local area network approach, it makes no sense to go into detailed analysis to optimize the network further.

5.7 Conclusions

Local area network standards will be the means to interconnect processing modules in systems of the future. The number of nodes per local network in the savings banks will be rather limited and hence SDC must choose a concept, that has been designed for medium size of systems as well as for low cost, as the use of any central control facility will give the same problems as in the current TP2 system.

Two possibilities exist for this: the Ethernet and the IEEE 802 standard. The Ethernet is available today, whereas IEEE 802 has not yet reached the final specification stage. However, as a long term decision, the IEEE 802 will provide SDC and the savings banks more degrees of freedom in adapting the use of local networks to the actual requirements, while being able to keep a common interface at the higher levels. Anyhow, IEEE 802 will probably be able to support Ethernet using the IEEE 802.3 CSMA/CD sub-standard (see Datacomm, 1982.2).

As such, SDC should follow up very closely on the IEEE 802 standardization efforts, while in the meantime, it should use Ethernet to gain the internal know-how on how to adapt other parts of the systems to a natural usage of local networking capabilities.

The current tp-monitor concept of TP2 should be enhanced with a presentation level strategy, that allows inhomogeneous systems to exchange data, as well as the lower layers of tp-monitor should be analyzed to take into account, that they must be used in conjunction with local area network layers in the future.

This page intentionally left blank

CHAPTER SIX

BACKPLANE BUS ARCHITECTURES

6.0 Introduction

Chapter 3 gave an overview over the progress in chip technologies and microprocessor architectures. The results were the following:

- denser circuits
- computers-on-a-chip
- larger memory capabilities
- cheap processors
- complex I/O
- change

The utilization of these circuits is done by collecting them together on circuit boards, where the interconnections between the individual components are placed as wires. Many microcomputer systems today are implemented on one single circuit board, where all components from processor, memory to I/O are placed. This approach can be justified for many simple systems, and is indeed valid in the personal computer market, where price must be low, and where computers are disposed and thrown away, when new and more advanced models comes one the market. Change is not a keyword for this type of systems.

Change is the problem in designing larger computer systems. The rapid innovation rate of components gives possibilities for many type of new applications of computers, but to cope with this change, fixed and static hardware design fails. As systems grows, complexity does not increase monotonically with system

size, but rather exponential if some kind of structure is not applied to the system. An analogy can be taken from large software systems, where this type of problems are well known.

Structured programming or modular programming are tools, that are used to control and ease software development. The same type of tools must be applied to hardware design to control and maintain a constant utilization of new concepts.

Modularity in hardware is represented by designing the system as a set of modules connected together through a common electrical highway. This highway is commonly referred to as a **bus**.

Buses can occur at many levels. In hardware design, implementors are normally faced with component level buses and subassembly level buses, also called backplane buses.

6.1 Component level buses

Component level buses are generally a set of wires and conventions used to link together the individual integrated components. No standards exist in this area, except some electrical 'standards' (like TTL, ECL, CMOS) specifying the electrical characteristics of signals on individual lines. One attempt has been made to make such a standard in the past. This was the Microbus from National Semiconductor, but this never won acceptance and has been aborted. Several 'de facto' standards exist, which are linked to one manufacturer or one special type of microprocessor, and which are not independent in any way. Examples of this are the Intel 8080-type of interfaces, Motorola's 6800-signals and Zilog's Z-BUS or ZCI (Zilog Components Interconnect).

We can ask the question, whether a standard at this point is wanted or not. The fact is, that most systems do overcome the problems of interconnecting components from different manufacturers, as it is only a matter of a small amount of additional logic to interface non-dedicated chips.

Also, the design of a specific subassembly (e.g. a circuit board) of integrated components are normally done by one manufacturer, and does not have the need to be able to exchange components on the circuit board. Rather the circuit board will be redesigned.

The need for modularity occurs at the board level, where system designers need the flexibility to implement a system based on different circuit boards from possibly different manufacturers, where these circuit boards are connected together via a backplane. This facilitates the need for backplane bus standards.

6.2 Backplane buses

Standards for backplane buses can be considered to consist of the following items:

- a mechanical format of the board
- electrical specifications
- a set of protocols
- a set of signals, which attempts to minimize the circuitry required to interface one or more commonly used microprocessors to the backplane

The design of backplane buses addresses the various needs for the use of such a bus. Two basic approaches exist. The first views the backplane as a pathway between the processor and its memory and I/O devices, i.e. as an instruction execution or instruction fetch path. The second views the backplane as a communication path between processors, i.e. as a message exchange path. This difference also occurs in time, as the latter is the type of newer designs in buses, as technology has made it possible to integrate the actual processing capabilities together with the needed memory and I/O on each individual circuit board.

The use of backplane bus standards has been largely accepted in the past five years by American companies. The reasons are many:

- new components can be utilized in existing systems earlier
- market share is greater
- systems can be designed with off-the-shelf boards from different manufacturers, allowing the designers to add value to the system through software or specialized boards.

However, many European companies has not seen this, and use efforts and costs in implementing their own type of design, in appraisal of the NIH (Not Invented Here) principle.

Standards will not live forever. Technology progress will make standards obsolete in the same way as systems. The first backplane bus standards were defined on the basis and requirements of the 8-bit type of microcomputers. As 16-bit microprocessors were announced, the need arose to enhance these standards or even to define new ones. This chapter is about this.

6.3 Old type of backplane buses

The old type of backplane bus standards originate from the early days of microprocessor design. All of them originate from specific manufacturers or designers. No official or theoretical analysis was done originally on them. Nevertheless, they are the basis of the entire microcomputer market.

6.3.1 The STD bus / IEEE P961.

The STD bus (see Elsmore, 1983) dates back to 1973, where Pro-Log defined the first version in an attempt to chose a card size, that allowed modular convenience to simplify design. The first version was a mess, where system I/O lines were mixed with the processor bus on the backplane. Pro-Log redefined the bus in 1976 and restricted the backplane to the processor bus and required I/O to take place at the opposite card edge.

In 1978, Mostek joined Pro-Log and these two companies finalized the specifications. The bus specification was given free for the market and in 1979 the STD Manufactures Group (STDMG) was founded. This group monitors the usage of the bus in order to maintain order and make recommendation on usage, so

all boards from all manufacturers will work together.

In 1981, the STD bus was taken up by the IEEE Microprocessor Standards Committee (MSC) and given the project number P961 in order to make it an IEEE-standard. In 1982, work started on defining a STD bus proposal implemented on a Eurocard DIN connector.

The STD principle has been that of system partitioning by function, i.e. CPU on one board, memory on another, I/O on a third, etc. The benefit of this comes in applications where cost is a key factor, as it allows a highly modular system using small cards.

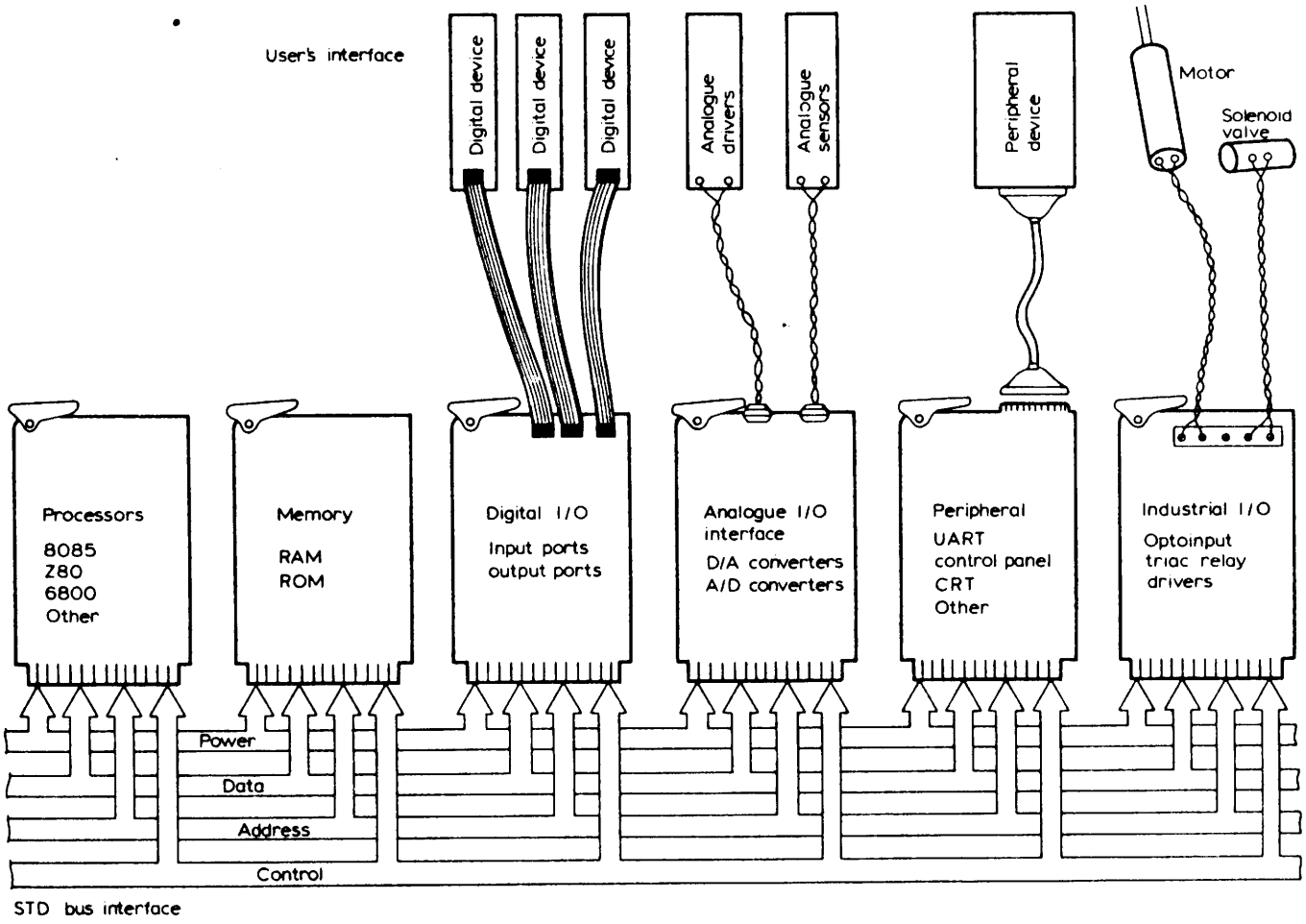


Figure 6.1: The STD bus concept

Basically, the STD bus is an 8-bit bus, although proposals are under consideration for the Eurocard-version to allow 16-bit operations on this.

6.3.2 The S-100 backplane bus / IEEE 696

The S-100 bus (see Garetz, 1983) has been one of the most discussed and condemned standards in the backplane world. It originated as an amateur bus, with an ad hoc collection of silly signals and formats. It has been copied and modified by so many manufacturers, that they were no longer compatible, and then, in 1979, it was taken by a group of enthusiasts and transformed into a single technically competent and usable standard under the IEEE (see Stewart, 1983). The S-100 name comes from the fact, that it has 100 pins on the connector.

The IEEE 696 bus, as it looks today, supports both 8- and 16-bit processors, 16 Mbyte of memory, 16-bit I/O addressing and is virtually processor independent. More than 100 manufacturers are delivering boards for the bus and there is more than 500 different boards available.

6.3.3 Multibus / IEEE 796

The Multibus™ was originally designed by Intel in 1974 to support the 8080 type of microprocessors (see Harrison, 1983). At that point in time, it reflected the current state of technology, like defining the need for multiple power supplies of +5, +12, -12, -5 and -10 Volts. The -5 and -10 V has been abandoned in the newer versions of the bus, which also implements 16-bit transfers and 24-bit addressing.

The IEEE started analysis of the Multibus in 1978 and a proposal was ready in 1980 (see Boberg, 1983). It was not approved as an IEEE standard until 1982, because of a dispute on nomenclature: The original Intel specification and the proposal used hexadecimal notation for address and data bus signal lines, whereas the IEEE computer standards committee proposed decimal notation. The problem was solved in 1982 by the addition of a paragraph stating that boards and documentation created before December 1987 using the hexadecimal notation are considered compliant with the standard (see Stewart, 1983)|

This is just an example on the kind of issues, that may be of great importance in specifying bus standards. In the meantime, Multibus has become one of the most used bus standards with more than 120 manufacturers producing more than 800 different products.

6.4 Future directions

The IEEE 961 (S-100) and the IEEE 796 (Multibus) are examples of system buses used in the majority of small computer systems today. The continuing support of these buses means that users can augment existing systems with more processing power, new peripherals or new types of interfaces without major problems.

Being a standard is one reason, but another important factor is the physical board size, which for all the buses above are comparatively small. This means, that actual board design is a minor task, where the functions on each board are limited to the amount of chips, that can reside on the board, and where routing problems are less than on larger boards. This again makes the design cycle for a board shorter, which means that board products comes on the market earlier.

During the standardization efforts of the bus standards, requests have been made to increase the board sizes for the reasons of increased functionality and resources to increase the number of applications satisfied by one board and to increase the cost/effectiveness of a bus based system by integrating more functions on one board, thereby sharing components like bus drivers and in some cases reducing the total system board count to one.

When it comes to figures, it was however impossible to find any, who could present a complete analysis of and the solutions for the technical problems and the economics involved in having additional board sizes, and thus the standardized buses have remained small without any noticeable effect on manufacturers (see Boberg, 1983).

These factors should be seen in contrast to the backplane architecture of the TP2 system, where the bus structure of the local computer is fixed, static and proprietary bus, using large boards, which cannot be redesigned without a major effort, where memory addressing cannot be enhanced, and where new boards will never be designed. Also taking into account, that current 16-bit micros all outperform the processor of the TP2 local computer, the fact is, that the general microcomputer market is offering system solutions with more power, more flexibility and - most important - a well-supported migration path to newer technologies, just because of the usage of current bus standards.

The current bus standards above will however not support all of the needs in future microprocessor based systems. The reasons are:

- they are 16-bit buses, where next generation of processors will be 32-bit

- they are primarily instruction fetch buses, where next generation of systems will have sufficient on-board local memories for instruction fetching

- next generation of processors will have facilities for protection and operating system support, that is not supported by these buses

- next generation of processors will allow for large physical addressing than the 24-bit (16 Mbyte), that is maximum for the current buses.

This is why, next generation of systems design will be based on new types of advanced bus standards.

6.4.1 TUBUS

In the first part of the EF-57 study (i.e. 1979-80), such standards were not available. At that point in time, our target was to produce a prototype of a future type of processing node for banking systems, based on the new processor technology. I realized, that the most important task in this process was to define a suitable 'standard' - a bus specification, which took into account the new possibilities.

One must remember, that the only experience, that SDC had in this area, was the previous design of the McTan-bus, an 8-bit backplane bus for SDC's Z80® based McTan computer system designed in 1978 by Anders Nielsen and myself. This bus was designed from scratch, without any previous knowledge on bus design, electrical characteristics or protocols. It was however, according to Zilog, the first bus design able to run

the 4 Mhz Z80 chip at that time.

The new bus design was called the TUBUS. The design goal from the start was the following:

- it should support 32-bit operations
- it should support the Z8000™ family of processors
- it should use new concepts
- it should use Eurocard board sizes and the Eurocard DIN connector

The only starting point was the pin-descriptions of the Z8000 microprocessors and the proposed Z-BUS component interconnect bus from Zilog. This was the only hints to indicate the type of future directions of the Z8000 family towards 32-bit processors.

The actual ideas behind the TUBUS are described in the accompanying document, The Z8000 TUBUS design (see Bertelsen, 1980.2). TUBUS itself is described in a separate specification document (see Bertelsen, 1980.1).

The features of TUBUS to be discussed here include:

- the mechanical standard

  the choice of the Eurocard format and the indirect DIN connectors was then not generally accepted. Buses then available was done on American board formats and with gold-plated finger-edge connectors, and this was by many considered to be the optimum choice. We rejected that approach, because for the reasons stated above, we needed the small board sizes, and we needed more signal pins than on the current buses. Finally, we considered the indirect DIN connector more reliable. Since then, these thoughts have been adopted by others (see

Elsmore, 1982), and today we can find more than 60 different microcomputer buses based on the DIN connector, more than exist on all other connectors put together (see Borill, 1982).

Today, proposals are underway for putting both the S-100 and the Multibus on the DIN connector.

- Electrical design

The TUBUS design was made a 5-Volt only bus, as we could forsee the progress towards 5-Volt only chips, and as other voltages, like +/- 12 Volt for RS-232 drivers, could be easily generated by on-board integrate voltage converters. Also, proper grounding and decoupling was taken into account, a feature not previously found in microcomputer bus designs.

- Logical design

TUBUS was made to support 8, 16 and 32-bit transfers on the bus. It introduced the concept of separated physical and logical addresses on the bus, allowing systems with distributed memory management units. It separated bus arbitration for DMA transfers and multiprocessor access on the bus, allowing different arbitration schemes and priority schemes for the two types of bus usage. In that sense, it should support both the instruction fetch type of usage with one processor board connected to memory and I/O boards, and the message oriented type of usage with individual processors on each board, possibly connected to a global memory.

During the work, we found that we were the only Danish company actually involved in bus design, and thus we were not able to get much feedback on our work. The first major feedback and inspiration came with the announcement of the Zilog ZBI™ backplane interconnect and the Motorola VERSAbus™ in 1980. These two buses showed, that our initial thinking would prove right: The ZBI was a 32-bit bus, implemented on a single Eurocard connector, but not with the same facilities as TUBUS and with built-in potential noise problems. The ZBI today is primarily used by Zilog itself in their UNIX™ based systems and by Exxon in their Office Automation Systems. The VERSAbus was also a 32-bit bus, oriented towards the 68000 microprocessor, based on finger-edge connectors on huge circuit boards. The VERSAbus is now losing its market share to the VMEbus™, which is approximately the same, just based on Eurocards.

The TUBUS design did however cause feedback from outside. First of all, AMD showed interest, when I had discussions with their bus-designers during my visit in Silicon Valley in 1980. At that point in time, they were trying to define their own standard, called the AMD Bloc-Bus (see AMD, 1980), but they never really succeeded in that. Secondly, I came in contact with Andrew Allison, who was then the chairman of the IEEE P896 committee, which had then started the definition of the 'future bus' standard proposal, P896, which should be an advanced 32-bit backplane bus to support future processors in the next ten years. After having seen the TUBUS documents, I was invited to participate in the IEEE P896 committee, which I joined late 1980.

Before discussing the IEEE P896 bus, it will be worthwhile to have a short look at the alternative bus standards.

6.4.2 Zilog ZBI

The ZBI™ (Z-Bus Backplane Interconnect) is not interesting in the terms of standardization, as its usage is mainly restricted to Zilog and Exxon. This originates from the fact, that shortly after the ZBI was introduced as a concept, and a series of ZBI-compatible boards was announced by Zilog, Zilog suddenly dropped plans for selling boards on the board-market and restricted board-manufacturing for internal usage only. This was mainly a matter of organizationally and resource problems within Zilog.

The interesting idea about ZBI is the fact, that it implements a fully 32-bit bus on one single Eurocard connector. This potentially allows for very small boards, and also the bus supports both 8, 16 and 32 bit processors. This means, that as long as memory boards and controllers support 32-bit transfers, all types of processors can use this memory through the same bus.

In order to do this, a complete 3-row Eurocard connector is used. The 3-row connector has 96 pins in contrast to the more used 2-row connector with 64 pins. The bus uses a lot of pins for power voltages (+5, -5, +12 and -12 Volt), status signals and interrupt and bus acknowledge chains, so even though the 32-bit data and addresses are multiplexed, the 96 pins are needed.

The bus is primarily an instruction fetch bus, where one master processor board uses to bus to communicate with memory and I/O, although I/O is intended to be performed by intelligent controllers. There exists however a set of control signals on the bus, that can control several processors usage of a single, common resource, e.g. a disk controller, and these signals are supported by synchronization signals and instructions in the Z8000 CPU, but until now, no systems have been seen, that uses

these signals. The normal board format of ZBI boards are double Eurocard, where the second connector is user definable, and commonly used for I/O.

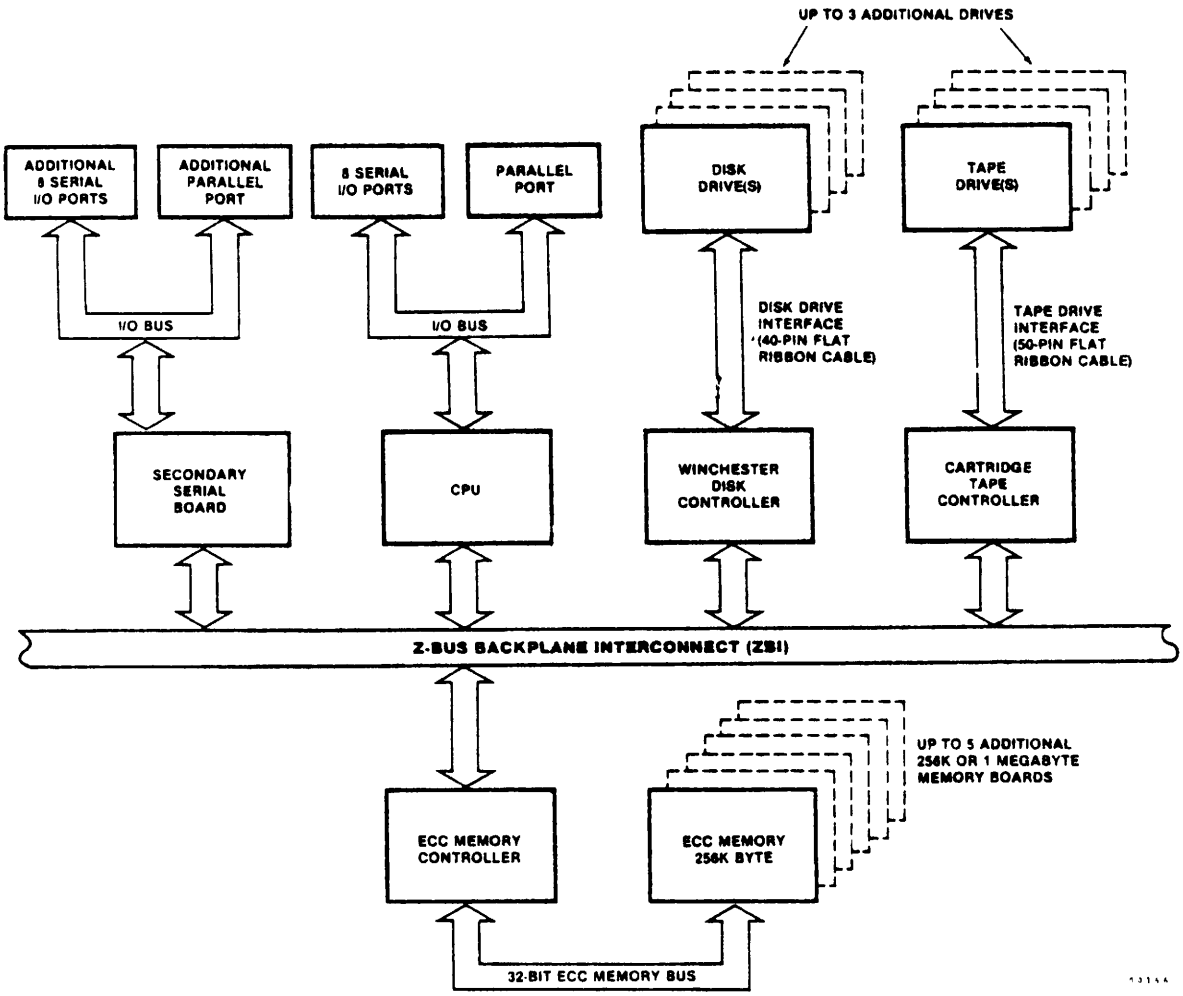


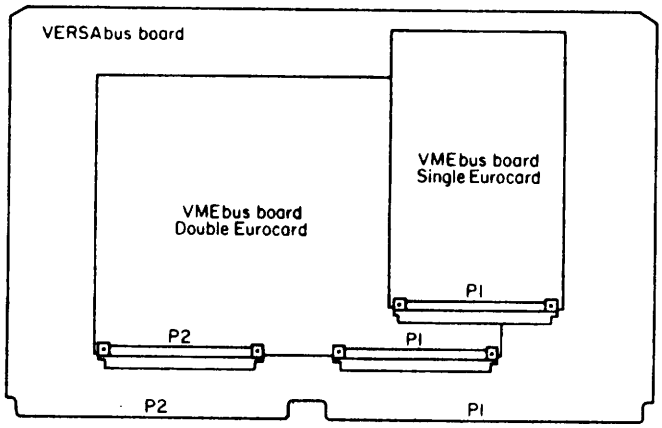
Figure 6.2: ZBI bus structure

As such, the ZBI fulfill its intended purpose, and it is sufficiently general for the traditional type of business systems, that is implemented with a single, central processor board and a certain amount of memory and peripheral boards.

6.4.3 Motorola VERSAbus and VMEbus

The Motorola VERSAbus™ was designed to support the 68000™ microprocessor as the ZBI was designed with the Z8000 in mind. The VERSAbus is however some more general in its concept, which has been one of the reasons for its acceptance. Another factor is its early introduction at the same time as the 68000 went into mass production. Actually, the VERSAbus can be considered as an offspring from the IEEE P896, where many of the initial ideas and suggestions in P896 has been carried over to VERSAbus by IEEE committee members.

VERSAbus is physically a huge bus, using a board size of 23.5 times 36.2 centimeter. This have caused considerable interest in obtaining a smaller version, which has been defined as VMEbus. VMEbus is based on single and double Eurocard sizes (10 times 16 centimeter and 23.3 times 16 centimeter), and today the VMEbus is seeming to win the race. VERSAbus and VMEbus are almost identical. The major differences are, that VERSAbus contains parity on data and address lines, where VMEbus does not, and that VERSAbus have one more bus request level than VMEbus. Also VERSAbus has pins for analogue power.



Board size comparisons (component side)

Figure 6.3: VERSAbus and VMEbus board sizes

Logically, VERSAbus and VMEbus consist of five separate buses: a data transfer bus (DTB), a priority interrupt bus, a DTB arbitration bus, a utility bus and a interintelligence bus.

The data transfer bus comes in two version. The basic version is a 16-bit data bus with 24-bit addressing. The expanded version using a second connector on the VERSAbus and requires the double Eurocard VMEbus, implements 32-bit data bus and 32-bit addressing. Separate paths exist for data and address in contrast to the ZBI, where these are multiplexed.

The interrupt bus is extensive, where a seven level interrupt structure is implemented with interrupt acknowledge through a daisy chain. This structure has its origin in the old type of interrupt architecture from the 6800 microprocessor, but has however implemented the type of truly vectored interrupts known from e.g. Z80 chips.

Multiple masters can utilize the bus. As such, VERSAbus and VMEbus are not only instruction fetch buses, but also serve as a more general type of common resource in multiprocessor systems. The implementation of bus arbitration is however still centralized through a daisychain, where the first master in the daisy chain has the highest priority.

The interintelligence bus is intended for multiprocessor applications. The reason is, that in multiprocessor systems, the data transfer bus can become a bottleneck if multiple masters shall communicate through a common memory together with other bus transfers, like disk to memory transfers. The interintelligence bus is a two-wire 4 MHz serial bus, much like a short local network, that is to used to transfer messages between intelligent boards without invoking the data transfer bus.

An asynchronous protocol is used for bus transfers, which is a benefit, especially in migrating systems, where faster boards shall communicate with older, slower boards.

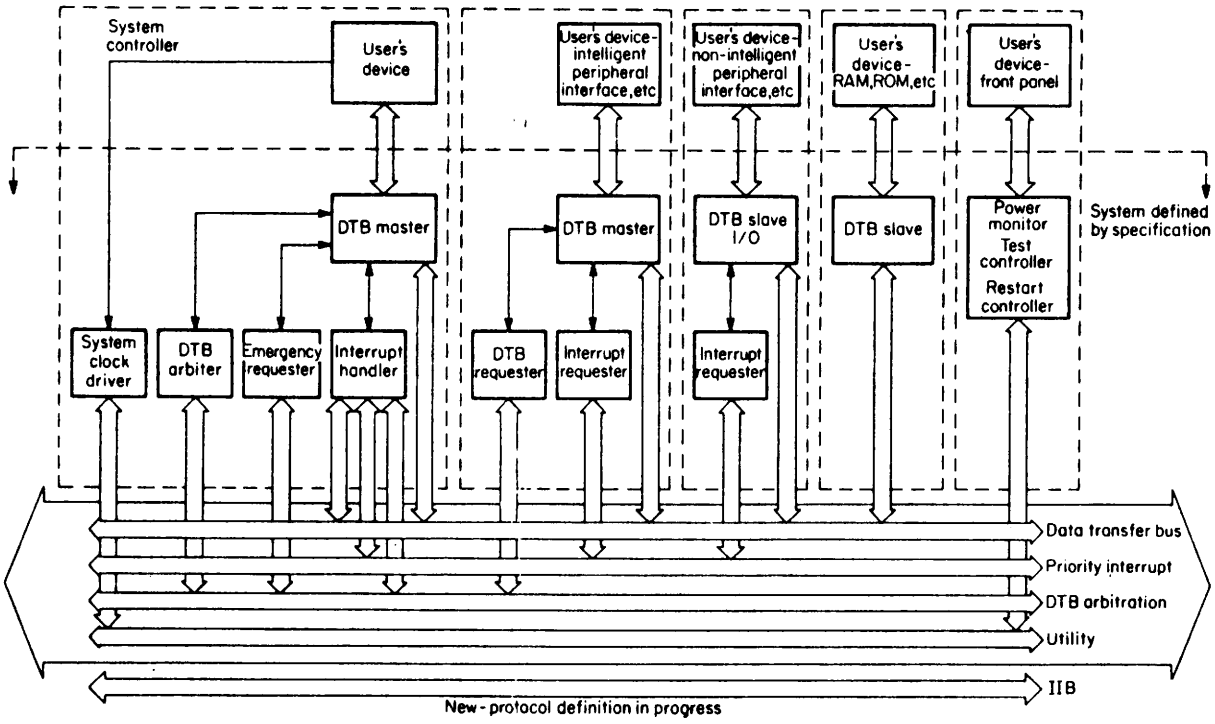


Figure 6.4: Modules and buses within VERSAbus and VMEbus

For users wanting to go into system design today (mid 83) with a standard backplane bus using 16 bit processors, but with a clear built-in migration path to 32-bit, the VMEbus is the only available alternative, that has gained industry acceptance, and is off-the-shelf available from many independent suppliers. It is still not an IEEE standard, but this will eventually happen (VERSAbus is being considered by IEEE under the name P970).

6.4.5 IEEE P896 - The futurebus project

P896 is not a de facto standard. It is not a manufacturer's standard. It is a try to create an a priori standard for the future types of multiprocessor systems.

In 1978, the IEEE decided, that a new bus standard was needed to support the new types of microprocessor systems, that was anticipated to be constructed as multiprocessor systems. This effort should both prevent the need for another generation of de facto buses from the manufacturers and prevent the need for yet another future bus standard in at least a decade.

The formal project of standardizing the P896 bus was started in 1979.

These are the goals and objectives of P896:

- independent of processors and manufacturers. The bus will allow different processors to be used in the same system, and the attempt is to equalize the interfacing penalty over all manufacturers processors. The bus will also be technology independent, i.e. not restricted to TTL or CMOS.

- independent of system architecture. This is probably the most important fact of the P896. It means, that true multiprocessor systems can be implemented, using a distributed control philosophy, which is nor reliant on any specific circuit board. It is achieved by

  - no daisy chain arbitration
  - no interrupt scheme
  - communications-oriented architecture instead of instruction fetch architecture

- no implied constraints on operating systems software
- support of multiple parallel buses

The traditional instruction fetch architecture can still be implemented on the bus, although such an architecture probably would take more benefit of using one of the existing bus standards.

- High performance. The bus is designed for a data transfer rate of more that 10 Mtransfers/second. Being a 32-bit bus, this allows for a bandwidth of 40 Mbyte/second. In comparison, the VERSAbus and VMEbus has a bandwidth up to 20 Mbyte/second.

- High fault tolerance. The design objective of the bus for data transfer integrity is one correctable error per day and one uncorrectable error per year. Design objectives include support for fault detection, diagnosis and system reconfiguration. Also features for low mean-time-to-repair are being considered, like live insertion and removal of boards.

- available with mid-1983 technology, which means that the bus does not depend on the development of new devices or new technologies. This will allow for fast implementation, as soon as the standard is finalized.

- Mechanical implementation will use the DIN 41612 connector (603-2-IEC-C096-M) and the IEC 297 SC48D card size family, better known as Eurocards.

- The P896 goals will not be constrained by cost. This is done to take advantage of the technology available under the assumption, that integrated logic is essentially free. It does not mean, that cost is not

considered, but the bus is not optimized towards very small or very cheap systems.

These goals, however obvious they may look, has caused many problems in the P896 design, and has delayed the standard many times. One of the reasons is the way, a committee is organized and work. Participants in the P896 committee have come from many different types of environments, semiconductor manufacturers, universities, computer companies, industrial control companies. In this sense, it is interesting, that SDC has actually been the only user representative present in the committee. The many backgrounds have also issued different objective proposals, and has caused the work to virtually stop during the years. One remarkable factor has been the participation of universities, which tried to introduce some very odd ideas, which could be used as a basis for theoretical theses, but would conflict with the general scope and industrial applications of the bus.

My participation in the committee gave me a lot of experience on this type of work. My own TUBUS had been done by very few people. In the IEEE committee, more than 50 people participated with their own views and capabilities, and it showed me, that a single person is unlikely to be able to design all aspects of such a bus himself. It also showed me, that previous ideas on how things should be designed, no matter how well they have been analyzed, does not have effect, unless the person is open-minded and willing to change his ideas when other persons begin to argue.

Another noteworthy point is, that participating in such a committee actually takes a lot of time and that active participation is done best, if one can come to the committee meetings. Although a European subgroup is present within the P896 committee, most working meetings takes place in the United States. This can be a problem, if the company, where one is

working, are not willing to invest the amount of travel expenses needed for attending such meetings.

Together with me, Claus Ringborg from the SDC Microlab group also participated as co-worker. When the McTan X.21 project was started (see chapter 10), all our spare time within Microlab in SDC was concentrated on completing this project, and it actually did leave no time for concentration on P896, which at that point in time was demanding close attendance, as several conflicts had grown up. These conflicts resulted in a total reorganization of the P896 project in the start of 1982, where the old chairman resigned and different non-P896 proposals arrived (see I896, 1982). As a consequence of the missing time to do a proper work, Claus Ringborg and I dropped further activities on P896 in 1982 and decided just to follow the progress until we could find room and schedule for more permanent activities.

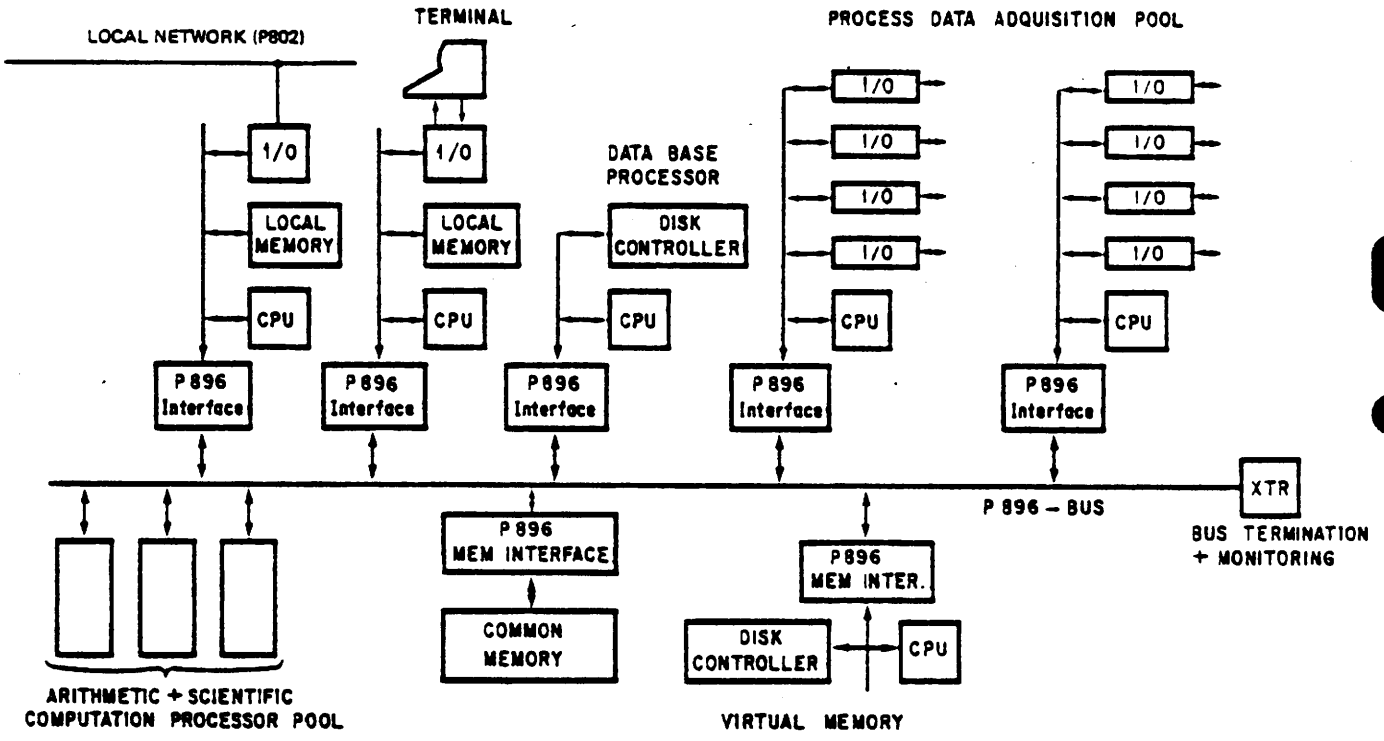


Figure 6.5: IEEE P896 backplane concept

In the following, some of the problem areas in the P896 design will be shortly discussed to show some of the many factors playing a role in backplane bus designs.

The discussions have concentrated around:

- Arbitration concepts, i.e. how different processors get access to the bus in a system with multiple processor. One of the design parameters here is the number of processors, that should be supported as well as the concept of fairness, i.e. that no processor should be able to monopolize the bus (see Borrill, 1981.3; DelCorso, 1981.3; EDISG, 1981; Lyman, 1981; Lyman, 1981.2)

- Bus protocols, i.e. how data is transferred on the bus (see Nicoud, 1981; Logg, 1981; Kirrmann, 1981.2; Gustavson, 1981.2; Borrill, 1980.2).

- Supervisors, a concept, that was proposed to be included in the bus, which could go in and interact with data transfers and modify data or cycles, as a means to ease debugging and configuration changes (see Dumoulin, 1981; DelCorso, 1981.2; Gustavson, 1981; MacKenna, 1981). This concept, which would have implications of system security, was eventually dropped.

- Control, i.e. whether control should be distributed or centralized (see Muelemann, 1981; Deiss, 1981).

- Electrical considerations, i.e. the type of drivers and termination networks, noise and speed (see Borrill, 1981; Gustavston, 1981).

- Other items as leveling of the bus specification, the use of a serial bus on the backplane (see Beuchat, 1981; Kirrmann, 1981), whether 64 or 96 pins on the Eurocard connector should be used, whether 24-bit addressing would be sufficient, justification of bytes within words and factors like terminology and notations.

- The most important trend has come in the adaptation of the ISO OSI reference model for open systems interconnect. Applied to a backplane, it means that the backplane interface is divided into layers.

  This division allows layers to be changed if needed, i.e. the physical layer on the P896 bus could be realized with TTL-logic, CMOS-logic or ECL-logic, while keeping data transfer protocols in the 'logical link' layer intact. The approach is actual quite simple and obvious, but it took a long time before anybody came upon the idea in P896.

As such, the P896 backplane bus represents a suitable backplane bus standard for many of the futures new microprocessor systems. It will certainly be a target for many designs based of the new generation of 32-bit microprocessors (see chapter 3), which will need the features of P896 together with the enhanced addressing capability of the bus. However, one contributing factor for its success will be the availability of support circuits, e.g. bus drivers and arbitration circuits, from the LSI-manufactures, because they will be needed to allow smaller boards and a reasonable amount of the space on these boards to be used for processing logic instead of interface logic (see Borrill, 1980; Hersch, 1981).

The current state of P896 is, that a new draft version should be ready for comment by the end of 1983. As the work was started in 1979, it shows the time lag inherent in these standardization efforts, which tries not to be reliant on just one manufacturer. Secondly, it shows, that not all concepts within bus design can be clarified from the start, and that no single approach will solve all problems. Standardization is a matter of compromise, not a matter of doing the perfect research for just one application.

6.5 Related bus standards

The hitherto described standards have covered the area of interconnecting circuit boards into a system. The same criterias of modularity and flexibility will also match with another family of hardware-oriented standard buses, namely the peripheral buses used to connect peripheral devices, like disks or tapes, to a board controller. These standards have emerged as a result of the rapid innovation in the peripheral market, where users need to be able to upgrade their systems with e.g. new disks, without having to replace controller boards. A driving factor has also been the trend to put microprocessors within the peripheral devices themselves, so the data transfer between these devices and the system takes place in a parallel form, rather than having a direct wire to a read/write head.

The result has been, that the controller boards which connect to the backplane bus, can be made more general and more independent of specific types of peripherals.

Still progress is being done in defining those standards, although some has already become industry standards. This is done for the disk area, where the SASI bus (see Barker, 1983) and the SMD (Storage Module Drive) interfaces are commonly used. Other standards to become effective are the Enhanced Small Disk Interface (ESDI) (see Warren, 1983), the ANSI X3T9 I/O interface, the Local Distributed Data Interface (LDDI), the Small Computer System Interface (SCSI) and Intelligent Peripheral Interface (IPI) (see Parker, 1983.1).

Today, SDC cannot easily take advantage of the peripheral developments in the TP2 system. Future system designs must therefore take these standards into account.

6.6 Conclusions

The design of a microprocessor-based computer today is done by the use of a suitable backplane bus standard. Such a standard allow for a technological migration within the computer system as well as the availability of system components (boards) from a wide range of independent suppliers.

Many of tomorrows systems will not be based on such a standard, as the technology allows for computers with sufficient performance to be integrated on one single board, but in the case where multiple board systems are needed, the use of a versatile bus standard is a must.

None of the existing TP2 equipment allows for the usage of other vendors buses, and current and proposed buses from the supplier of the TP2 system, Olivetti, will not be supported by any in the future. A reverse effect of this is, that none of the existing hardware boards can be utilized in new systems with higher performance processors.

SDC can choose to follow this approach, or we can choose to specify the usage of a standard backplane bus for the next generation of systems. There will be two advantages in that:

- SDC basic problems in system design lie in the interfaces towards the environment. This include screen controllers, communications controllers and disk/tape controllers. If such controllers are non-intelligent, i.e. without their own processor and memory, an instruction fetch architecture has to be used, and then the transportability of the interface is dependent on the main processor used. Using intelligent controllers, these will contain the needed software for handling the interface, and the access to the board will take place

through the standardized message exchange protocols defined for the backplane bus. One example is a X.21 controller, which in SDC's case would contain all the protocol software needed for X.21, HDLC, Path Control, End-To-End management, File Transfer etc., and thus could be used over the years in any type of system employing the same bus standard, but with different application processors.

Also, as new technologies emerge, which requires new input/output types, like voice modems, optical disks etc., these can be integrated into existing systems just by adding the needed boards together with the suitable piece of applications software.

- SDC can employ different system configurations adapted to the various needs of individual savings banks. Different types of processors can be used to adapt the processing power to the users need. Simple systems may need only two or three boards, but these can be augmented to support e.q. high resolution graphics for text processing, mass storage disks, voice identification, ATM controller interfaces. As such, the price/performance remains virtually constant with increased features, and new features does not imply a total reinstallation of a new system.

The main objective for SDC will be to be able to cope with change as fast as possible, and to minimize cost over the lifetime of the project, not cost at installation time.

Therefore, at the level of choosing individual computers for functions in the savings banks, which must be present over a 10-year period, and must be upgradable through that entire period, my proposal for SDC is the following requirements:

- require the use of an international bus standard supported by many manufacturers, like VMEbus or P896.

- require the use of truly intelligent board modules, so changes in other parts of the system do not affect the documented function of that board.

- require the use of peripheral interface standards

If not, SDC and our suppliers will be preoccupied forever with reinventing the wheel again and again, every time a system becomes obsolete.

This page intentionally left blank

CHAPTER SEVEN

SYSTEM SOFTWARE

7.0 Introduction

With the implementation of TP2, software is becoming a major issue in the decentralized part of the savings banks system, and software has been one of the delaying factors of the TP2 system.

A common concern in system design is the famous myth of the hardware/software cost ratio (see Cragon, 1982; Boehm, 1983; Frank, 1983). This myth has been a long-lived fact, although various attempts have tried to counterfeit it. The myth says, that during time, software costs will account for a larger part of the total system expenditures, and that (in some versions) software should account for 80% of the total system cost today.

The price for the TP2 system as it comes from Olivetti is about 500 million danish kroner. In this figure, it is hard to separate hardware from software, but an estimate will be about 100 million kroner for software. The majority of software, however, are developed within SDC. About 100 persons are working on designing applications software. This corresponds to an annual figure of 25 million kroner, not including overhead for administration, machinery etc. Even though this figure is doubled, it is comparable with the annual hardware maintenance cost for the TP2 system!

Seen from this point of view, the hardware of TP2 plays a significant part of the total system costs. Also, the hardware is the major performance bottleneck at the moment and the factor most likely to be outdated very soon. This is why, the major concentration of this report is on hardware.

Software, on the other hand, have longer life than hardware, and this brings up requirements, that software should be transportable over different hardware modules over a certain span of years. Transportability means, that software modules can be transferred from execution on one processor system to execution on another processor system with a minimum of effort. True transportability, in the sense, that object code can be directly transferred without changes between different processors does seldom exist in practice. Normally, some form of installation effort or recompilation are needed.

Software transportability can be a problem in several ways. First, the differences in instruction sets are basic problems, that traditionally are overcomed by coding in a high level language, which hides the basic architecture of the processor from the programmer, and where transferral are managed through a recompilation. Secondly, differences in file systems or input/output methods can cause severe problems. These differences are typically seen in the interface towards the operating system, although they can occur at the language level as well. Thirdly, problems can arise from the usage of system dependent features within the application programs, like the usage of special terminal features or control codes, or a dependency on a specific architecture, like a master/slave relationship. A final problem area is the presentation of data, where some processors store words with the least significant byte first in memory and others store it correctly with the most significant byte first. Other examples in this field can be the alignment policy of the processor or the way, floating

point values are computed.

Besides software transportability, the way of acquiring software can be a problem. Traditionally, SDC has done all applications development by itself, or have worked out a set of specifications, which then has been implemented by a vendor specific for SDC. This approach can cause severe delays in the completion of a certain software facility for the savings banks or cost more money than necessary.

The software market has changed since the introduction of the microcomputer. This change has brought a concept of standard software, that is transportable between many different systems. This standard software consists both of application packages and of software tools, that ease the development of applications. Most of this software market is oriented towards business usage at all levels, from text processing, accounting, data base handling to managerial tools. Due to the high-volume market, the products will be cheap. As such, these software products will have aim at the savings banks also.

This trend has been virtually unknown for SDC and other companies as well, but it will have a major impact on the requirements for applications packages, that the savings banks will place on SDC.

The rest of this chapter will concentrate on system software issues in a future system architecture. The basic issues covered will be those of transportability of software, those of software change and those of using new standard software packages. These will be covered, as they will be most important to SDC in terms of economics. As such, the chapter will not describe a ready-to-use software implementation, but will more generally focus on the concepts, that links together the hardware and the software.

Dealing with distributed systems, one could expect a quite different approach, than I take, especially when one is not involved in practical system design. However, to get control of all aspects of distributed processing software and turn it into a workable system is not a one-man job in just two years, but a long-range team joint effort (see Lantz, 1980.2). Interesting enough, such projects seldom impose specific requirements on the type of machinery needed, but rather keeps control on a conceptual level, using off-the-shelf computers.

In contrast, this report advocates specific requirements on hardware (in order to implement systems from standard components, including computers) and instead recommend the use of off-the-shelf software.

7.1 TP2 software structure

I will not try to give a complete overview of the entire software in the TP2 system. This can hardly be done by anyone in SDC. But basically, the software can be divided into three areas:

- the operating system
- utilities/applications environments
- applications

The operating system used in the TP2 LC is called COSMOS, and is not very different from the vast majority of vendor-specific operating systems on the market in the sense, that it has what is needed and besides that, it is incompatible with everything else.

On the Terminal Computers, an operating system called TCOSMOS is used. The same remarks hold true for this.

Most of the applications uses the operating system through applications environments, i.e. layers of software, which implements SDC-specific utilities and requirements. As such, the applications can in general be considered to be independent of the operating system.

Applications in TP2 are written in the PLZ/SYS language. PLZ/SYS is originally developed by Zilog as a system programming language for the Z80® microprocessor and was chosen by Olivetti in 1979 as Olivetti's internal system language. The reasons for this are quite clear:

- PLZ/SYS is a modular, type-checking language, based on Pascal, MESA, C and Modula

- PLZ/SYS is processor independent

- PLZ/SYS is I/O independent (it contains no I/O statements!)

SDC choose PLZ/SYS as the programming language for TP2. This have provided for a good starting point into future migration projects. The reason is the following:

> PLZ has forced application development in SDC into the use of modular, goto-free programming. As PLZ does not have I/O as a part of the language, all I/O-routines are thus implemented as external modules in PLZ, and thus in this way provides a user-hidden separation, that maps the interface between the application programs and the operating system. Thus, ideally only these I/O modules must be altered, going to new operating systems.

Two problems remain: one is the fact, that current application programs are designed with the TP2 centralized architecture in mind, i.e. with some part of the application running on the LC (local computer) and some on the TC (terminal computer). This can impose some restructuring efforts in transferring programs to another type of system architecture. The major problem is however, that SDC lacks design of generality into applications. The TP2 software design, which are oriented towards interactive dialogs, was started without having any experience in interactive software methods or know-how on how video display terminals can be used. No prototyping was done. Screen pictures was designed on paper. As such, what is missing, is an awareness of general principles of communication, interactive dialogue design or physical characteristics of interactive terminals and how these characteristics are handled in the rest of the market.

The derived result has been, that there exist no general format of data within transactions (as stated in chapter 5), and this again means, that for every type of answer, a dedicated piece of application software exist just to show this answer on the terminal. First, this means a heavy use of disk space just to keep application programs for the more than 1000 types of answers, and secondly, it forbids (or at least make prohibitively expensive) the implementation of simpler types of systems, like customer terminals, portable terminals or cheap terminals for the smallest saving banks. Also, it means that data formats are locked to the type of display format once chosen. This is for example the case with several types of transactions and answers, where alphanumeric data supposed to be displayed in 3 fields of 40 characters each are transmitted as a 120-byte long string with blanks in unused places, even though only 20 bytes contain valid information. The reason is, that this is the only way, that people knows how to separate 3 fields of varying length. The effect of such an absence of

basic knowledge is one half a second larger response time for these transactions forever. As noted in chapter 5, steps have been taken for a general data presentation standard (see Bertelsen, 1982.3), although the project is running slowly. Due to the other problems in TP2, this proposal will however not be likely to have any effect before the system is stabilized and the major part of the systems installed.

7.2 Application needs

Applications within the TP2 system are actually quite simple in terms of systems requirements. Applications serves two needs: they enables the user to communicate with the central SDC databases and they carry out local task, like local computations, word processing, file handling and office automation.

The general philosophy behind the TP2 application development is, that applications are divided into two groups of systems: support systems and execution systems.

The support systems contain helping functions, authorization system, product guidance functions, subject archival functions and others. The executions systems contain the teller applications, the back-office applications and the office automation functions.

Basically, application programs need access to the local file system, to user I/O through screen, keyboard and printer and to the communication network. As such, they are no different from any other type of applications in the world. At one point, they may differ, and that is in the requirements of short response times.

To investigate the various needs, applications can be generalized in the following areas:

1) transaction oriented applications

These are the normal type of transaction handling within the system, which has been carried over from the old TP1 system concept. They fall into the category of teller applications. They cover data entry of a limited amount of information, typically a transaction number, an account number and a money amount. They can be described as "form filling" applications, where a predefined form on the screen is filled with the appropriate information. They may need the concept of "hidden" forms for user help, where a fast form update is occasionally needed. Processing is mainly (in TP2) performed by the TC, where LC time is primarily used for control, validation, update of counters and communication with the network. Primarily, it is a one-to-one type of communication, where the application interacts with the central database in SDC. These types of applications are also the most time critical, where a short response time is needed.

2) procedure oriented applications

These are applications normally falling into the back-office category, that uses a form of the transaction oriented transactions as a foundation for more complex types of work. In TP2, they are controlled by the SBPL (Saving Banks Procedural Language) concept (see OL, 1981.5), where several transaction units (TUs) can be combined to form a type of computerized office procedure, in the progress of which several individual transactions may be sent to SDC. The SBPL belongs to

the TP2 support functions.

This type of organization can be characterized by many forms and incidentally a more complex form filling than the pure transaction oriented procedures. Also, communication with SDC through the network may be more extensive, as several information exchanges between the application and SDC can take place without the user's notice. From a computational point of view, these applications do not burden the processor, as the prime task lies in control, but the many forms need frequent file accesses, and exchanges over the high speed link between the LC and the TC.

3) office automation applications

The procedural applications described in 2) and the office automation applications can be referred to as being primarily back-office applications. By this is meant, that they will not be used at the teller workstations, but rather behind the desk. As such, they may not require the same type of fast response time as needed for teller applications, where response time is a critical element in the total customer turnaround time.

Office automation applications can be described in two levels:

Level 1: is the type of applications requiring lots of data entry, like word processing. The complexity of forms are limited and restricted to the type of selection menus. However, these applications require extensive processing capabilities, needed to handle

large amount of non-structured data, text composing, search and sort facilities. They may need extensive use of input/output facilities, like file accesses for text swapping and screen I/O for fast screen updates.

Level 2: these applications uses lots of communication capabilities and include concepts like electronic mail. Processing is needed for file access and control, for store-and-forward types of communications and for data retrieval and search facilities. Form complexity is rather low.

Office automation functions will be those, where TP2 will show its greatest performance penalties in terms of a bad architecture. Sufficient advanced tools for office automation, which lifts the user above the level of a simple editor, will be too large to fit into the TC and too timeconsuming on the LC, which furthermore is not designed to perform that type of processing.

On the other hand, office automation as being offered on the independent microcomputer market will need for more advanced hardware features, like high resolution screens, graphics capabilities, more memory and a more natural command interface, which requires even more software and processor support, than can be handled by TP2 (see Williams, 1983).

4) "slide-rule" oriented applications

To this latter category belongs the computational-oriented type of processing, which main purpose is to handle numbers. Pure computational programs is the

ideal case for the TP2 LC, as these may fully utilize its internal stack structure. However, these types of applications are rare in the TP2 environment. In the years to come, we may see applications, which need this computational power, but these application will not be programmed, but generated through the use of spread-sheet calculator program, which cannot operate on TP2, and where the major part of the program is assigned to user presentation and screen handling.

As can be seen from above, the major issues in software are placed in three areas: performance, input/output and inadaptability of TP2 to new and larger applications.

The only way to solve the performance problem, rather than coding in assembler or acquire a more optimizing compiler, is to provide each user and each intrinsic function in the system with a separate processor. The way to do this are described in the other chapters of this report.

The input/output problem centers around, how application programs address the outer world. The outer world is composed of the user's world, i.e. the keyboard, screen, printer, the system's world, i.e. the file system and the network's world, the link towards other installations or SDC. The way to solve this is by applying the appropriate standards, so any device dependencies are removed from the applications.

The new applications problem is partly a performance issue (more speed, more memory) and partly a market issue (how to integrate standard software from outside vendors). How these issues can be managed is the key of the next paragraph.

7.3 Software in future systems

The current TP2 system software can be more and less controlled because there is only one master processor in the system together with several identical slave processors.

As was indicated in the previous chapters, future system will be based on concepts, where several processors participate at several levels in the system. Individual nodes in the system will have their own processing capabilities, and each node may be constructed by several circuit boards, each having its own processor.

This means, that software will penetrate a future system in a way radically different from today. This can be managed in two ways: by having the same type of processors throughout the system, and thus insuring some kind of software transportability or by having several types of processors, each running their own, individual operating systems, but communicating by a set of standardized conventions.

I believe the latter approach to be the one, that SDC will need to follow.

First of all, boards in future systems will contain their own processor and their own operating system. SDC cannot place any requirements on the type of operating system to be used, as these operating systems will be dependent on the type of function carried out by the board, and the type of processor. Most types of processors will possess a built-in operating system or a standard operating system dedicated to the board will be used.

Standard operating systems will be needed to use standard software products on the market. This will especially be true for the type of applications in the office automation area. In this area, SDC have and will have a constant back-log of requirements, which can only be satisfied by using off-the-shelf software from other vendors.

Today, nobody can deny the importance of standard operating systems. Apart from the IBM mainframe area, where a standard operating system has been a factor for the last ten years, not much has been practically done to provide such a thing. By a standard operating system is meant a operating system, which runs identically on different vendors different systems, possibly but not necessarily using the same type of processor architecture. The benefit of a standard operating system is from the user's point of view, that the interface to the system software is the same, no matter which hardware system he might be using, and that any applications software, that runs under the standardized operating system, can be easily transferred to another vendor's system using the same operating system.

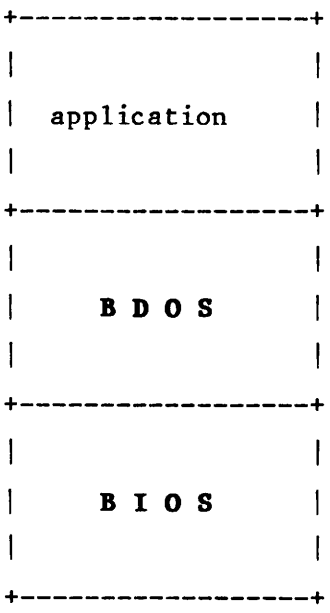
The concept of a true standard operating system for small computers came from the microcomputer market, where CP/M™ (Control Program for Microcomputers) in a couple of years took over the entire market for 8-bit processors. CP/M did not come from extensive research or from the educational institutions responsible for making progress in the science of data processing. It arose from a single man's need to interface a floppy disk drive to a microprocessor system.

CP/M <u>is</u> the de facto standard for operating systems for 8-bit computers today, with more than 1 million systems from 600 vendors installed. CP/M is primitive, with many deficiencies, but it works without errors, it is simple and it is concise enough to take advantage of the continuing migration and change

of microprocessor-based systems. The impact of CP/M does not come from the internal facilities of the operating system, but from the wealth of application software, that is available for CP/M, and which can run on almost any system using CP/M. Sorting out games and diverse hobbyrist programs, there are over 2000 professional software packages available for CP/M from independent vendors, many of which have more than potential interest for the savings banks.

What made CP/M so widely accepted, was its ease of implementation. CP/M is logically divided into a system-independent kernel called BDOS (Basic Disk Operating System) and a, possibly user-written, system dependent I/O-interface, called the BIOS (Basic Input/Output System).

All application programs interface the system through the BDOS with standardized calls. As CP/M for the 8-bit version is designed for the Intel 8080 processor, it means, that programs will run on any 8080-based or Z80-based system with a reasonable architecture. This is what can be called a true transportability, which has been important, since the majority of programs for the 8-bit market were originally developed in assembler.



The BIOS takes a couple of man-months to design and implement, which means that the investments in software doing a computer design are actually very small.

All other standard operating systems now on the market are done by the same fundamental strategy. These includes CP/M for 16-bit microprocessors, concurrent versions of CP/M allowing multiple tasks, MP/M™ for multiuser systems, MS-DOS™ for 16-bit processors and UNIX™.

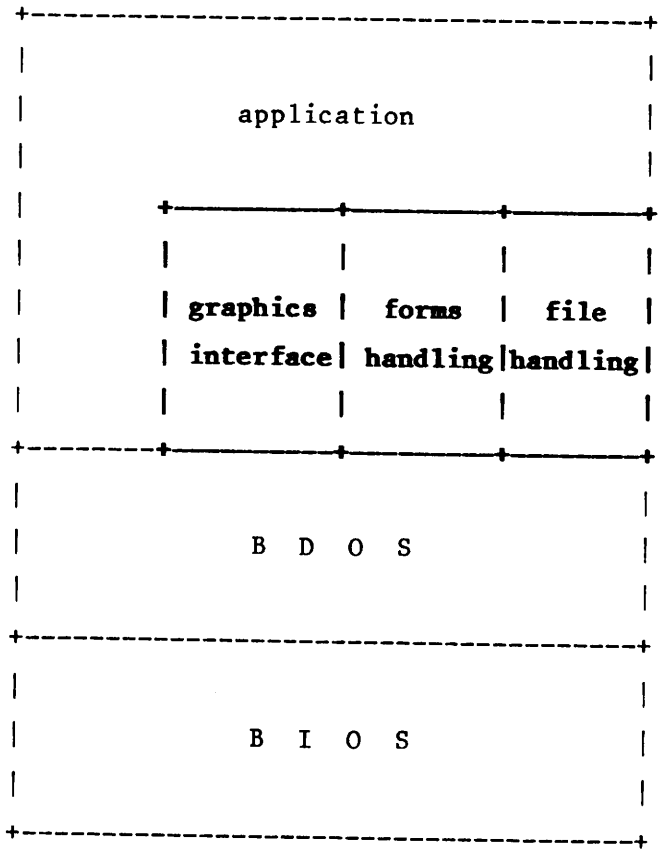
Transportability on the 8-bit CP/M market is assured by having the same processor architecture and the same way of handling memory in all systems for this operating system. This allows software to be transportable at the binary code level, without any need for recompilation.

Transportability, however, is handled differently when going to the 16-bit market. The trend for application development on this market is oriented towards the use of high level standardized languages, like C or Pascal/MT+, which relieves

the designer from paying attention to a specific type of processor, by using a set of standardized interface calls from the high level language level. Another way to solve transportability is to provide systems with multiple processors of different type, e.g. an 8-bit and a 16-bit processor within the same CP/M system. Interoperability are then assured by CP/M's standard file format, which can be read by any application program, whether it runs on an 8-bit or a 16-bit processor.

Systems from many vendors also mean many types of peripherals. This has introduced a concept of device independency within the application programs for the standardized operating systems, i.e. the application program can support different manufactures' screens or printers. This is normally achieved through the concept of an "installation" procedure, by which the user is taken through menu-oriented steps, where the different types of media can be selected by their name, or the product can be modified to suit specific needs for the user, like display colors, line width etc.

To facilitate this concept even further and to provide even more standardization, especially within the graphics area, the operating systems are now enhanced with high level drivers, which lie above the actual operating system, but which the application programs consider to be a part of the operating system. These drivers maintain a high level interface from the application programs towards graphics, screen forms handling, structured file accesses just to mention a few:



This trend will make it even easier to transport application software packages between different type of equipment and to adjust the software package to the various forms of input/output facilities, that the user has.

7.4 Standard operating systems.

It is very difficult to project, which way standard operating systems may take (see Freedman, 1983). However, some trends are clear. One of the major impacts will come from the UNIX market. All of the new generation of microprocessors will be used in UNIX-based systems, and the entire microcomputer software industry is moving towards UNIX. Still, the offerings on the personal computer market using CP/M and MS-DOS are more numerous than on the UNIX market, but these markets will adapt.

The latest versions of MS-DOS is almost UNIX-compatible, and CP/M is now being rewritten in C, with the intention to support all UNIX system calls under CP/M (see Black, 1983; DRN, 1983). This means, that it is only a matter of years before most standard software for these operating systems will be compatible with UNIX (see Halfant, 1983).

Other contenders for this evolution will be the Pick operating system (see Lewis, 1982; Gooding, 1983), the OASIS operating system (see Guyod, 1983) and possibly the p-System (see Overgaard, 1982; Manison, 1983). It is foreseeable, that the savings banks will acquire system based on these operating systems to be connected with the branch office systems, even though the branch system uses another standard.

The basic problem is not to solve the operating system issue, but to devise a set of rules by which applications on different machines can intercommunicate. This necessitates the need for a message exchange protocol, which defines the formats and rules, that are used, when applications must interact.

The following figure shows the prime areas of responsibility of software development in a future system concept, which are implemented through different computers running standard operating systems. Many applications will still be designed by SDC, but in the case where standard software tools can be used, these will be implemented on the system. Application services or environments, defined by SDC in the current system, will be more vendor based in future systems, due to the use of standards. However, these standards will make the application software independent of the lower layers, which means that the concept of multiple operating systems will not be a hindrance. Operating systems will come from different sources. Some system may use UNIX, some the current COSMOS, some CP/M and some MS-DOS. Drivers, needed for input/output, may depend on the type of hardware being used, but generally, drivers will be

implemented on a separate processor on a specific circuit board with possibly its own operating system.

	current	future
applications	SDC	SDC/vendors
application service	SDC	vendors/SDC
operating system	Olivetti	multiple vendors
drivers	Olivetti	multiple vendors/SDC

Note, that this separation is actually independent of the physical hardware architecture. A specific processor may support one user, multiple users or only one layer within the software.

The result will be the following: applications software will run under a standardized operating system, which will enable SDC to take advantage of the standard software products market. In the case, where this operating system does not satisfy all the requirements for applications development within SDC, or where software interface standards are needed towards external devices, SDC may acquire or develop a set of suitable extensions, which are unique to the particular operating system, but which provides a common interface towards the application programs.

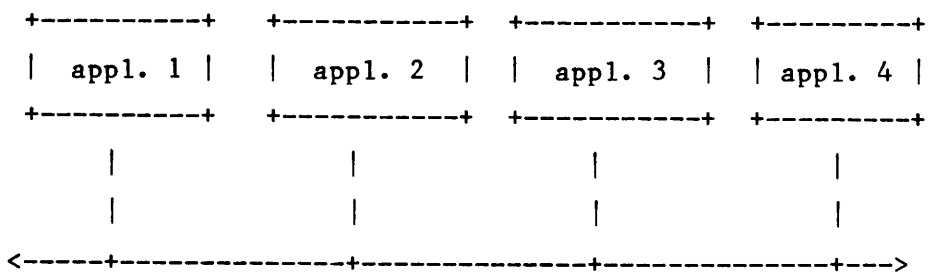
The operating system may run on any system component in the entire system. It can run on a self-contained system, directly connected to the public network, it can run in a self-contained component connected to a local network or it can reside on a

single board, which is part of a larger system. The operating system may support several users or just one single user.

This type of freedom can only be achieved, if an application program can be guaranteed, that no matter where it is in the system, it can always access other applications, like the central services of SDC, in the same way. That is, it shall be possible to open and maintain a software communication path within the system.

Note that this path physically may be implemented in various ways. It may use a type of network, public or local, it may use a bus structure or a common memory between multiple processors or it may even take place between applications running on the same processor. As such, it may make use of various protocols defined for the path, but the application program should not be aware of this. The application program will need a logical path through which it can communicate with all other applications.

The only way to do this is to introduce the concept of a message bus. This bus is not a physical bus, but a software construct, which allows applications to intercommunicate over the bus. Software communication follows the same rules as hardware communication, which means, that a set of protocols has to be implemented for this type of bus.



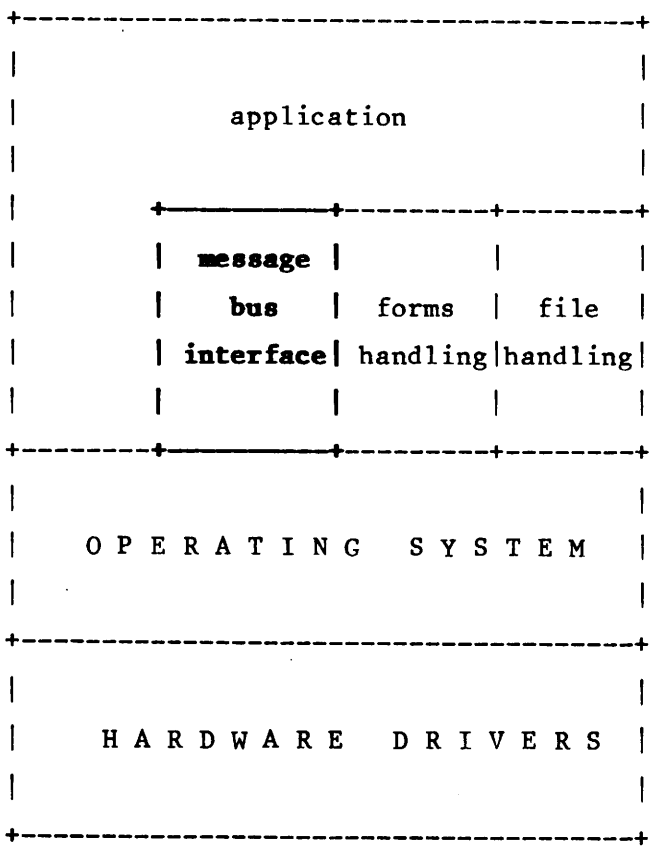
message bus concept

The message bus protocol consist of (1) definitions of the

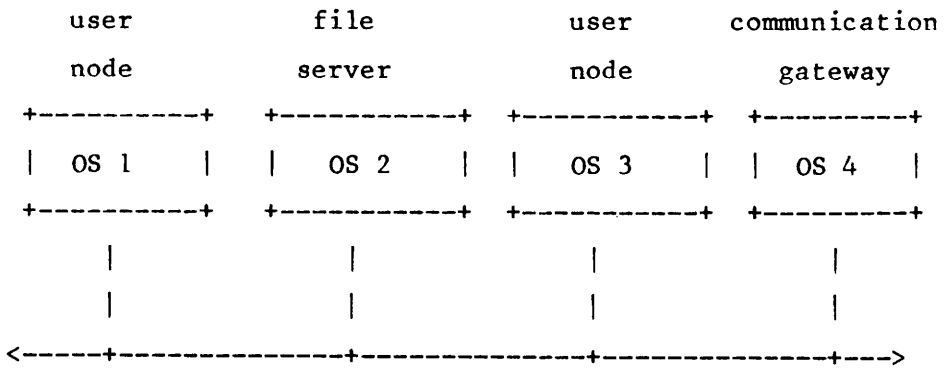
format of the information being exchanged and (2) rules for this information exchange.

Note, that the type of information being exchanged is of no importance in this definition. Information can actually be anything, from SDC's traditional transactions, file requests, operating system calls, electronic mail data or digitized voice. The main objective is to provide a common method to allow two applications to intercommunicate.

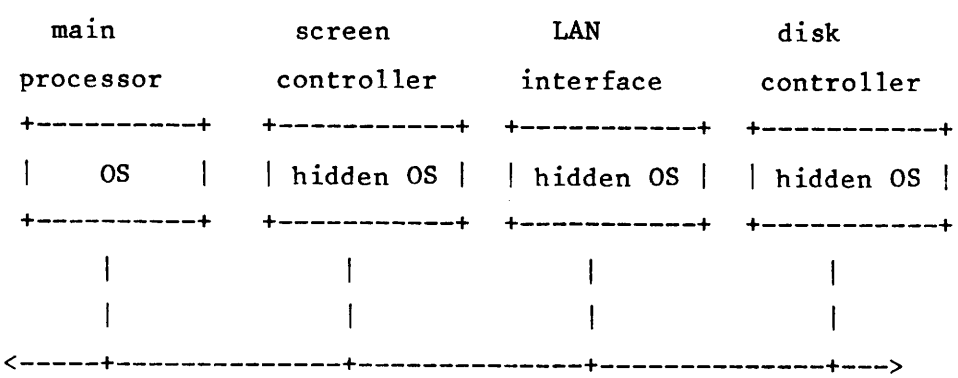
Seen in correspondence with the operating system efforts, it may be probable, that such a feature will be implemented as a standard extension to existing operating system. Some systems already incorporate the message-based communication concepts, like the Ridge OS (see Basart, 1982) or Intels iRMX™ (see Heider, 1982). Still SDC may need to provide to the applications the proper interface, which does not have to be an integral part of the operating system itself:



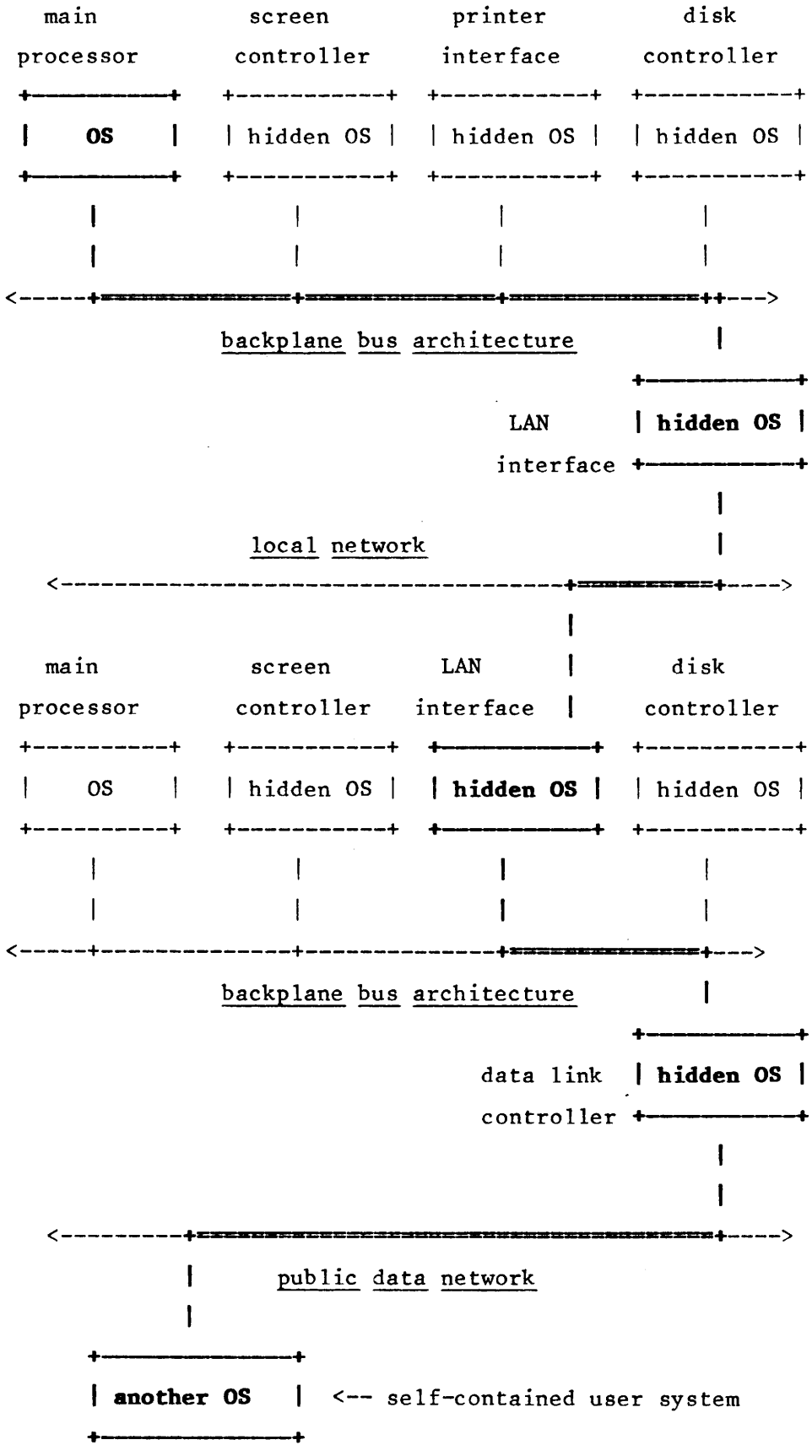
One should also note, that the use of a concise message system is needed to take full advantage of the type of hardware architecture proposed in the other chapters. A branch office system designed around several individual nodes tied together through a local network will contain operating systems in each node, and these operating systems may very well be different, according to the node's needs. A node controlling a database or a node controlling a public network link may need a substantial different type of operating system support than a user's node:



Again, as each node can be designed from individual circuit boards each containing its own processor, each board may be equipped with its own dedicated operating system with different facilities. Note, that normally one of these operating systems will be the one assigned to the application interface, and the one that the application and the application programmer see. The others can be denoted as "hidden" or embedded operating systems, as they carry out the specific tasks of that board, e.g. controlling a local network line or managing a screen display:



The message bus concept, which enables two applications to talk together, embraces this entire organization of individual operating systems, and in a given case, the message path may take any way through the system:


main processor
screen controller
printer interface
disk controller
OS
hidden OS
hidden OS
hidden OS
backplane bus architecture
LAN interface
hidden OS
local network
main processor
screen controller
LAN interface
disk controller
OS
hidden OS
hidden OS
hidden OS
backplane bus architecture
data link controller
hidden OS
public data network
another OS
<-- self-contained user system

7.6 Conclusions

As can be seen, a future software strategy must penetrate all layers of the various hardware architectures and indeed be more and less independent of the hardware architectures. If this is done by using standard operating systems and a standard message exchange protocol, conceptualized here by the message bus analogy with hardware buses, the way is open for change.

Again, this concept does not go into details. For the various types of operating systems used throughout the system to implement e.g. the local network protocols, the special issues for that type of service must be addresses and taken into account. Standard software drivers will provide for this in most of the cases.

The responsibility of SDC will be to define the way how to open for an easy usage of standard, user-oriented application products. By doing this, many immediate needs of the savings banks will be served, and the programming resources in SDC can be utilized to develop the applications, that are unique for the savings banks, and for the rest take benefit of what is happening on the market.

To achieve such a state, further work is needed in SDC or at the vendors. This work will concern the following areas:

- SDC must acquire a detailed know-how on standard operating systems and their interfaces. This know-how should be used to set the guidelines for future applications development and be used to implement prototype systems of applications in the border of TP2.

- SDC must acquire know-how on the standards area concerning I/O devices and impose a requirement for using such standards in applications programming.

- SDC should follow the industry trend towards C as the software transport medium. This does not imply, that current TP2 software cannot be transported, but rather that SDC should possess the PLZ compiler and code generator in a version written in C. This will allow SDC to migrate to new processor architectures in a more steady fashion.

- SDC should define a message protocol, which both takes into account international standardization activities and the concepts of data communication in local systems.

These suggestions require, that resources are allocated within SDC. Some of the suggestions may however be fulfilled by standard offerings on the market. In any case, it will require a substantial effort, outside the scope of this project, to detail the aspects of these concepts, but it may be needed to take the full benefit of the future systems without dropping TP2 completely.

This page intentionally left blank

CHAPTER EIGHT

THE TARGET SYSTEM ARCHITECTURE

8.0 Introduction

This chapter contains the new system architecture proposal for the decentralized part of the savings banks' data processing systems.

In chapter II, the old and current system architectures of TP1 and TP2 systems were described together with some of the problems inherent in these architectures. Also the current level of technology was stated, together with the basic assumptions of the project.

The following chapters showed the trends of distributed data processing systems based on microprocessors in related areas as hardware technology, communication, architectural elements and software. Two key points are obvious from these chapters: (1) the technology pace and the diversification of possibilities, which may cause todays systems to be outdated and useless before they are actually paid off, (2) how international or de facto standards can be applied at appropriate points by the industry with a catalytic effect on usability.

8.1 Design for change

The only constant item in the Danish banking environment is change! Technology, demands and application needs are changing fast. Any attempt to enforce a static use of technology onto the savings banks are doomed to fail. This is why **TP3 must be designed for change!**

TP3 will be different from TP2 in structure and in technology. The structure defines the long-term logical implementation of the system. The technology defines the short-term physical implementation at a given point in time. As such, the same structure may embody several generations of technologies, allowing the structure to take benefit of new technologies in terms of processing power, maintenance, price.

I will do this by setting up the skeleton for the next generation of decentralized data processing systems in the savings banks as a system architecture, which defines a both strict and loose framework under which, new systems can be designed in a orderly, but still fast, way during the next decade.

The **system architecture** is a description of the modules, that constitute the entire system, **and** the interaction between these modules. However, a good system architecture must show a synergistic phenomenon, making the benefits of the architecture greater than the sum of benefits for the individual modules themselves.

SDC and the savings banks will need such a system architecture in four cases:

1. to define how the systems carrying out customer-related operations in the "teller" environments are to be designed and should interact. These systems are more or less present in both TP1 and TP2.

2. to define how the systems carrying out customer-related operations in the "backoffice" environment are to be designed and should interact. These systems have been introduced in TP2.

3. to define how the pure internal systems for the savings banks in the "backoffice" environment are to be designed and should interact.

4. to define how possibly stand-alone pure customer systems, like ATM's, self-service terminals or business terminals, which has not been an integral part of the TP2 system, are to be designed and how they should interact with other parts of the entire system.

SDC and the savings banks should benefit from a new system architecture not only on the direct cost of the system, but also on indirect savings. These benefits will occur from a number of background requirements:

1. Easy installation of new systems, ultimately as customer-setup.

2. Easy integration of new systems into existing configurations, without the need for technical assistance from SDC.

3. Minimal impact on SDC in planning and controlling an installation.

4. Minimal impact on SDC in the introduction of new features.

5. Minimal maintenance effort on existing systems or functionalities, when changing physical modules.

6. Prototyping can be introduced for new projects without interference with existing systems.

These requirements actually have their counterpart in the features offered by the standard proposals within the local networks and the computer backplane buses area. As such, the fulfillment of these requirements are a direct implication of following such standards.

The only problem is that of combining these standards in a suitable fashion into a system architecture, so they can work together and make sense. If they do not, it may be impractical to use them, and the users, who is the target of the system architecture, may never see the fundamental ideas behind the structure.

This is the approach, that lies behind the system architecture presented on the following pages. Concepts are presented rather than details, because if the fundamental concepts are not completely understood in SDC, this type of architecture can never be implemented. Details will be reliant on the actual

standards, that are being chosen, and their inherent capabilities.

The system architecture proposed in this chapter will not give all these features at once, but it will form the background for further work in this area. To implement it will require, that resources are moved to this area of architectural design. If the work done in this area however can cause some problems to disappear from the old system, it will be worthwhile.

As such, the proposal are not complete nor final nor unchangaeble. It is a step based on the new technological options, which are penetrating the market. It is a step towards a system architecture.

8.2 A System Architecture

The proposal will be called **A System Architecture (ASA).** The reason for this is to release the traditional views in SDC of considering the decentralized projects as being terminal oriented (like TP1, TP2, etc.). Indeed, the savings banks' employees view SDC through terminals, but SDC problems comes from the missing awareness of the architectural base of such projects.

Also, ASA will not reflect the entire system configuration at a given point in time, linked to a given set of static vendor modules, but as a framework describing the main roads of alternate, constant migrations, that will occur in the years to come. As such, ASA may not be the only architecture usable for SDC, but it will at least be a stated level of the directions, where SDC will go, as a help to possible vendors.

As such, ASA will be in accordance with, if not a little step ahead of, the general product evolution in the microcomputer industry. ASA should eventually become realizable by standard product offerings from the industry. If a system architecture is not able to do that, it may eventually end in the same locked state as the TP2 system. Hence, rather than circuit diagrams, ASA is based on concepts, not currently available in SDC planning, but which are necessary to define, where SDC should go in product acquisitions in the years to come.

It could be possible to define a system architecture, which would totally define the entire decentralized banking system of the savings banks in so to speak one equation. Such an architecture, however, would impose restrictions, which might not be consistent with the current system organization and it would probably be a total departure from the current TP2 system with no intermediate steps.

Also, a system architecture, which would require a complete homogeneous system, seen from the architectures point of view, where all elements were needed to possess all functionalities of the architecture, might solve substantial problems in TP2, but the questions are, whether any manufacturer would support it and whether this architecture would be prepared to incorporate changes and new concepts.

What I propose is a concept of leveling. This leveling will allow and indeed enforce an inhomogeneous system, where certain elements do not adhere to the proposed system architecture for the internal design of such elements. However, the introduction of levels will allow for a stepwise migration towards the final stage, where all elements in all levels meet the desired architectural requirements. Furthermore, it will allow rethinking as time and technology goes on, necessitating levels to be redesigned and restructured.

It must be emphasized, that once this type of architecture has been chosen by SDC, continuing work must be done, both to impact the general idea in SDC, and to follow (and preferably participate) in the implementation work and the standardization work being done on the different levels at different manufactures and in the entire marketplace.

Therefore, ASA is a leveled architecture, where the same basic structure and concepts appear at all levels. The levels are:

1. Nation-wide level
2. Branch level
3. User Node level
4. System Software level

The ASA levels defines the decentralized part of the entire savings bank system. The centralized part in SDC, Ballerup, are considered to remain architecturally unchanged, although it can be implemented with exactly the same concepts as in ASA!

ASA leveled system architecture

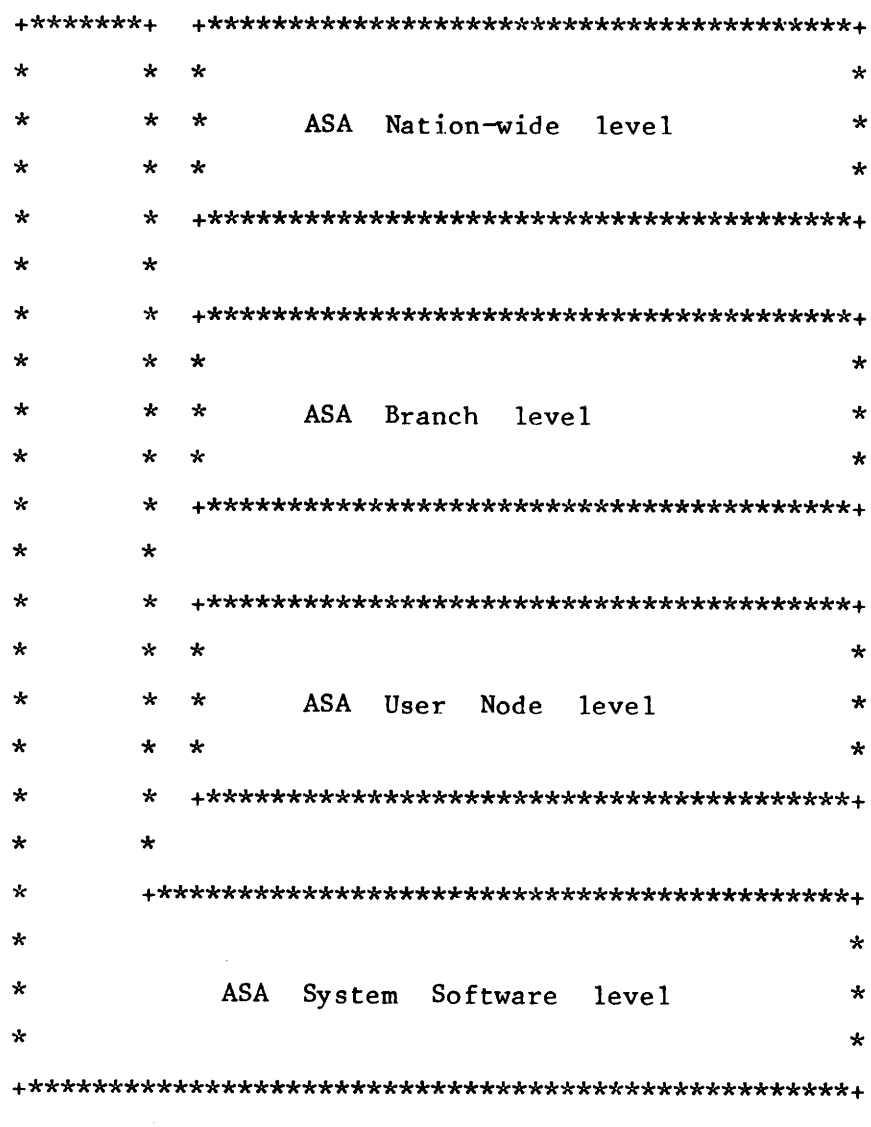


Figure 8.1

These levels divides the decentralized systems into logical partitions, that may individually be analyzed and acted upon:

- If one wishes to attach new systems to SDC, like self-service terminals, data center gateways, business computers or info center services, the ASA Nation-wide level will set up the guidelines.

- If the need arises to expand a branch office installation with managerial workstations, local data base systems for document retrieval or an branch inventory control system, the ASA Branch Level will show how.

- If the performance in time becomes too low to support new software options or new advanced peripheral units comes on the market, the ASA User Node level allow the easy modification or enhancements of existing system hardware.

Finally, the ASA System Software level is the general method used in SDC to ensure connectivity between software modules and through this way ensure portability. One should note, that the software level in itself is self-contained, but also covers the other three levels. This is because the concepts within the software level must be present, although not identically implemented, in all components attachable to the other levels. As such, the software level cannot be uniquely assigned to one of the other levels or to certain components within these levels. Also, the software level does not encompass all software in a given system and especially not the application software, but only the general methodology to make application modules interact.

8.3 ASA Nation-wide level

The nation-wide level was actually introduced as a concept with the TP2 system, even though SDC still consider it synonymous with the Olivetti system. It is the network level, which interconnect branches and other stand-alone components with each other.

These components may be:

- branch office systems
- back office systems
- SDC central data center
- other data centers
- business systems
- stand-alone customer equipment (like ATMs)
- other networks

ASA considers these components to be independently connective to a network architecture called the nation-wide level architecture.

ASA nation-wide level can be considered as a bus structure, where the public data network constitutes the bus, allowing distributed access and control of nodes on the bus (see figure 8.2).

ASA nation-wide level is not concerned with the internal architecture of systems, that connect to the level. As such, it can be considered to stop at the interface connector from the system to the DCE on the public network. But in fact, it goes a bit further, as it also defines the fundamental protocols used on the system to access the services on the network.

In this sense, ASA nation-wide level includes:

- use of the Danish Public Data Network with X.21 as the connection protocol and balanced HDLC as the link protocol.

- use of the TP2 path control layer

- use of the TP2 End-to-End layer in an expanded form, which are not just oriented towards the TP2 transaction communication and file transfer.

Within a given system, ASA nationwide level thus includes both the needed hardware for accessing the network and the piece of software implementing the network protocols.

Taking a look on this level architecture, one finds that

- access is distributed and homogeneous
- access is random demand-driven
- multi-vendor access is possible

It must be noted, that the current tp-monitor concept of TP2 is nearly sufficient for the ASA nationwide level and will as a first phase serve as the basis for this. However, some amendments must be done to define a secure access control to the network (so accesses are retried in case of DCE/DTE collisions) and to define an internal data presentation layer within the tp-monitor application layer.

ASA nation-wide level structure

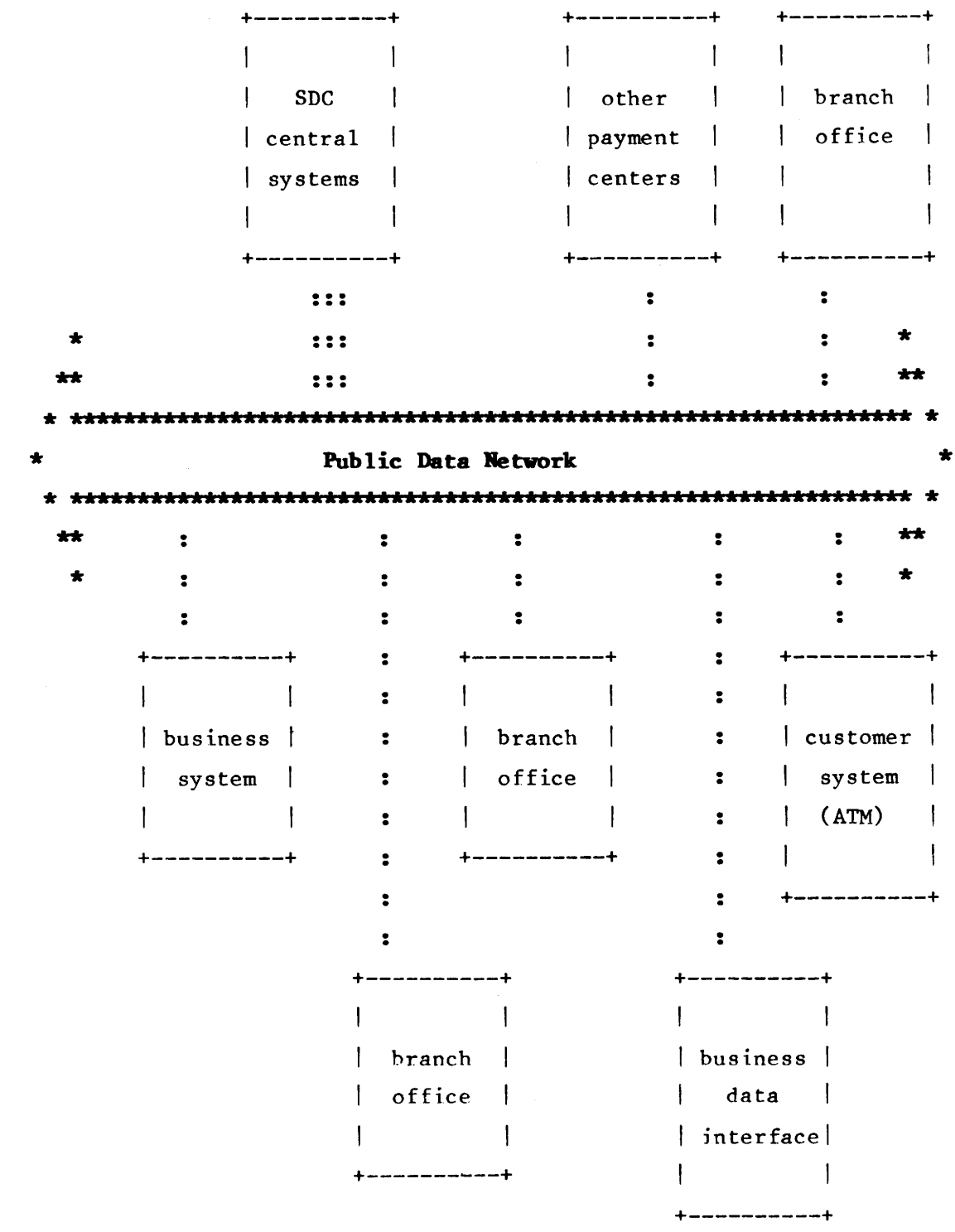


Figure 8.2

Due to security reasons, the savings banks' use of the network is restricted to a closed user group within the total public network, but the closed group does facilitate the distributed aspects of the architecture and does not inhibit the basic goals of the ASA nation-wide level.

ASA nation-wide level architecture will enforce SDC to drop the current view, that Olivetti systems are the only official things attachable to the network, and that external devices only can be supported for the network through Olivetti systems. One example is the ATM implementation, where the entire ATM applications and communication interfaces today has to be rewritten and reimplemented in the case of a migration project to new hardware in the branch offices.

In the ASA nation-wide architecture, such systems will be connected directly to the network, and as such will not be affected by any changes in other system parts of the network.

8.4 ASA Branch Level

The ASA branch level architecture is concerned with the actual configuration of a branch office system from a systems component point of view. System components in this sense includes:

- processing power
- storage devices
- peripheral devices
- communication capabilities

The current TP2 architecture locks SDC into a position, where it is very difficult to take benefits from the progress of technology. The requirements for a new architecture at this point include:

- the ability to support both small and large branches with a constant cost/performance ratio.

- the ability to expand a given configuration with new facilities or workstations without the need for central planning and system reconfiguration

- the ability to remove performance bottlenecks by adjusting the processing power provided to different users

- the ability to integrate new functionalities into a configuration without the need for redesign or reconfiguring existing functionalities

This is done by introducing the concept of intelligent, stand-alone system components, which may vary from installation to installation and in time.

ASA introduces the local network as the means to interconnect these system components in an orderly fashion. The reason for this is the concept of having processing power at the user level:

> Each system component in the ASA architecture is supposed to have built-in processing power enough to support the functionalities of that system component. This implies, that there is no central part of the system architecture, which must be shared to make the branch office system work.

Hence, the local computer, known in TP2, has disappeared in the ASA branch office level.

Note, that ASA branch office level defines the internal connection scheme of modules within a branch office. One of these modules may be a connection to ASA nationwide level, but in fact the branch office level is independent of any nationwide level, that might be chosen.

Also, ASA branch office level may not be employed in all branches and may in some branches be implemented stepwise over a certain span of years in coexistence with current TP2 system.

ASA branch office level

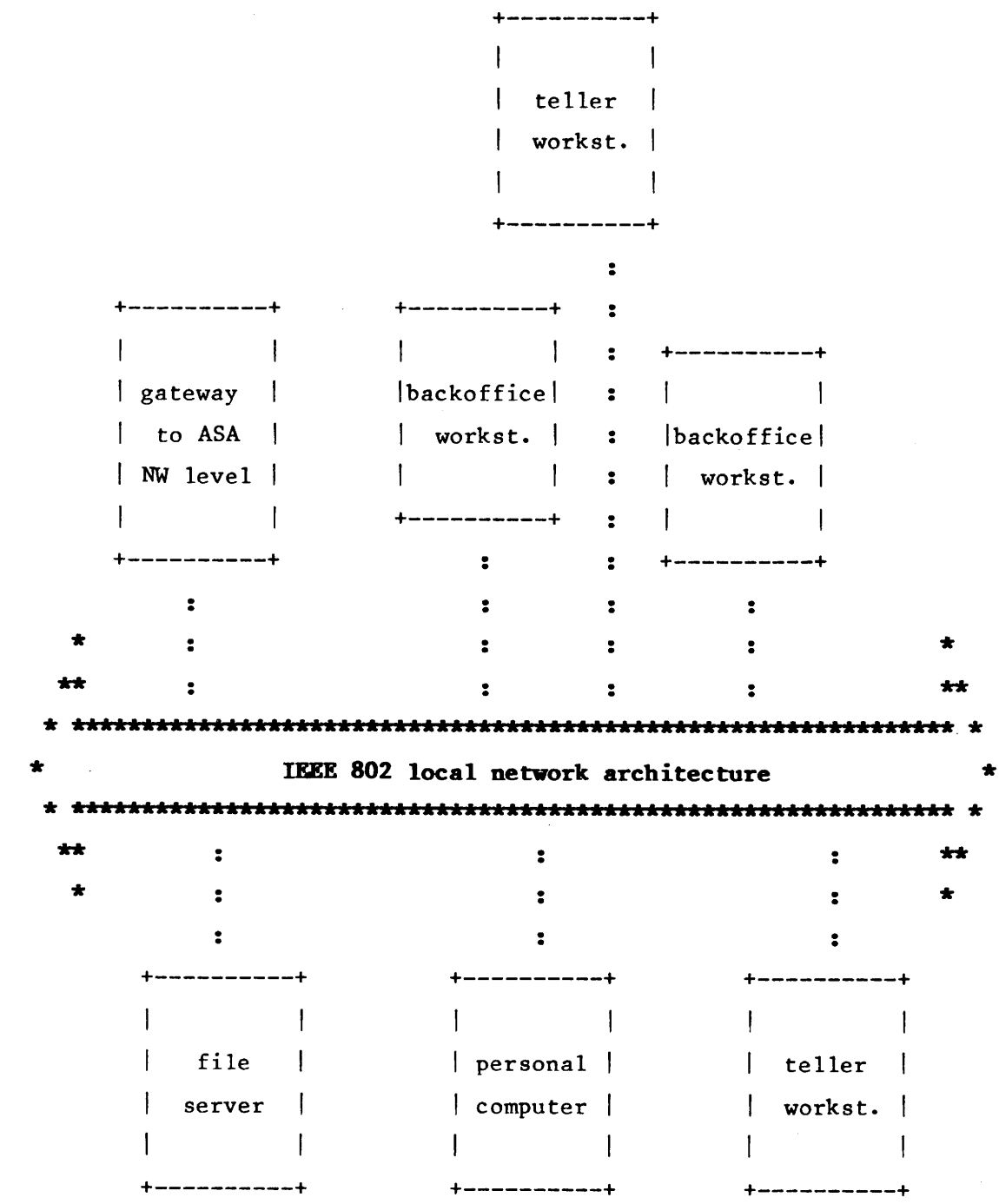


figure 8.3

Although the concept of a standard local network is fundamental for the ASA branch level architecture, several standards may contend for the role, and indeed live up to the requirements, that SDC will place on such a network.

As a long term strategy, ASA branch level architecture proposes to use:

- IEEE 802 layered LAN network standard, with
  IEEE 802.2 Logical Link Control
  IEEE 802.3 CSMA/CD bus access method

- A SDC defined version of higher layer protocol for teller communications

- A standard higher level protocol for applications interfaces

The IEEE 802 LAN model is chosen for a variety of reasons: First, the Logical Link Control part of the standard is the same for all sublayers of the 802 standard. This means that SDC may choose to use the token access methods in some installations, where this can be beneficial, or if it turns out to be the ultimate choice of IBM and the market.

Secondly, even though Ethernet is the ultimate LAN choice on the market at the moment, and indeed can be installed as a first generation implementation of ASA branch level, IEEE 802 is estimated to have a larger support at the end of this decade than Ethernet, due to the independence between the LLC and the physical access method. However, as the latest revision of the IEEE 802.3 CSMA/CD specification now embraces the key points of Ethernet, Ethernet devices will eventually be attachable to IEEE 802.

Thirdly, the current TP2 TP-monitor specifications at the Link Control Layer resembles the 802.2 logical link control so much, that SDC's TP-monitor concept can be augmented quite naturally, so it fits in on top of 802.2. This further features the ability to separate functions into self-contained system modules without giving up the benefits of the current TP-monitor.

As such, ASA branch level architecture will feature:

- industry compatible interfaces allowing multi-vendor system components
- distributed access to system resources
- minimal impact on SDC when introducing new system components
- support for small branch offices and large back offices

Secondly, ASA allows multi-vendor equipment in the branch office as long as this equipment obeys the rules for node intercommunications. This will be the fact, when having the standard local network architecture established for as many layers as possible.

At this point, ASA makes no assumptions regarding the actual node architecture, and therefore gives the system designer a high degree of freedom to design the node. This is because nodes may be selected from many criteria:

- SDC-acquired special branch systems (e.g. from Olivetti)

- off-the-shelf network services (like file-servers, communication-servers, printing servers)
- personal computers
- word processing equipment

However, when SDC acquires special equipment, like teller systems, back-office systems etc., these must be designed according to the **ASA node level architecture.**

8.5 ASA node level architecture

The **ASA node level architecture** is concerned with the internal design of nodes, i.e. the way a specific computer in the system is to be designed. The basic modules in such a computer comprise

- processing modules
- communication controllers
- memory modules
- disk controllers
- tape controllers
- peripheral I/O controllers

These internal modules must be interconnected in a way, that gives the highest flexibility and the fewest restrictions in a certain time frame.

As described in chapter 6 on specific backplane architectures, this is one of the areas, where standardizing has provided new concepts into computer design in the microcomputer area. Even though many of the lost-cost system modules in the ASA branch level architecture will implement the internal modules above on single-board constructions, the general and high-level market area will tend to put these internal modules on separate boards.

The **backplane bus** is the method used to define intermodule or interboard communication within a single node. Selecting a vendor specific backplane bus will cause SDC to be locked out from the ongoing developments and technology utilization in the standard boards market. This may be justifiable in terms of costs in some situations, iff the vendor specific bus is sufficiently advanced.

However, utilizing a standard backplane bus will provide SDC the ability to individually design a given node without technical considerations on connectivity. It will allow SDC to upgrade or add processing modules while still using old memory and controller boards. It will allow SDC to implement 8-, 16- or 32-bit processors as needed in the same physical cabinet.

This could of course be achieved by redesigning existing systems from scratch, either by SDC or by the vendors, and this can be a solution in certain cases (see chapter 9). However, in the long run, SDC must release itself from such a dependency on specific hardware re-design requirements, and direct its efforts against new design, using off-the-shelf products as much as possible.

This is why, ASA user node level architecture is basically a backplane specification:

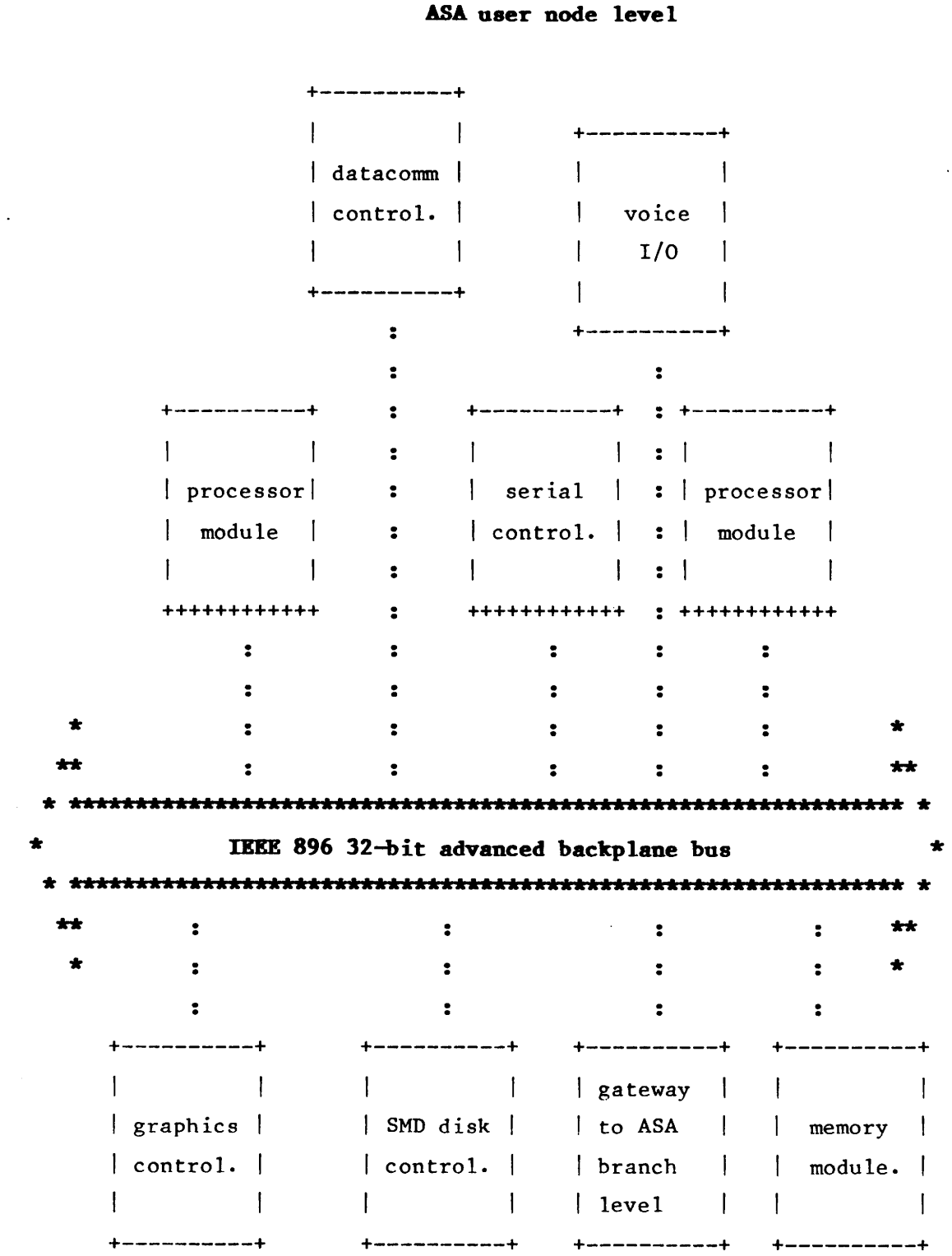

ASA user node level
datacomm control.
voice I/O
processor module
serial control.
processor module
IEEE 896 32-bit advanced backplane bus
graphics control.
SMD disk control.
gateway to ASA branch level
memory module.

figure 8.4

The concept of a standard backplane bus is fundamental of ASA user node level. However, several standards may apply to the requirements for such a backplane structure.

Hence, as a first generation step, SDC should require, that systems being acquired do use an industry standard backplane bus, i.e. MULTIBUS or VMEbus.

As the next generation step, ASA propose the use of the IEEE 896 future bus standard as the fundamental specification of the node level architecture. Choosing this has several reasons:

- IEEE-896 is a 32-bit processor- and vendor-independent bus architecture designed to last and support microprocessor systems for at least 10 years from now

- IEEE-896 is a distributed bus architecture

- IEEE-896 allows multiple 8-, 16- and 32-bit processing modules within the same system

- IEEE-896 is the only bus architecture available, which is being designed as a layered architecture in accordance with the ISO Open Systems Interconnect model

- IEEE-896 is the only bus architecture currently available, which may take benefit of the possibilities of future advanced VLSI and VHLSI circuits to reduce the physical board size. In fact, it allows a 32-bit computer to be designed on a single Eurocard.

- IEEE-896 will not impose any significant limits on throughput, memory addressability or internal node architecture in node design. The only limit, which will be a dramatic benefit for SDC, will be that of having to distribute intrinsic functions to separate intelligent boards.

Using IEEE-896 will remove the static nature of SDC's current system offerings to the savings banks. A user node can be incrementally designed and expanded to serve almost any needs. Workstation nodes may be build by connecting a processor module with the needed I/O and LAN communication modules and memory modules as needed. The 4 Gigabyte physical addressing range should be sufficient for SDC needs. Communication gateways would be designed with the same boards, and so would local database processors. Advanced graphics would be implemented by adding a graphics controller board to an existing workstation. Voice applications likewise.

These boards may originate from different vendors, but be common in the sense, that any board from any vendor designed at any time in the standard's lifetime will interconnect to any other board.

One should note again, that the ASA node level architecture is independent of the nationwide and branch level architectures. This means, that the node level architecture can adopt to other requirements for overlaying architectures, as well as nodes do not necessary have to be designed according to this architecture.

8.6 ASA software level

The ASA model is useless without software, and software is probably the most critical part of the branch systems. This is due to the huge investments in software and the fact, that the lifespan of software may exceed the lifespan of hardware.

SDC must be prepared to change views on software implementation in the ASA model. The basic reason is that ASA by its nature is a polymorphic system, as it will be impossible to select a single operating system and software tools for all system components.

However, this may not be a hindering factor for system development carried out in SDC, as this development will take place with standardized languages and standardized application interfaces. ASA is a open systems structure, where application execution takes place on individual processors for each user.

Although not a part of the ASA software level, SDC will need to impose some requirements on future software:

- standard operating systems must be used where applicable. Several operating systems may be used, as long as they provide the same basic interface to applications.

- standard application services must be used. This is particular true for device drivers, i.e. the way screens, printers etc. are controlled.

- standard applications must be used where applicable. These must run under a standard operating system and not be modified to fit into existing systems.

- a standard language should be used, which is supported by all processors and operating systems, and this language should be used as source language for other language compilers or SDC-made utilities used throughout the entire banking system.

Applications will however communicate with each other, and the intercommunication is the key factor for ASA software level. The trend in open systems design is indeed to make applications intercommunication independent of the physical processor and the underlying operating system.

ASA software level introduces a concept of message passing between applications. This is in accordance with the ISO OSI model, but rather than using this, intercommunication can be described in the same terms as the hardware-oriented ASA levels.

Message passing in ASA takes place through a **message bus**, which is a logical bus, covering the entire ASA structure. Applications or programs in ASA, which intercommunicates, can reside in the same processor, in different processors in the same node or in different processors in the same network, local or nation-wide. Indeed, IEEE-896, IEEE-802 and current TP-monitor will support this type of application communication.

This gives the following picture of the ASA software level:

ASA software level structure

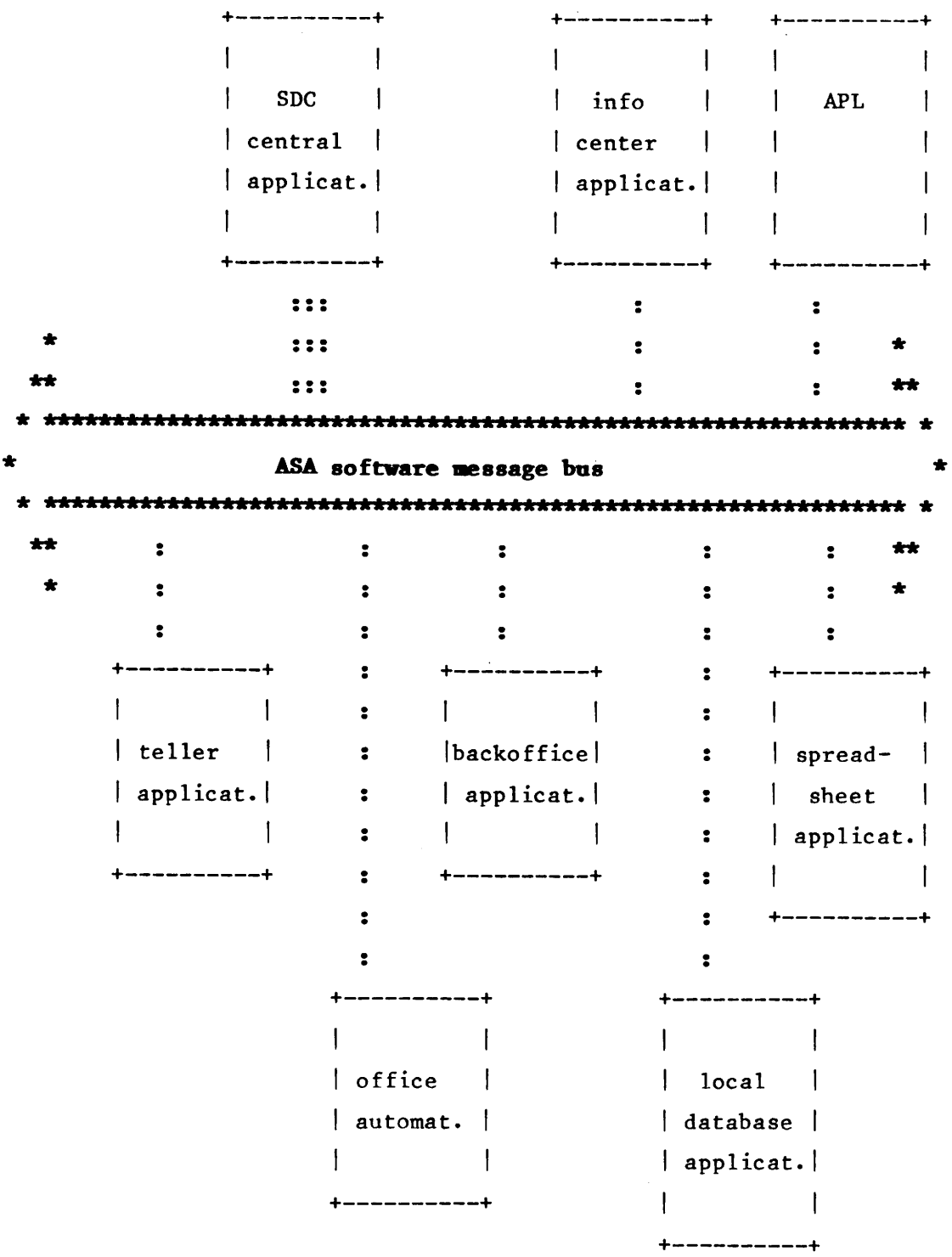


Figure 8.5

ASA software level will require, that the already started project on transaction standardization should take this type of message passing into consideration. It will also require, in some cases, a restructuring of existing applications, which are related to the current TP2 architecture. In all cases, it will be an amendment to the current tp-monitor concept in TP2, and it will require work to be done in SDC, before it can be realized.

However, it will mean a simplification of the application structure in the decentralized systems and ease the implementation of new and more advanced applications.

It will require a much more fundamental documentation of the communication strategy between applications, than is available today in SDC. In the long run, it may even change the structure of the central SDC applications, as handling of the End-to-End Layer moves from the front-end system to the central applications processors, so real End-to-End and hence a more session-oriented application to application communication can take place.

It is important to realize, that the ASA software level, once implemented, will penetrate throughout almost the entire system, as the type of message passing needed by ASA may be a function, that is supported by many different modules in the system. These modules may be entire systems, nodes within a system, boards within a node.

Also, since ASA software level is basically a communications oriented way of interacting, it can be considered as being an upper layer in the SDC layered model, between the application layer and the end-to-end layer. As such, it will interface the end-to-end layer and indeed may consider the end-to-end layer as a communicable application. Thus, ASA software layer is not only a horizontal hierarchical layer as the others, but also a

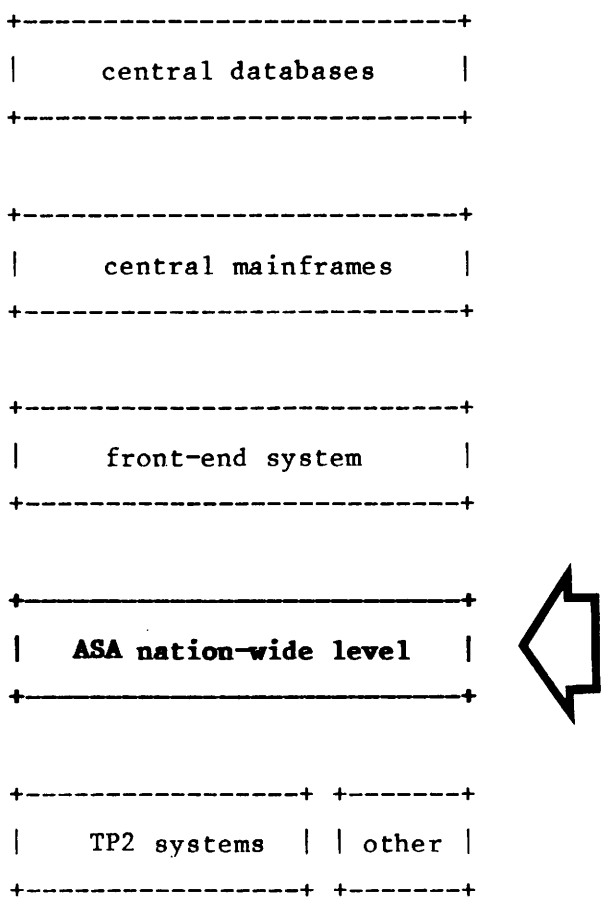
vertical layer with impact on the software in the others.

8.7 ASA integration into current system

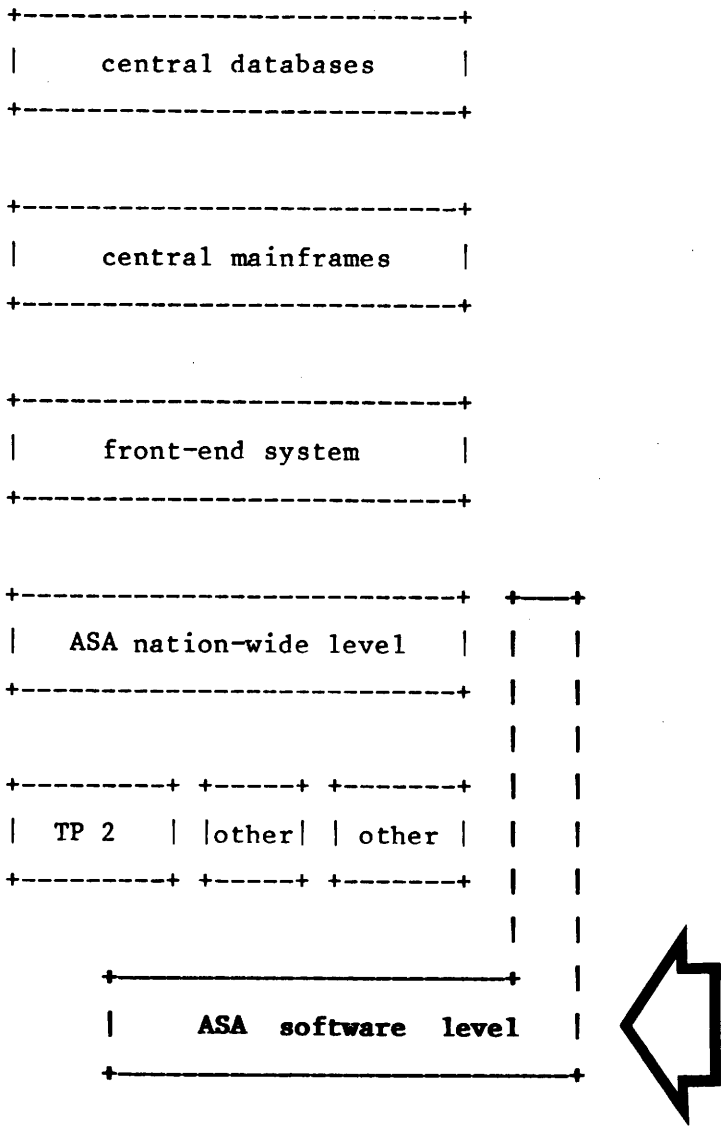
The total system architecture of the entire branch office system is, as described in chapter II, build around five major parts: the central databases, the central mainframes, the front-end system, the network and the decentralized equipment. It is important to notice, that ASA leaves the centralized part of this entire system intact, with the only restriction, that the central part of the software may in time change.

Moving from TP2 to ASA will be a long project with several intermediate migration steps to control the progress. Anyhow, ASA is formed, so these migration steps are facilitated rather than requiring a complete new system from the start.

As such, ASA will at the start implement the nation-wide level by standardizing the public data network interface and allowing non-Olivetti equipment to access this network:

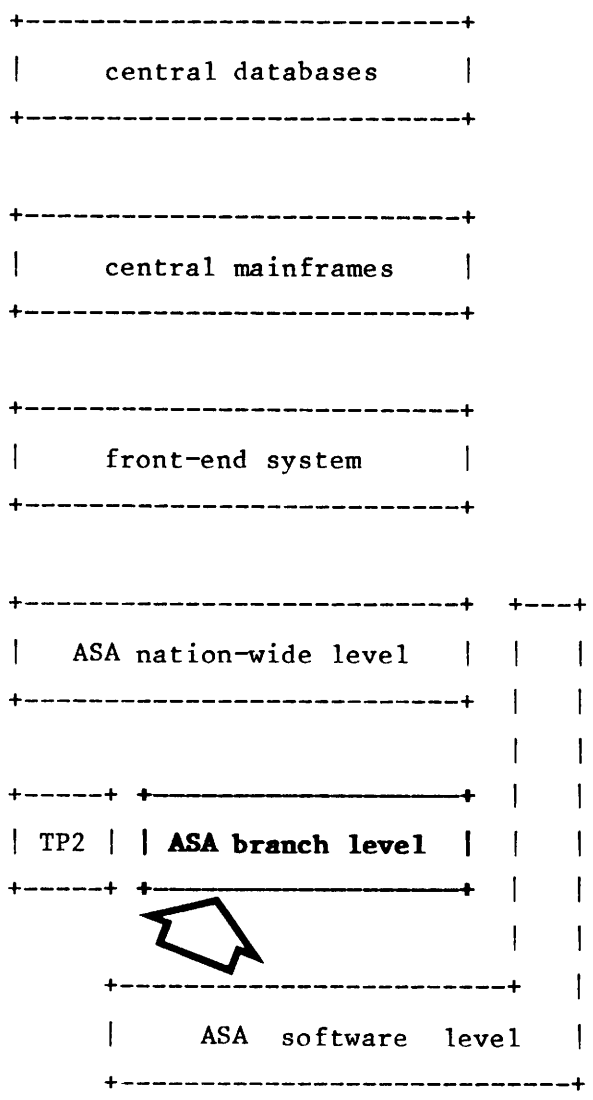


The next step will be to define the software level and make that level cover both new systems attached to the network and possibly TP2 systems:



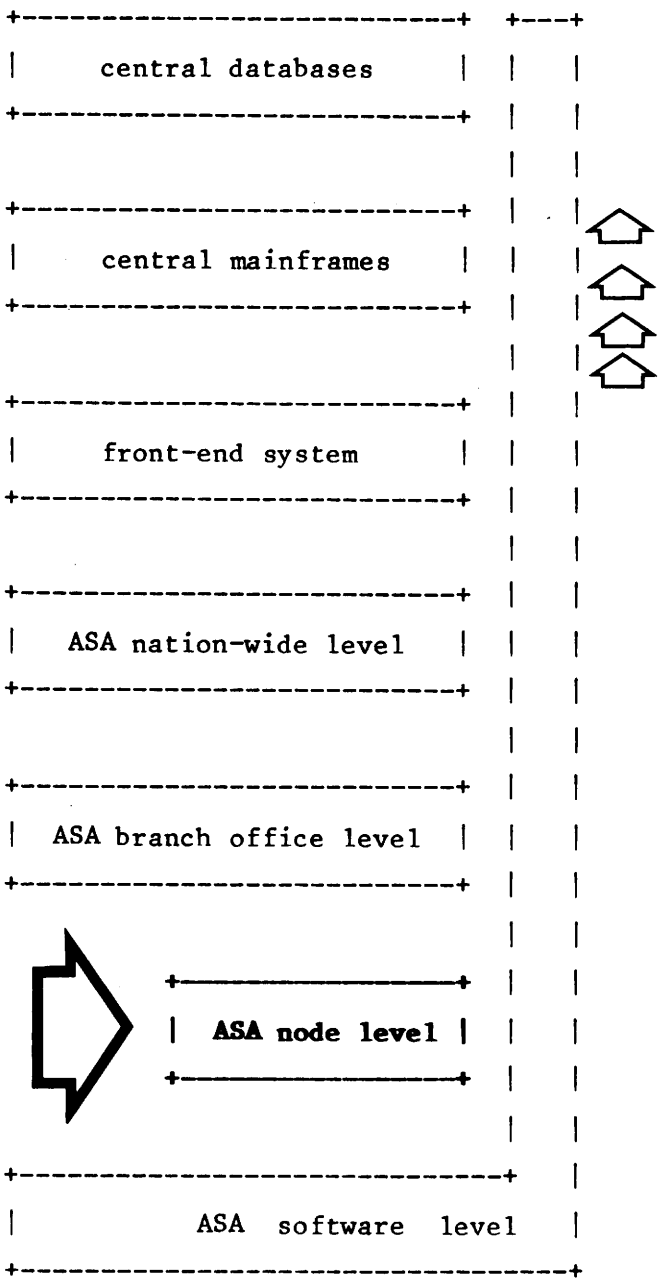
When the software level is ready, it will allow applications on different processors within the system to intercommunicate, as well as the interface towards the central mainframes has been standardized. This will allow SDC to distributed the current LC applications in TP2 to separate processors at each user.

Then these processor systems should, as well as other modules, which connects to the nationwide level, conform to the ASA branch office level and possibly SDC could start to exchange or integrate TP2 into this level:



Together with this transition, the ASA node level may be implemented. This can also happen before or after, dependent on the type of branch office node modules, that is needed:

At the same time, the ASA software level may be extended to cover all parts of the entire SDC banking system:



As such, the ASA model can be used as a step-wise introduction of new concepts in the design of decentralized systems for the Danish savings banks.

8.8 Conclusion

ASA defines the framework and concepts for future migrations of the decentralized system for the savings banks. The preceding paragraph gave an overview of the ASA concepts and how the should be introduced in the current system. The next chapter will go into technical details on specific migration activities towards ASA and the use of new technology in the branch office systems.

This page intentionally left blank

CHAPTER NINE

TP2 TO ASA MIGRATION ACTIVITIES

9.0 Introduction

This chapter will concentrate on migration activities, that will help the current TP2 system to migrate to the leveled system architecture ASA proposed in chapter 8. It will also discuss software migration activities for the technical migration proposals presented here, as well as organizational impact on SDC.

9.1 Migration towards ASA

To migrate towards the ASA concept, several steps will have to be taken, to control this migration and to keep as much as possible of the initial investments in TP2 intact.

The target for migration will be to:

- provide flexibility and change in the decentralized system
- provide greater performance
- maintain software investments and allow for using standard software packages.

My proposals will be oriented towards a support for the ASA nation-wide level, a support for the ASA branch level, and some support for the ASA node level.

One should note, that the basic impact of ASA compared to the current TP2 architecture is to remove the LC concept and keep the TC concept, however with larger TC performance and mutually independent TCs. As such, application execution will reside in the TCs, whereas the other two functions of the LC, the file system and the network interface, can be located in special servers on the local area network or on the TCs.

9.2 ASA nation-wide level migration

This part of the migration has the purpose of easing the interface of new systems to the nation-wide network. This will be needed soon in the future, as new type of systems come along. These systems will include decentralized voice response systems, customer systems or even systems for the smallest savings banks.

A solution for the nation-wide interface will also be needed, if changes are made in the module structure of the current TP2 system, e.g. by removal or replacing of the LC.

Connection to the nation-wide level requires, that a manufacturer of systems must know and implement the SDC tp-monitor concept. This can be a problem and a delaying factor in projects, that wish to use other computers than Olivetti's local computer.

As was shown in the practical case of the EF-57 project (see chapter 10), the amount of hardware and software needed to support the tp-monitor is reasonable, provided that one has a basic know-how on how tp-monitor interacts with SDC's central

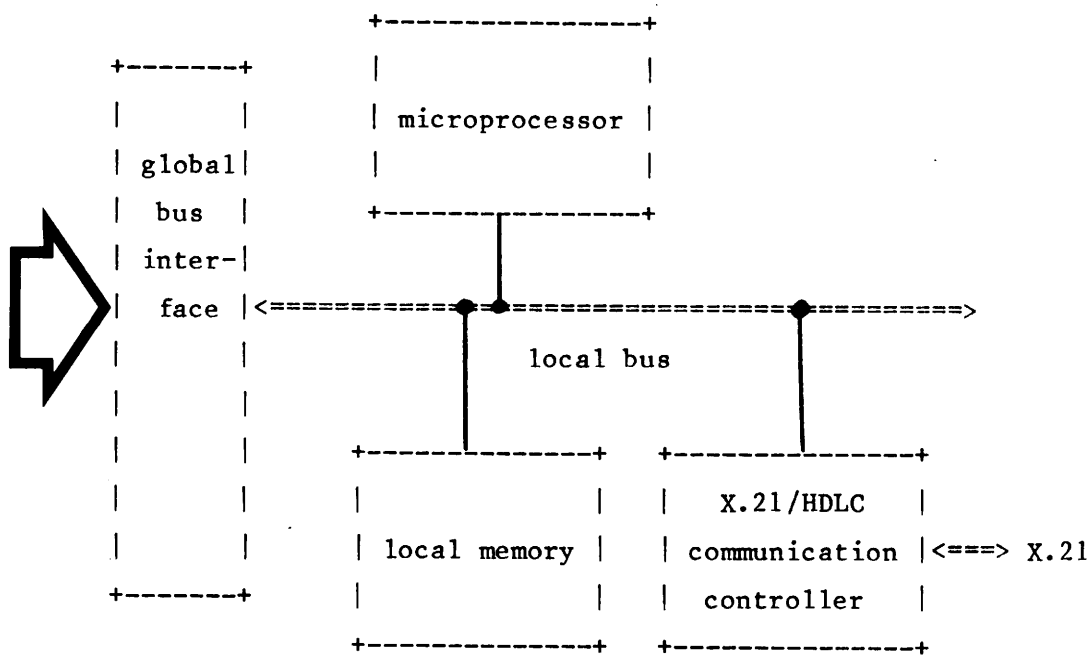
systems.

Tp-monitor can be considered as being self-contained seen from the applications point of view. As such, the layers down from the End-to-End layer in tp-monitor is actually system independent, provided that a suitable interface strategy between the applications and the End-to-End layer is defined. Hence, the tp-monitor can be separated as a self-contained function running on a private processor in any system design.

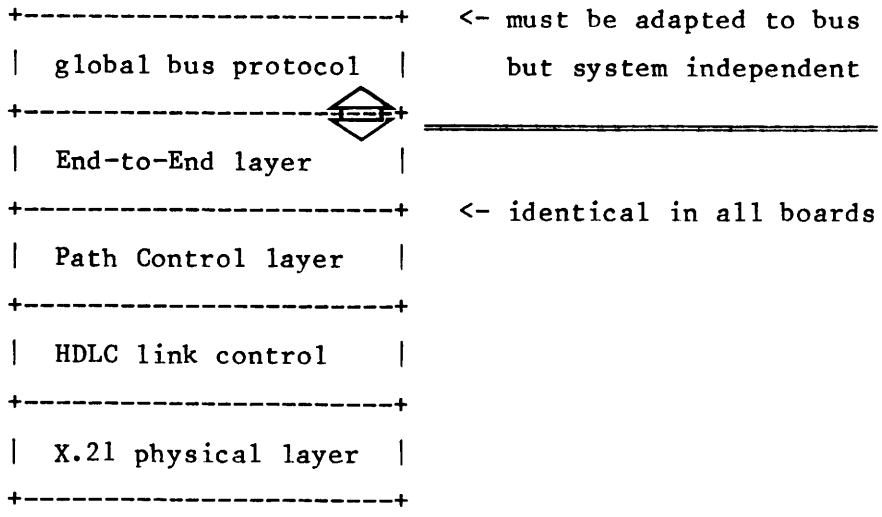
Hence, SDC should insist on separating the tp-monitor functions from applications on separate processors in any future system design. This can be done straightforward by implementing the following proposal:

- SDC should design its own tp-monitor processor, based on the know-how obtained in the practical part of this project. This processor will consist of a network interface, microprocessor and memory, and can easily be designed to fit on a single circuit board on any of the existing bus standards of interest.

- Hence, basically the same design can be used for implementing a MULTIBUS tp-monitor board, a VMEbus tp-monitor board or a IEEE-896 tp-monitor board. The only difference will be the interface towards the global bus:



Software will be identical except for the message interface used on the various bus architectures:

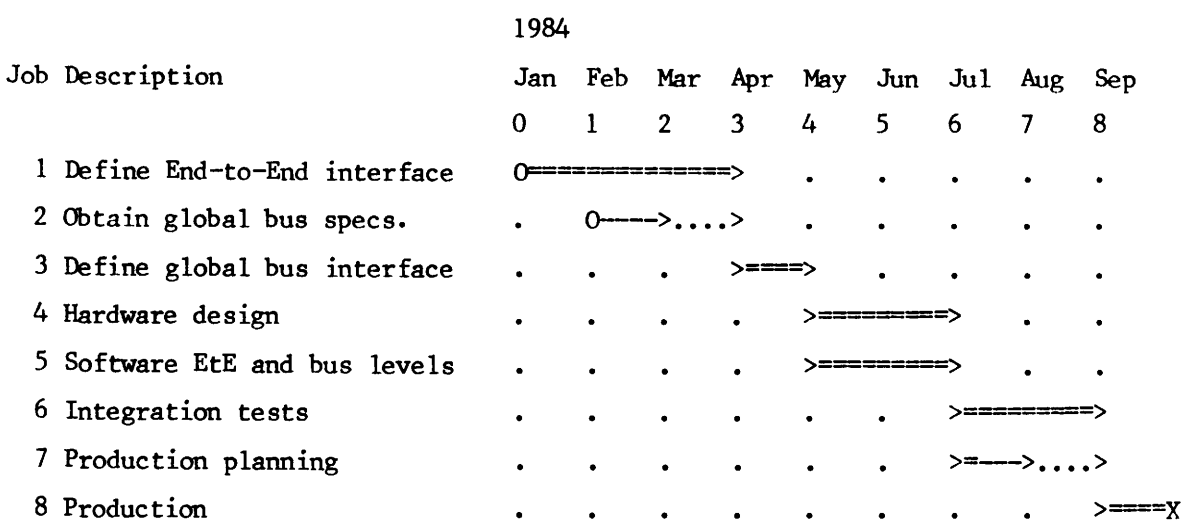


Using this design approach to implement a tp-monitor controller board for MULTIBUS, VMEbus and IEEE-896, will allow SDC to be able to choose freely among a wide range of systems with a guarantee, that these will be able to communicate with SDC. What will be needed on these systems, is a small device driver, that can transfer messages from applications through the operating system on the main processor to the tp-monitor controller. The development time for such a device driver will be considerably less than any other attempt, and it keeps the fully control of changes in the tp-monitor specifications within SDC.

Secondly, it will facilitate any further migration activities towards the ASA branch level and the ASA node level.

Time schedule for designing a board like this will dependent on how easy the tp-monitor concept are changed to support a message passing method. The following time table shows the fundamental activities, beginning at an arbitrary starting date of January 1st, 1984:

Tp-monitor board design 1st vers, Revision 2, 8/1/83
Prepared by Tue Bertelsen

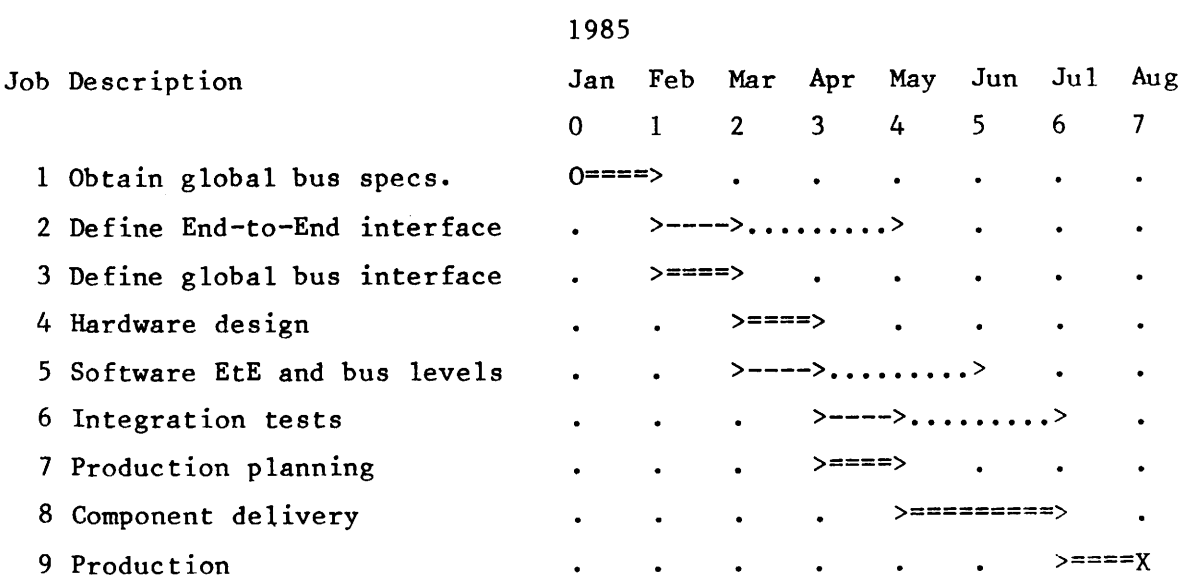


Symbol - Explanation

>---->	Duration of a normal job
>....>	Slack time for a normal job
>====>	Duration of a critical path job
>::::>	Duration of a completed job
*	Job with zero duration
O---->	Job with no prerequisites
>----X	Job with no successors

Boards for other buses will be easier, as the fundamental hardware and software design has already been done. Hence any subsequent buses will be implemented faster, as shown on the next time table:

Tp-monitor board design, new bus, Revision 3, 8/1/83
Prepared by Tue Bertelsen



Symbol - Explanation

>---->	Duration of a normal job
>....>	Slack time for a normal job
>====>	Duration of a critical path job
>::::>	Duration of a completed job
*	Job with zero duration
0---->	Job with no prerequisites
>-----X	Job with no successors

As such, the design can be carried out within a reasonable time frame.

The production cost for such a board will be comparable with the price level on the general board market for a given bus.

An estimate of design activities in man-power for such a project is difficult, as many will participate for short periods, but it will probably be less than one and a half man-year.

9.3 TP2 performance enhancements

Performance enhancements in TP2 will be achieved by implementing the new technologies within the existing TP2 system. The TP2 system is not designed for this, so the proposals in this paragraph will require, that SDC itself performs a complete redesign of the interior of the system.

This approach is not consistent with the proposed ASA node level architecture, but it will support the ASA branch level architecture, thus being a viable step, that can justify the investments.

The place to start is to redesign the current terminal computer to use newer technology. This should be done by the following strategy:

The TP2 terminal computer is built around a set of circuits boards with:

1) the processor and memory
2) the display controller and memory
3) the high speed link controller
4) the mini modem for the high speed link
5) serial controller boards

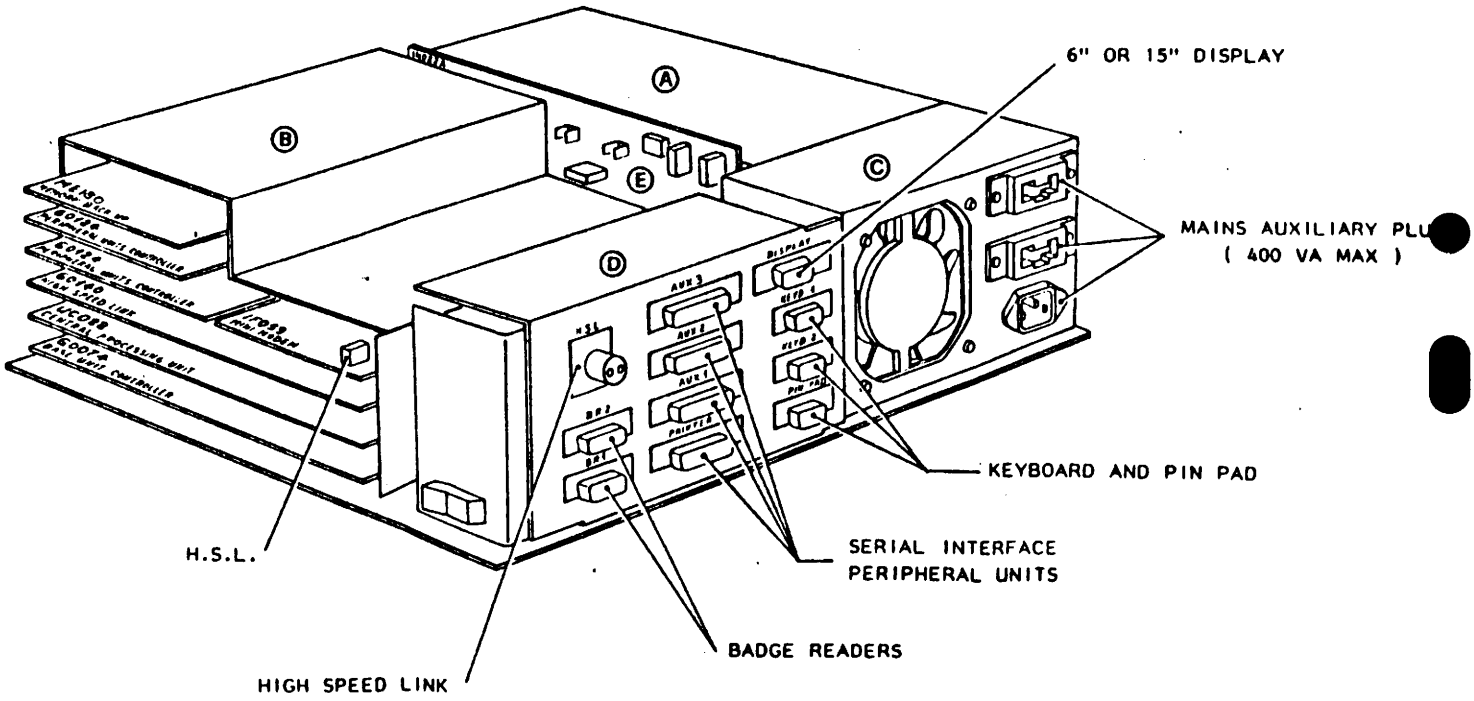


Figure 9.1: Printed circuit boards in TC

Ⓐ-ALI 239 AND LINE FILTER

Ⓑ-CARD CAGE

Ⓒ-MAINS GROUP AND FAN

Ⓓ-EXTERNAL PLUGS FOR
PERIPHERAL UNITS

Ⓔ-MOTHER BOARD

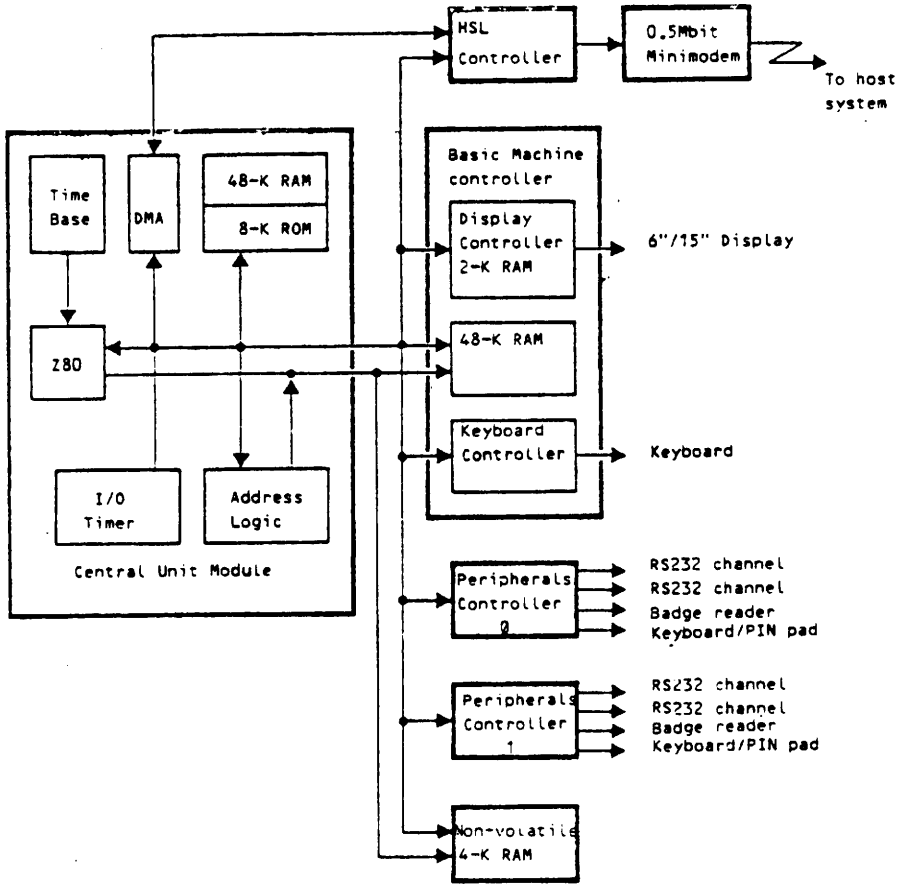


Figure 9.2: TC1808 block diagram

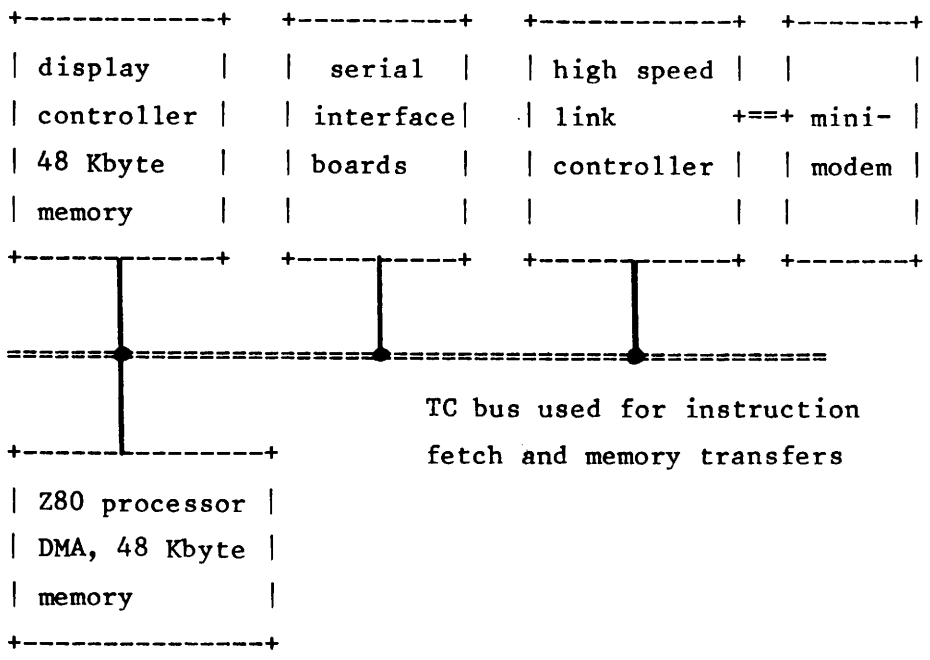


Figure 9.3: TC reference block diagram

A new processor board should be designed using the new Z800™ chip from Zilog (see Whitcomb, 1982; Carter, 1983). Z800 is an enhanced Z80, that is completely Z80-compatible, i.e. code written for Z80 will execute directly without any change. Z800, however, has additional features, which can be used as a part of software change, and where SDC may benefit:

First, Z800 is a high speed processor, with versions in the range from 10 MHz to 25 MHz. This should be compared to the current TC's speed of 2.5 MHz.

Secondly, Z800 has built-in memory management. It comes in two versions: one that addresses up to 512 Kbyte of memory and one that addresses up to 16 Mbyte of memory. The built-in memory management means, that address translation takes place immediately without any reduction in speed, as may be the result using external memory management units, like with the Z8000. Also, Z800 has a built-in high-speed cache memory of 256

bytes, which means a further improvement of performance in those cases, where iterative code or commonly used data are accessed.

Each of the two versions described above comes in two bus versions, one which use the traditionally 8-bit Z80 bus, and one, which uses the 16-bit ZCI component bus. Choosing the 16-bit bus version allows potentially higher speed, as the bus transfer rate is doubled. Also the ZCI component bus is the same as used by the 16-bit Z8000 and the 32-bit Z80000 (see Banning, 1979; Zilog, 1983).

Furthermore, Z800 has more instructions than Z80. These instructions include 16 and 32 bit hardware multiply and divide and a more flexible addressing scheme, giving shorter and faster code, especially for PLZ code generation. Also, the concept of user/system states are included as on the Z8000 with privileged operations and system calls. Also test-and-set operations for semaphore handling is included. The memory management utility allows for instruction restart, which means that virtual memory can be implemented.

Timing is programmable for allowing interface to old type of peripheral chips and the Z800 even contains its own on-chip peripherals, including timers, 4 DMA channels and a 1 Mbit/second asynchronous serial interface! The chip can even boot itself through this serial line in a communication environment.

The migration proposal for the TC will be to design a new processor board for the TC with the following characteristics:

10 MHz Z800 with 16 Mbyte addressing and 16-bit bus
64 Kbyte PROM storage for current TC boot program
1 Mbyte RAM storage using 256 Kbit chips
existing TC processor board peripherals
new floating-point coprocessor chip
new data encryption chip
socket for 32-bit Z80000 using the 16-bit bus.

The board will be provided with the current TC bus interface to the other existing boards in the TC, as well as the memory bank switching circuitry in the current TC. The 10 MHz clock will by the processor be divided to 2.5 MHz for usage for existing chips in the system. Existing memory (non-volatile and the 48K on the basic controller board) will be used as always.

As peripheral chips on the board will be the same as on the old board and with the same physical addresses, this will immediately give a higher performance TC without any changes to existing software other than 10 instructions in the TC boot PROM to select the programmable timing of the Z800.

Hence, this approach do not need any changes in applications development at SDC and software will be identical at the binary code level for the new TC and the old TCs. However, software will still be enforced the same memory addressing restrictions as on the old TC.

For convenience, SDC's new version of the TC will be called the STC.

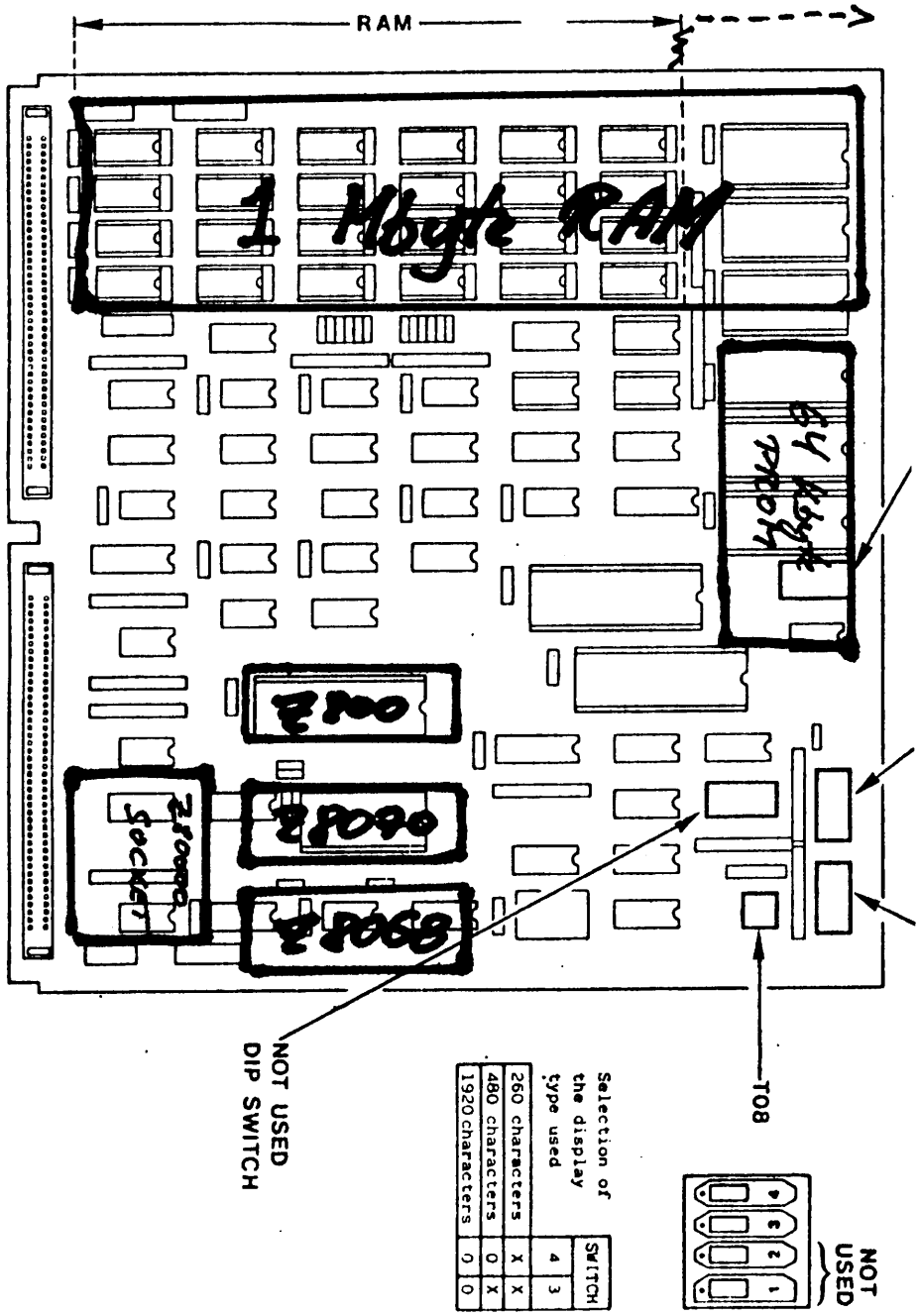


Figure 9.4: Z800 component placement on old TC CPU board

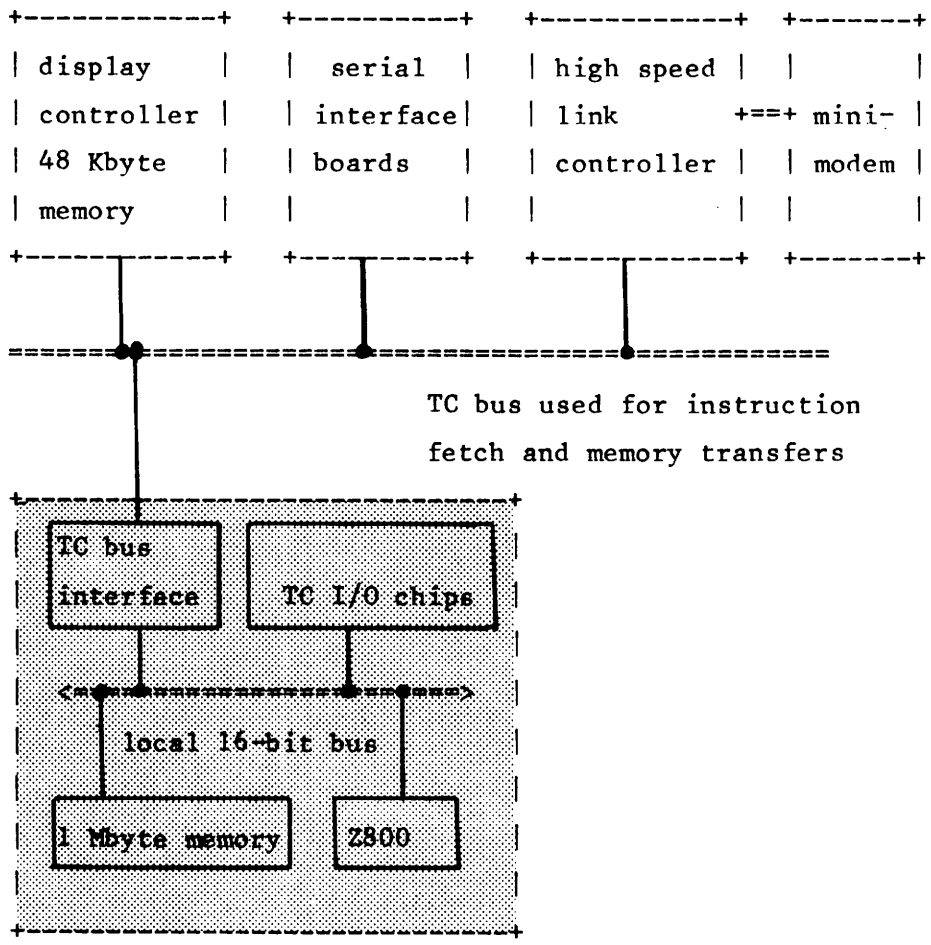


Figure 9.5: STC overview

When SDC is mature, the next steps can be taken: A new TCOSMOS operating system version can be made for the Z800, utilizing the bigger memory and the higher performance instructions. This should be accompanied by a Z800 code generator for PLZ/SYS. This code generator will require, that SDC get the source code from Olivetti, which should be in accordance with the joint-venture agreement between SDC and Olivetti. (Microlab has the original Zilog source code for PLZ code generator, but this does not include the float type implemented in SDC's version). This step should also be used to change the specification of procedure parameter passing, so it becomes compatible with the

original PLZ conventions.

Rather than rewriting the TCOSMOS, SDC should by this step implement a standard operating system, like CP/M or an 8-bit UNIX derivative on the new STC.

Besides implementing a new operating system on the STC, the following hardware enhancements may be designed to replace the other old boards of the TC:

Removal of the old display controller board with a new, microprocessor based controller. This will release the main processor from the burden of being a screen controller and allow for more ergonomic and faster display (programmable fonts, APL-characters, ANSI control codes, IBM control codes, high resolution graphics, etc.)

Implementing possibly intelligent serial controllers with internal processor and memory, communicating through a FIFO bus with the main Z800.

Removal of high speed link controller board and mini-modem board with an Ethernet or IEEE-802 controller, allowing the STC to connect to a local network, possibly the ASA branch level.

Replacing the processor chip with a faster speed version, e.g. 25 MHz.

More memory. The initial version of CPU board will be designed with 256 Kbit chips, which can be replaced with 1Mbit chips. This gives 4 Mbyte of memory available on the CPU board.

Further migration steps can be taken by replacing the Z800 with the Z80000 32-bit microprocessor. Since the Z80000 will use the 16-bit bus, the full speed advantage may not be obtained, but it will still have a throughput about 1 MIPS (see Alpert,

1983). Also software will have to be compiled, and a new operating system must be implemented. However, since the new operating system for the Z800 will be written in high level language, and the other controller boards are self-contained with their own processors, this is a rather feasible way to go.

This will give two features: the STC will have greater performance than Olivetti's current L1-line (!) and the STC will be able to run both SDC applications and standard software packages. Also, the interface towards any new system architecture using local area networks is there. Probably the most significant fact is an enhanced freedom in changing the performance options for the STC:

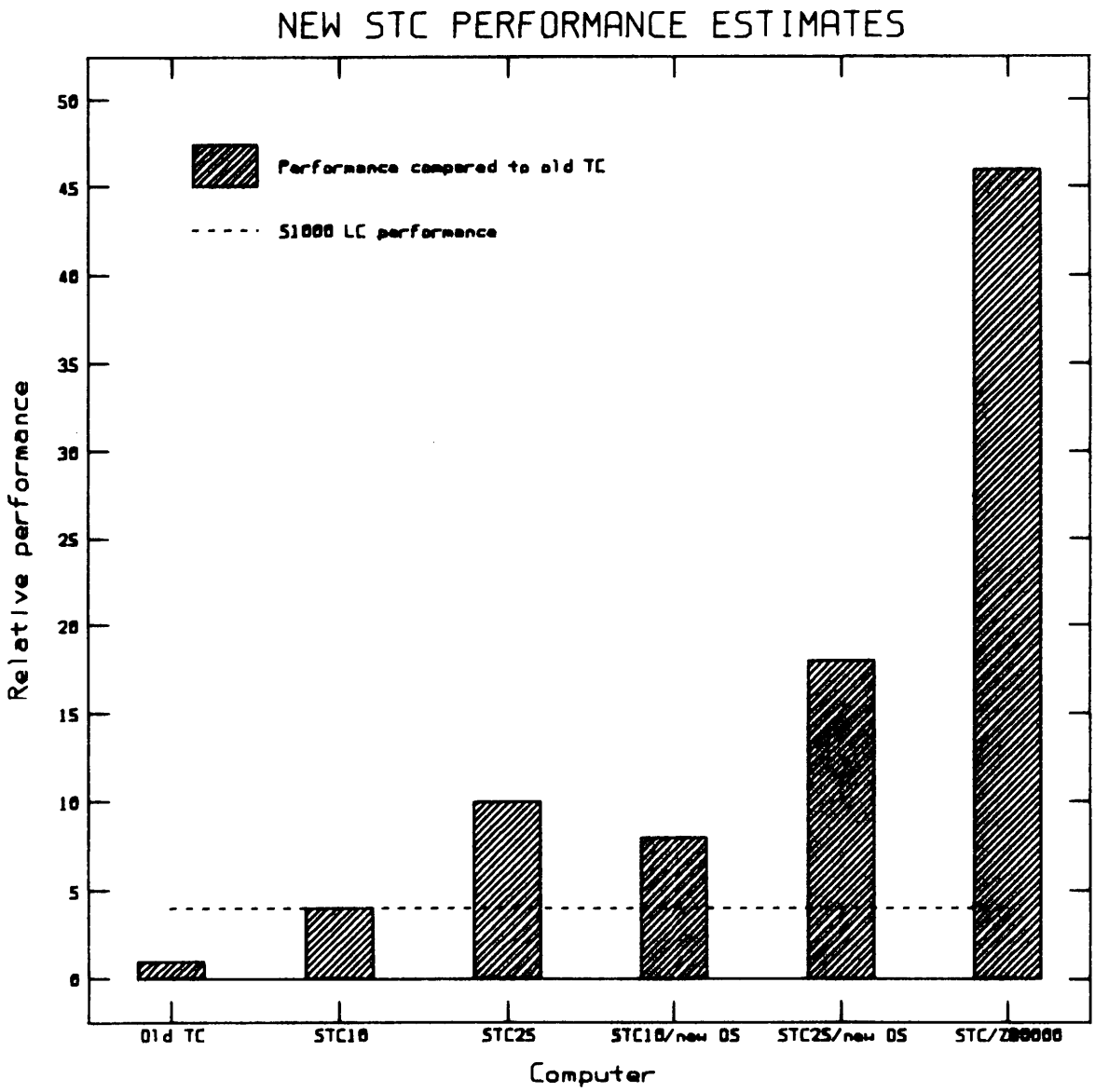


Figure 9.6: STC relative performance estimates:

†: STC10 = based on 10 MHz Z800
STC25 = based on 25 MHz Z800

The performance estimates above are not instruction execution performance relationships, but the expected performance seen from the applications point of view. A new PLZ code generator will contribute significantly to the performance increase, as the new addressing schemes with 16-bit transfers can be utilized, as well as the 16-bit arithmetic operations implemented in the Z800 hardware will contribute to a faster handling of array and structure indexing, which require software computation on the Z80.

What has been done is by a stepwise manner to change the internal architecture of the TC from being a restricted performance processor to become a multiple processor system, where specific functions are each assigned their own processing power, and hence making these functions independent of other functions in the system. Also, the TC bus has been changed from being an instruction fetch bus to a message passing bus:

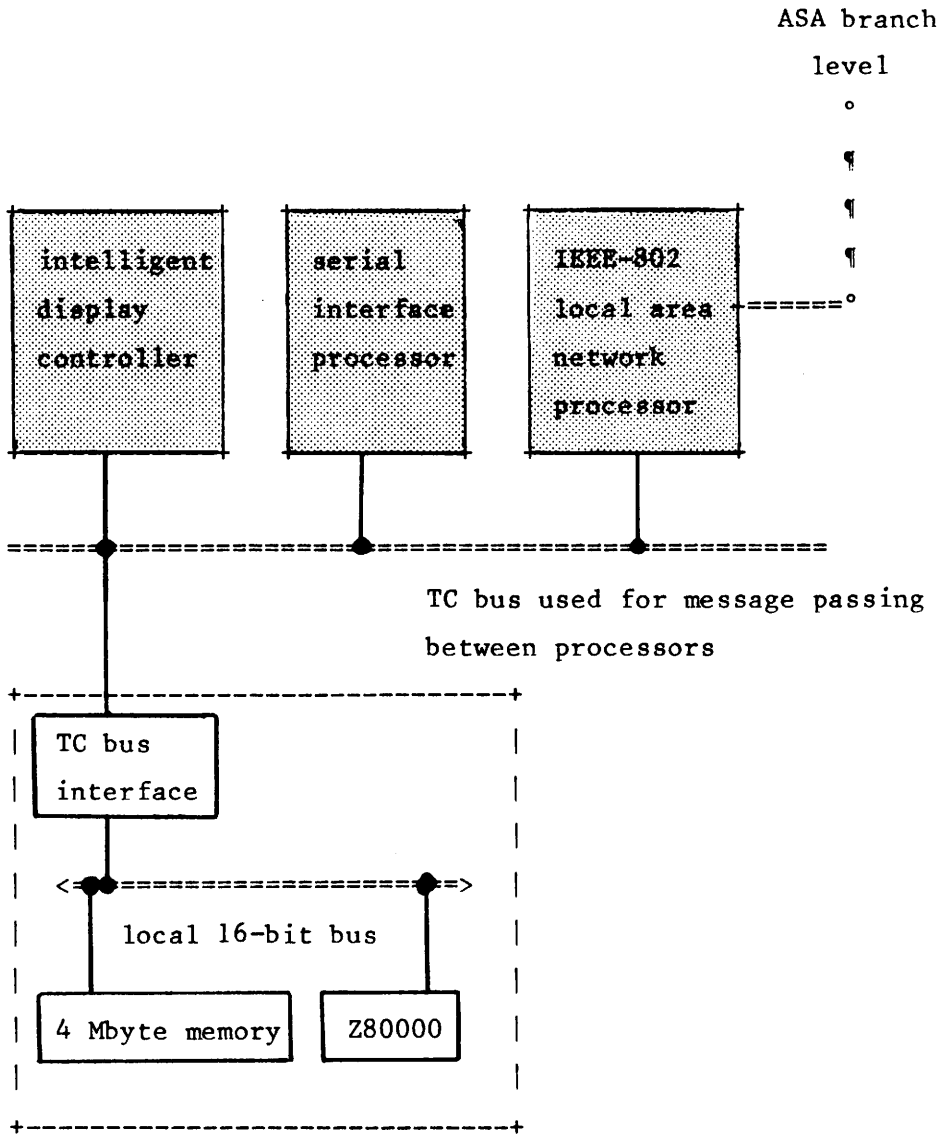


Figure 9.7: high performance STC ASA branch component

By giving the TC a reasonable amount of memory and processing power, the LC of the TP2 system loses its role. In fact, what is needed is a central file server and a gateway to the ASA nation-wide level.

Gateways to the public data network will use the tp-monitor processor board described above. File servers can implemented by using off-the-shelf systems and associated high level protocols or they can be implemented locally on the STC by using the new sub-4" disk products.

Basically, this solution allows SDC to re-utilize the "mechanical" parts of the TC: cabinet, power supply, connectors, backplane bus, cables etc. Also, installation is made by board swapping, which is a matter of minutes. Secondly, it gives SDC time to plan or alter any software changes.

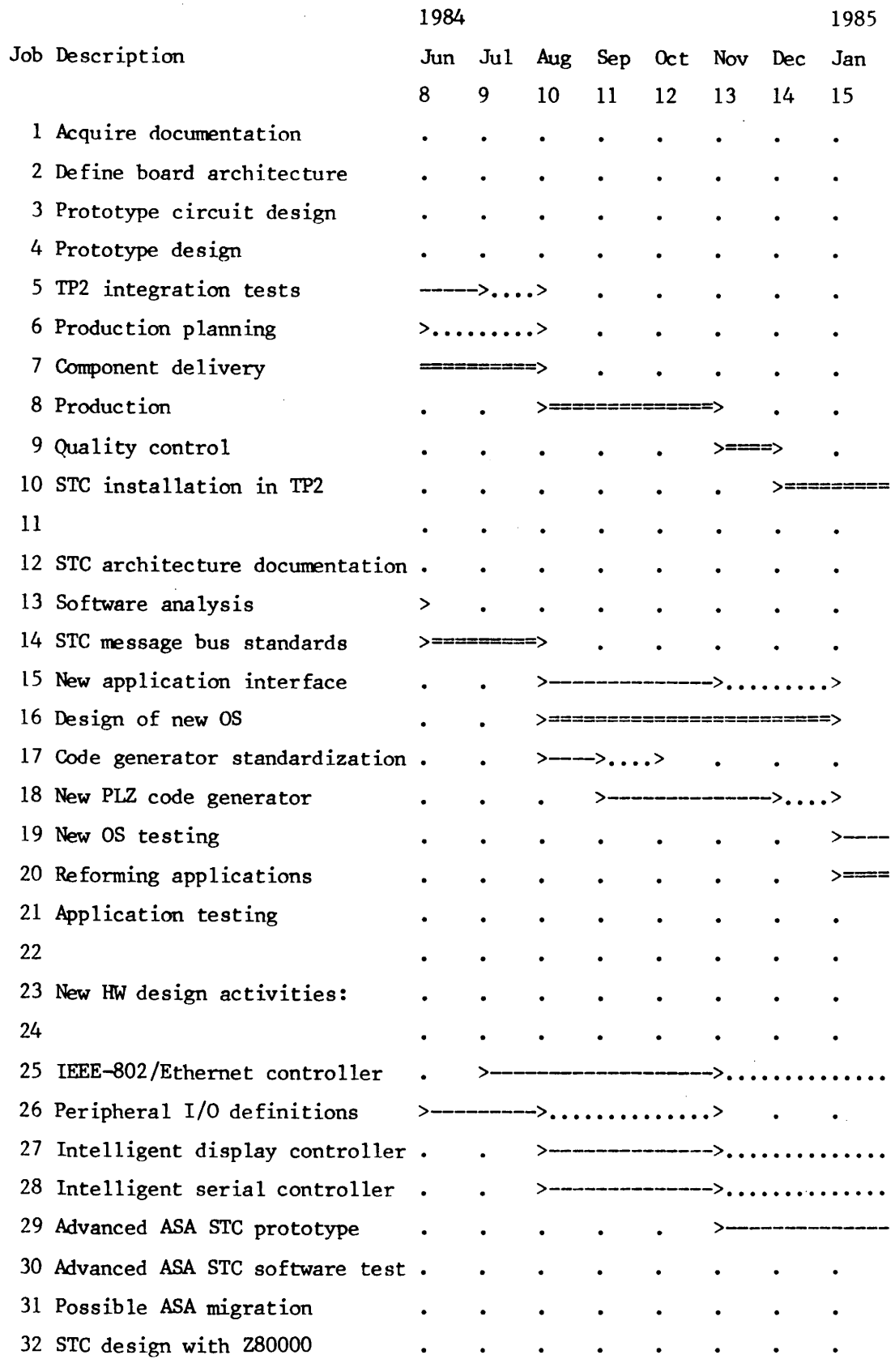
The estimated time schedule for implementing the STC is shown on the following pages. It includes the schedule for the basic STC design, and related schedules for the STC using the new OS and equipped with the new boards, suiting the STC to the ASA branch architecture.

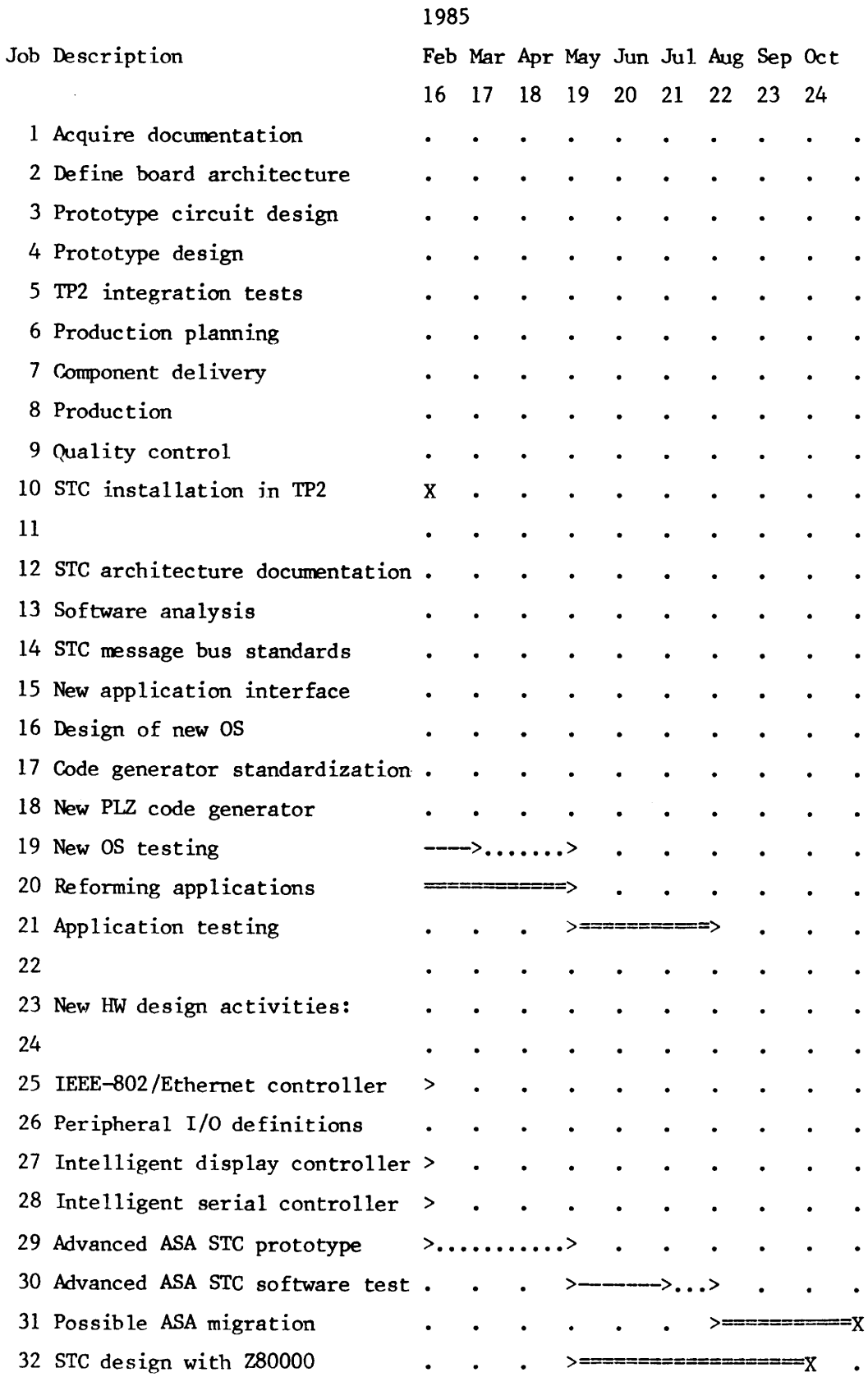
The following notation is used:

Symbol	- Explanation
>---->	Duration of a normal job
>....>	Slack time for a normal job
>====>	Duration of a critical path job
>::::>	Duration of a completed job
*	Job with zero duration
0---->	Job with no prerequisites
>----X	Job with no successors

New TC Z800 design, Revision 3, 8/1/83

Job	Description	1983 Oct 0	Nov 1	Dec 2	1984 Jan 3	Feb 4	Mar 5	Apr 6	May 7
1	Acquire documentation	O--------->	.....	>	.	.	.		
2	Define board architecture	.	O========>		.	.	.	.	
3	Prototype circuit design	.	.	.	>====>	.	.	.	
4	Prototype design	.	.	.	.	>==========>			
5	TP2 integration tests	.	.	.	.	.	.	.	>----
6	Production planning	.	.	.	.	.	.	.	>----
7	Component delivery	.	.	.	.	.	.	.	>====
8	Production	.	.	.	.	.	.	.	.
9	Quality control	.	.	.	.	.	.	.	.
10	STC installation in TP2	.	.	.	.	.	.	.	.
11		X	.	.	.	.	.	.	.
12	STC architecture documentation	.	.	.	.	>====>		.	.
13	Software analysis	.	.	.	.	.	>==========		
14	STC message bus standards	.	.	.	.	.	.	.	.
15	New application interface	.	.	.	.	.	.	.	.
16	Design of new OS	.	.	.	.	.	.	.	.
17	Code generator standardization	.	.	.	.	.	.	.	.
18	New PLZ code generator	.	.	.	.	.	.	.	.
19	New OS testing	.	.	.	.	.	.	.	.
20	Reforming applications	.	.	.	.	.	.	.	.
21	Application testing	.	.	.	.	.	.	.	.
22		X	.	.	.	.	.	.	.
23	New HW design activities:	X	.	.	.	.	.	.	.
24		X	.	.	.	.	.	.	.
25	IEEE-802/Ethernet controller	.	.	.	.	.	.	.	.
26	Peripheral I/O definitions	.	.	.	.	.	.	.	.
27	Intelligent display controller	.	.	.	.	.	.	.	.
28	Intelligent serial controller	.	.	.	.	.	.	.	.
29	Advanced ASA STC prototype	.	.	.	.	.	.	.	.
30	Advanced ASA STC software test	.	.	.	.	.	.	.	.
31	Possible ASA migration	.	.	.	.	.	.	.	.
32	STC design with Z80000	.	.	.	.	.	.	.	.


1984
1985
Job Description
Jun Jul Aug Sep Oct Nov Dec Jan
8 9 10 11 12 13 14 15
1 Acquire documentation
2 Define board architecture
3 Prototype circuit design
4 Prototype design
5 TP2 integration tests
6 Production planning
7 Component delivery
8 Production
9 Quality control
10 STC installation in TP2
11
12 STC architecture documentation
13 Software analysis
14 STC message bus standards
15 New application interface
16 Design of new OS
17 Code generator standardization
18 New PLZ code generator
19 New OS testing
20 Reforming applications
21 Application testing
22
23 New HW design activities:
24
25 IEEE-802/Ethernet controller
26 Peripheral I/O definitions
27 Intelligent display controller
28 Intelligent serial controller
29 Advanced ASA STC prototype
30 Advanced ASA STC software test
31 Possible ASA migration
32 STC design with Z80000


1985
Job Description
Feb Mar Apr May Jun Jul Aug Sep Oct
16 17 18 19 20 21 22 23 24
1 Acquire documentation
2 Define board architecture
3 Prototype circuit design
4 Prototype design
5 TP2 integration tests
6 Production planning
7 Component delivery
8 Production
9 Quality control
10 STC installation in TP2
11
12 STC architecture documentation
13 Software analysis
14 STC message bus standards
15 New application interface
16 Design of new OS
17 Code generator standardization
18 New PLZ code generator
19 New OS testing
20 Reforming applications
21 Application testing
22
23 New HW design activities:
24
25 IEEE-802/Ethernet controller
26 Peripheral I/O definitions
27 Intelligent display controller
28 Intelligent serial controller
29 Advanced ASA STC prototype
30 Advanced ASA STC software test
31 Possible ASA migration
32 STC design with Z80000

There are many critical activities in this project, especially those concerning software development. Another critical factor is the delivery of the Z800, which is expected to go into production first or second quarter 1984.

The project is currently being seriously evaluated within SDC, and contacts have been made with Zilog to get the first samples of the Z800 with the 16-bit bus.

A more detailed project schedule will be made, which takes into account the man-power needed at the different stages of the project. The hardware design project alone can take place within Microlab, SDC. The full-blown project will include people from almost every area within SDC.

One should note, that the STC architecture is not compatible with the ASA user node level architecture. It is however compatible with TP2 and can migrate to ASA branch office level in time, while at the same time offering a performance migration path.

No detailed estimates have been made of the production cost yet, but in large quantities (for the more than 4000 terminal computers) it will probably be around 5000 Dkr. This amounts to about 20 million Dkr., if the total TP2 system must be equipped with these boards. Even if the cost goes further down, it is an investment, that must be considered thoroughly.

Further investments of the same kind must be considered, if it is decided to migrate to the ASA branch office level, as this will need the local network controller board.

9.4 Software migration activities

The STC described above will in its initial version only provide a performance enhancement for the TP2 software currently residing on the TC. Any software running on the LC will still have the current performance penalties. However, the initial STP step can be performed without any application software changes at all, since the STC will be directly compatible with the old TC.

Implementing an new operating system on the STC and a new PLZ code generator, further performance enhancements can be achieved. These will however necessitate a movement of current application programs from the LC to the STC, hence imposing a need to remove the old master/slave relationship between LC/TC applications. This step can be made in two ways: (1) by changing the complete branch system to make full usage of the STCs or (2) by stepwise reconfiguring the STCs within a branch.

The problem involved in the latter approach is, that a LC will be needed with software, that supports both the old TC software and the new STC software. Probably, this will even further reduce the performance experienced by the users running the old TC software in the system. Hence, the former approach will be more feasible, as it will include a complete new LC software design as well, where this software only is responsible for file handling and public data network communication.

Still, however, this software will suffer from the same problems with the LC's file system and the low effective data rate of the high speed link, as exist today, and this can be prohibitive, if the LC must support frequent program loads of the large programs, that the STC can handle.

This can be solved by implementing local disk storage capacity on the STC for handling program files and user data files, but that will impose extra costs on the STCs rather than the direct board swapping costs.

Another point to realize is, that the applications that will use secondary storage most, will be the applications related to the back-office and office automation environments. Even though a minor delay can be tolerated during infrequent program loads, these applications will anyway need a much higher transfer rate for intermediate disk operations, than the TP2 high speed link can support.

A feasible solution will therefore be to restrict the use of the LC to teller applications, and instead implement back-office and office automation applications on new equipment, that is either based on off-the-shelf systems, conforming to the ASA branch level architecture or based on the high-performance STC with ASA branch level connection.

Hence, the ultimate best approach will be to drop the LC as soon as possible, replacing the high speed link with a local network, and using the tp-monitor board processor together with off-the-shelf microcomputer boards to implement a public data network server, attachable to the local network.

To do this, the application relationships must be documented and it must be defined, how applications should interact through the ASA software level message bus, when separated on individual processors. Probably the best way to do this would be to drop further developments on office automation on TP2, and instead start afresh using new computer systems, either supermicros or personal computers, with a standard operating system and possibly an interface to the SDC network through either the tp-monitor processor board described earlier in this

chapter or through the use of the McBusiness product described in chapter 10.

Although this approach will not be immediately accepted in SDC, it will leave the TP2 system available for teller applications, which can be controlled, managed and debugged without any risk for problem impact due to the implementation of the other applications on a already fully loaded system. Also, SDC will be freed from complaints from the savings banks on the performance of office automation functions compared to the standard market offerings.

9.5 Organizational impact

All of these migration activities will have a major impact on the SDC organization. The impact will come in two areas:

- As SDC will design or supervise the design of the STC, the tp-monitor processor board and the associated operating system and language software, a more general awareness and hands-on know-how of the products and standards within the microelectronic and data communication area will be needed within SDC and a group of people must be established to carry out the task of implementing a new operating system and a new code generator.

- As the ASA system architecture allows for multi-vendor systems, and as future products besides the STC are not likely to be designed by SDC, a much more general product know-how on the market, together with a built-in requirement for documentation on the inter-communication aspects between applications must be achieved within SDC.

This will require, that a larger part of the working force in SDC are brought in contact with the general system market, and are allowed to choose between systems to serve a specific task or requirement.

Also, data communication activities should be significantly enforced in SDC to include the local area network area and the higher level standards, that takes care of information communication, rather than the raw communication of data at the network level. Also, data communication know-how should be communicated to all areas of decentralized application development.

Of course, this will be a large task, but the result will most likely not be an increased need for man-power within SDC, as much of the work, that exist today in solving trivial problems or cutting down requirements so they fit to TP2, will disappear.

The management of a multi-vendor environment within the branch offices may require some analysis before it can be handled properly. The requirements for managing this will be:

- specification of all interfacing standards used in the branch system

- a technical oriented quality control, which knows these standards and which can check, that they adhere to the rated performance seen from the application level.

- a software oriented quality control, that posses the needed hardware know-how to check, whether system software can support the evolving hardware options, and that SDC-developed general applications do not rely on one specific computer or system.

- solving of the possible service and maintenance problems, that may occur if different vendors products are working together.

The efforts involved in this will however be problem-avoiding efforts and they will not be larger than the problem-solving and problem-reporting efforts inherent in the current system.

9.6 Conclusions

As a step towards ASA, SDC may initiate design of new processor boards for the current TP2 terminal computer in order to make this computer technologically up-to-date. Likewise SDC should initiate a separation of the TP2 tp-monitor functions, so they can be implemented on a separate processor board. This will allow SDC to take benefit of the existing systems on the market, that use a standard backplane bus architecture.

The proposed redesign of the terminal computer will allow for performance updates, if the redesign is accompanied by a new software design and if the TP2 high speed link is replaced by a local network in accordance with the ASA branch level architecture.

This page intentionally left blank

CHAPTER TEN

THE MCTAN X.21 PROJECT

10.0 Introduction

This chapter deals with the practical case of the EF-57 project, the McTan X.21 project. As the EF-57 project was interrupted in 1981 by the project, that constitutes this case, we found it in accordance with the goals and objectives of the study. Also, the case is an example of the research and development within the microprocessor area in SDC.

The chapter will contain a description of:

- the background for starting up this case
- the microprocessor developments already made in SDC
- the communication structure of TP2
- implementation details of hardware and software
- test activities
- production activities and manpower
- new products from the project

Running a project like this is not always straightforward. I have tried to describe some of underlying events and factors, which have caused trouble during the project, to show that problems cannot always be handled by the book and products do not always perform according to specifications or expectations. As such, this chapter both contains a technical description of the related issues in the project, as well as an organizationally oriented description.

10.1 Project background

The TP2 system was originally targeted for pilot installation and test in last quarter of 1981 and first quarter of 1982. In June 1981, Rockwell International, who owned the company responsible for the new TP2 front-end system, Collins Radio Company, informed SDC, that due to missing turn-over and financial problems, Collins would cease any further design and development of front-end systems, and activities would be scaled down to just maintenance after the orders on hand in the company had been delivered. Unfortunately, also a one year delay was announced for the new front-end system, which had been targeted for take-over late autumn 1981.

This was catastrophic information for SDC. The new front-end system was needed, because the TP2 system was supposed to communicate with the central mainframes in SDC through the new Danish Public Data Network. The old front-end system could not handle the new type of network interface and front-end systems for public data networks and customized communication protocols as in SDC's case are not off-the-shelf products.

A new front-end supplier would need at least 2-3 years to make a suitable system running, so this was no solution. The backlog in the Collins development was also so big, that at least an one year delay was justified, so this time could not be significant less.

The result would be a one year delay of the entire TP2 project, together with all the implied effects: production schedules would have to be changed, already planned installations in and rebuilding of the savings banks branch offices would have to be rescheduled, planned investments would have to be reconsidered and the education programs of the nearly 10000 employees in the savings banks would be altered.

The TP2 system could not easily be equipped with new communications facilities, nor could the planned network installations performed by the Danish PTT be easily altered.

No testing of the TP2 system could be performed without the network interface and without testing, debugging and optimization of the TP2 software could not be performed, nor could SDC get the experience or know-how needed for further application developments.

10.2 McTan proposal

Several proposals for a back-up solution were considered in SDC, and many parties were involved in the problem. Microlab, which is SDC's microprocessor group, got a request on, whether we had any ideas on how to solve the problem. In previous projects, we had managed to design a microprocessor-based loop line controller for the old type of loop lines used by the TP1 system in SDC, and managed to connect different types of equipment to the SDC network. In an earlier project evaluation, which never was carried out, we had proposed a way to connect some preliminary Olivetti computers to SDC's network through a serial interface to our own loop line controller (see Bertelsen, 1980.4). We were asked, if this old project proposal could be used.

We realized immediately, that this was no good. That proposal was reliant on some protocol development on the Olivetti system, and was not in accordance with the later designed TP2 way of communicating.

After a day's thinking, we came out with this idea: we could design a system, which on one hand would simulate the Danish Public Data Network towards an Olivetti system, and on the other hand would look like an old-type TP1 system, capable of connecting to the old loop lines. This box could then be put in as an interface between the new TP2 systems and the old network and contain software, that would transform TP2-transactions into TP1 transaction and vice versa, so the Olivetti system would not notice any difference (see figure 10.1).

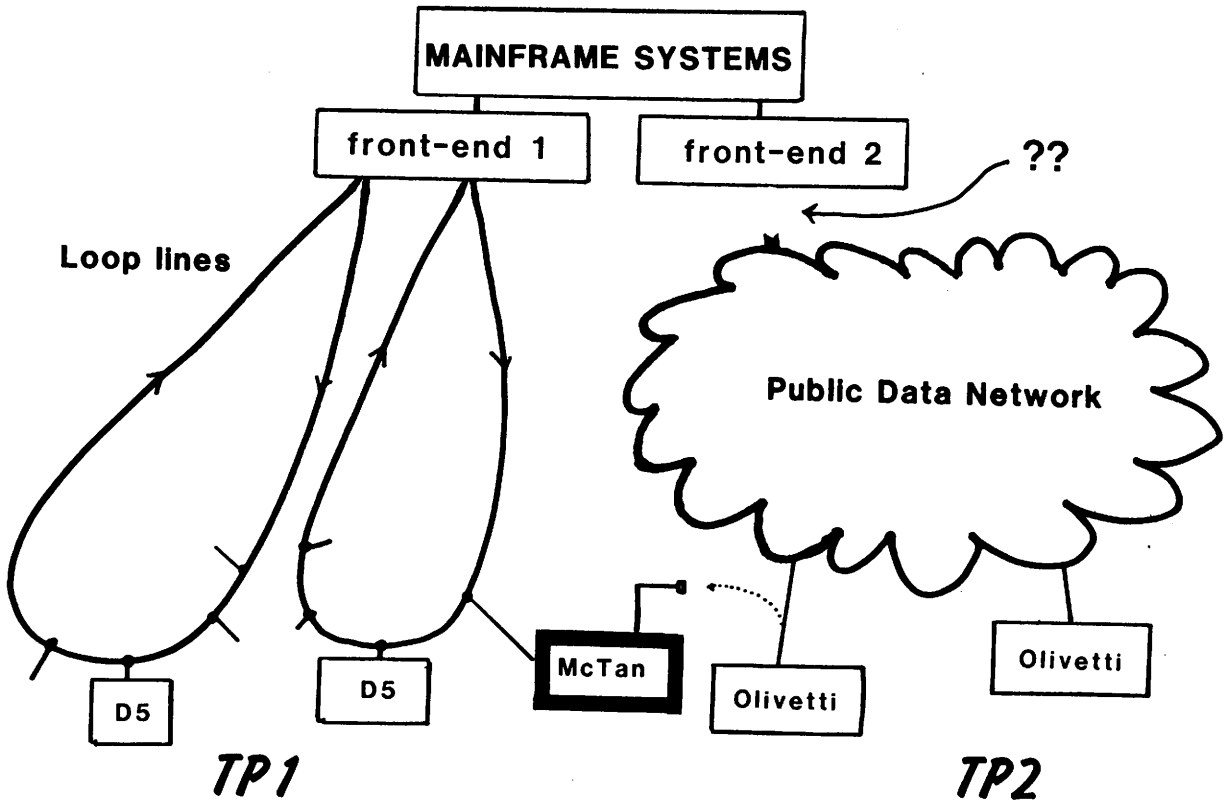


Figure 10.1: The McTan X.21 method for rescuing TP2.

To do this, we recommended the use of our own microcomputer, called McTan, and an in-house development of all software for that system. In this way, no modification would be necessary in neither Olivetti software nor hardware, and when the new front-

end system once was installed, all that would have to be done, was to pull out the cable connector from the McTan and plug it into the public network. By using in-house software development, we would benefit from our previous experience in and tools for microprocessor programming, as well as this type of project would require our detailed know-how on the TP1 loop line protocol.

10.3 McTan history

McTan is the name of SDC's own microcomputer system. The name originates from **Mc** (= scottish son of or more modern microcomputer), **T** (= my own first name Tue) and **an** (= the first name of the codesigner Anders Nielsen). For those afraid of product personification, the official meaning of the abbreviation is **M**icrocomputer **C**ommunications **T**ermination for **A**ccessing our **N**etwork (see Bertelsen, 1979)!

McTan was designed by Anders Nielsen and myself in 1978. Without any previous experience in microprocessor hardware design, we took a design challenge to design a computer based on the then new microprocessor technology, which could act as an interface towards SDC's loop line network. At that point in time, SDC had some requirements for systems to be connected to the SDC network, which could not be satisfied because of the use of a non-standard type of line protocol on the loop lines, that only Datasaab did support. On the other hand, SDC would like to maintain its concept of having only one type of central interface towards the decentralized systems in the savings banks. Hence, all such systems should connect to SDC through the loop lines.

The loop line controller in the old Datasaab system dates back to the beginning of the 70'ies. It is implemented as a huge circuit board, filled up with TTL logic. Anyhow, the protocol used is simple, although non-standard (see Bertelsen, 1979).

This means, that there did not exist (and still does not) a standard type of integrated communication controller, which would support this protocol. Hence, the usual way of interfacing serial lines to a microprocessor through LSI communication controllers could not be applied. We postulated, that instead of trying to replicate the Datasaab hardware, the entire protocol and interface could be simulated by software by such a micro.

We managed to figure out, how it could be done, and was given the needed economic support. Starting from scratch, it took one man-year to have the first system running, including hardware and software design.

The McTan is designed around the 4 MHz Z80A® microprocessor. The Z80 was chosen, because it was the optimum choice seen from our point of view at that time, even though market acceptance were still low. First we knew, that we would use the processor to simulate the loop line protocol in software. This would mean, that we would have to service every bit, which came along the loop line. Hence, fast interrupt processing would be needed, and the Z80 provides for this by having the double register set, which virtually implements two CPU's in one. Also, the interrupt structure of the Z80 allows for true vectored interrupts, which means, that no interrupt decoding has to take place at the start of an interrupt routine, as would be needed using the older Intel 8080. This again would allow for faster interrupt response. Also, the true vectored interrupt capability together with the ability to program the Z80 peripherals dynamically with new interrupt vectors, would allow an extremely simple and natural software implementation of the state diagram, that constitutes the loop line protocol (see Bertelsen, 1979).

Secondly, data communication means protocols and transactions, where data structures or individual bits within structures has to be tested, and data has to be moved fast around in memory. The Z80 was the only processor to provide for index registers and instructions to access data at a certain offset from the start of a data structure, bit manipulation instructions and block move instructions. Thirdly, we would need the better I/O instructions of the Z80, where I/O addresses can be dynamic instead of fixed instructions in the 8080. Contributing factors was the concept of separate I/O addressing, which we considered a must for reliable programming and a more natural and general symbolic instruction format, which ruled out the other contenders on that market.

One should note, that our attitude, originated in our experience from mainframe computers, was that of migrating downwards from large scale computing to microcomputing. As such, software development was not the basic problem, as we more or less knew how to handle this. Neither did we see the pure technology as a problem, as the microprocessor and corresponding circuits were off-the-shelf building blocks, which just needed to be put together to serve our needs without any need for theoretical research. The problem was how to design a system, which would give us minimal efforts to extend, maintain, repair and change, as time went on. We could well have chosen to design a single-board computer able to do the initial requirements of the system, but this would mean a fixed and static solution.

Instead, McTan was designed for change with modularity in mind. We adopted the single Eurocard format and designed our own backplane bus on a Eurocard connector, which could be implemented via a standard ribbon-cable. The idea was to divide specific functions, like processor, memory and I/O to separate boards, which could easily be changed in case of a failure. The

same was true with the power supply, which also became modularized.

The first system did contain some design flaws, mostly in the area of electrical noise decoupling, which later has been changed, but the basic design has remained unchanged until now.

After the initial design had been made and the approach was justified within SDC, three more persons joined the design team. This was Claus Ringborg, who went into further hardware design and Peter Laessoe and Ellen Randloev, who went into software design. These persons together with Anders Nielsen and myself constitute the microprocessor group within SDC today.

The accepted and commonly used name for this group is Microlab.

The McTan has been used for connecting various data processing machines to the SDC loop line network. These include data collection systems based on Olivetti DE-523, SWIFT communication systems based on the General Automation GA-16/440 minicomputer (see Bertelsen, 1979.2). Also, the McTans has been installed at various locations as stand-alone, multi-user workstations with terminals and printers connected to the loop lines.

10.4 The McTan X.21 project

Our background with the McTan system had provided us with a general know-how of, what a microcomputer-based system can do, what it cannot do, and how it eventually should be implemented.

This is the prime reason, why we quickly could overview the TP2 problem and derive to a solution. Actually, it took a few hours to figure out both that the problem could be solved and how the problem should be solved. We then used a week to work out a

written design proposal, where most of the general problems were examined (see Bertelsen, 1981.3). When this proposal were accepted, we used one month to work out a project proposal, covering all subtle aspects together with project schedule and economics (see Bertelsen, 1981.4). Finally, this was officially accepted in october 1981, more than three months after the first considerations.

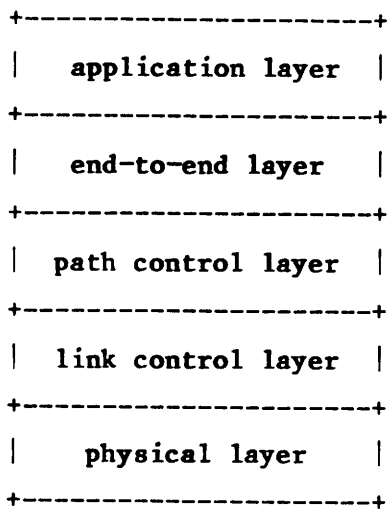
The design strategy was the following: we would design a version of the McTan, which would use the already available loop line protocol software (see Bertelsen, 1981.5) and, with a new hardware board, could simulate a DCE from the Danish Public Data Network. We had made some estimates of the system requirements, and had found that no other type of systems already on the market could satisfy these requirements. Thus we had to design both the needed hardware and software ourselves.

10.5 The TP2 communication structure

Before further discussions on the McTan X.21 design, it may be advantageous to have a look on the TP2 communication structure.

TP2 communication follows a layered approach in accordance with the ISO Open System Interconnect reference model (OSI). Not all the OSI layers are implemented in the TP2 version, called TP-monitor (see OL, 1982.2). This is primarily due to performance reasons, as the TP2 system needs small response times for normal types of transactions.

The layers are implemented as follows:



10.5.1 TP2 physical layer

The physical layer is constituted by the interface to the Danish Public Data Network (PDN), which is using the X.21 standard for call establishment. One should note, that X.21 is only defining how to connect to the network, i.e. how to dial a certain number. Also, X.21 implies the physical, electrical and logical interface standards ISO 4903, X.27 and X.24, but X.21 in itself has nothing to do with the actual data exchange. This is a common misunderstanding in SDC, where it is normally taken for granted, that a system having a X.21 interface, can be used for TP2. This is not the case.

10.5.2 TP2 link control layer

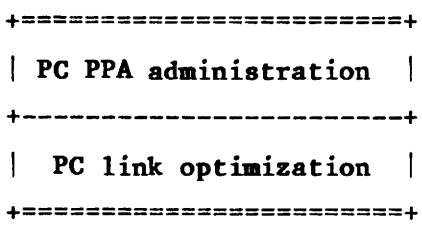
The link control layer is the one, which is really the work-horse for data exchange. The link control layer becomes active, when the physical layer (X.21) has established a connection, a circuit, between two points. Link control layer is using the HDLC asynchronous balanced protocol for data interchange. This is one place, where problems arise. The X.21 layer uses a byte-

synchronous communication, when the call is being made, whereas the HDLC layer uses a bit-synchronous communication, totally different from the byte-synchronous, but on the same wires!

10.5.3 TP2 path control layer

The path control layer is responsible for opening and maintaining a logical path between two units within the system. It operates by controlling a concept called Path Port Addresses, PPAs. When a TP2 LC wants to communicate with the central host (or with another LC), a PPA is established in each of the two systems. From now on, a logical path exist between these systems through which End-to-End messages can flow.

Actually, the path control layer is internally divided into two layers: one, that controls the administration of the PPAs and one, that maintains control of link optimization:

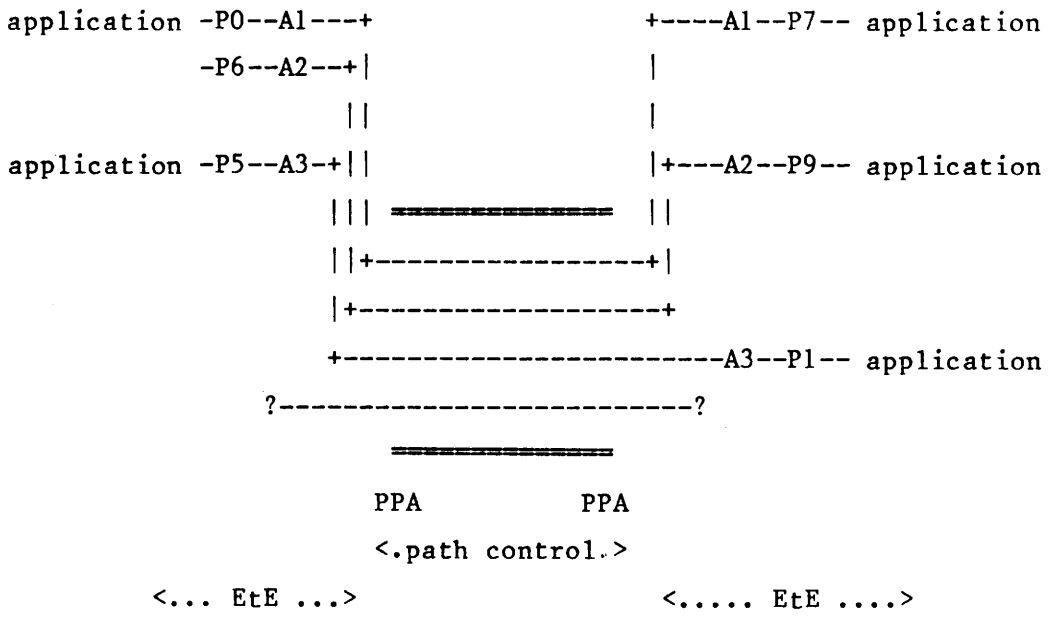


Link optimization is a feature, that takes into account the pricing policy of the public data network. In contrast to traditional data communication, a public data network link is only connected, when two systems have something to exchange. If the link becomes idle, the connection are removed and must be reconnected or redialled, when new data comes along. This is similar to the way, a normal telephone are used. Thus, a protocol must be defined, which independent of other protocols in the system, controls the usage of the link and prevent unnecessary usage of a connection.

In Denmark, the pricing policy is such, that when connecting to the public data network, the user pays a fixed fee, which corresponds to 5 seconds usage of the connection. That means, that if the connection only last for two seconds, the user will pay for five seconds, and he will pay for 5 more seconds, even if he connects again one second after. The PC link optimization uses this factor, by holding the X.21 connection at least 5 seconds after a connection has been made. 5 seconds, in the SDC system, are normally enough time to allow a transaction to be transmitted to SDC, processed at SDC and the response to be sent back to the calling system. The average response time for TP2 will be around 2 seconds. If, after 5 seconds, the line is idle, the link is disconnected. If traffic is underway on the link, it is kept open for another 5 seconds.

10.5.4 TP2 End-to-End layer

The End-to-End layer is responsible for connecting applications together. This is managed through the concept of an association between two ports on different systems. If an application wants to communicate with another application on another system, it first request the End-to-End layer to open an association with the other application. The End-to-End layer communicates with the corresponding End-to-End layer on the other system through the already established PPA, and through this way, a common agreement is made of establishing an association between the two applications, identified by a port number. As such, the identification of an application is composed of its network address (handled by the path control layer) and its port number (handled by the End-to-End layer). When this has been done, all communication between the applications take place through that association. The association concept can be considered as an individual wire between two applications, where a set of wires run through a trunk made up by the PPA:



[**Ax** = association **Px** = port number]

Figure 10.2: Association communication

The prime advantage of this concept is that the establishment of PPAs and associations ideally takes place only once a day. The rest of the day, all information needed to route data to the appropriate applications are available, thus eliminating the need for unnecessary protocol handshaking. This is also a contributing factor to shorten the transaction response time, which is one of the critical parameters in TP2.

The End-to-End layer allows two forms of data exchanges: One is the STP (SDC Transaction Protocol) and the other is the SEP (Standard Exchange Protocol). The STP is simple without any handshake at all and is used to send the smaller transactions needing fast response. STP only allows for 240 bytes of information to be sent. If more is needed, an acknowledging type of STP must be used in which the receiving application is responsible for sending a STP frame indicating an acknowledge

for each STP frame received. Alternatively, the SEP can be used. SEP allows for longer data fields, up to 2 Kbytes, which are then divided into smaller segments before they are actually transmitted. In the receiving End-to-End layer, these segments are reassembled, before the data are handled further along to the receiving application. SEP allows for a faster transmission and sequence control for long data, as windowing and delayed acknowledge are a part of SEP.

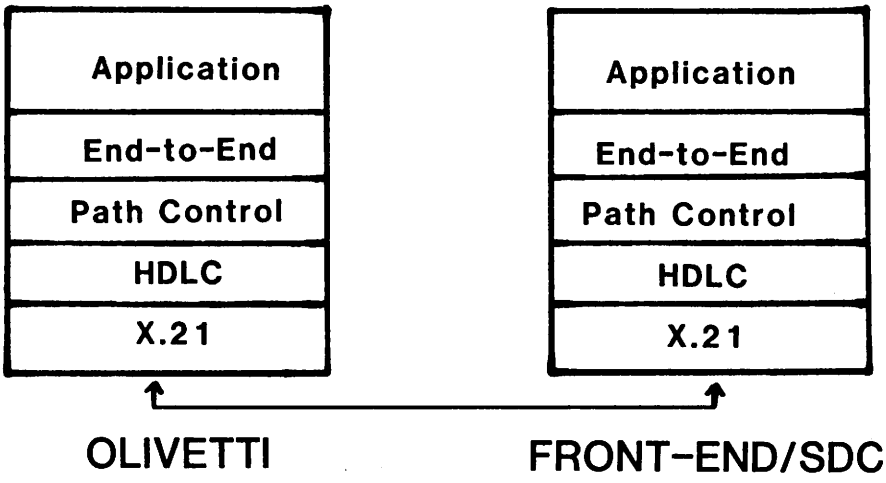


Figure 10.3: SDC layered model

10.5.5 TP2 requirements on McTan

Now these were the requirements for the McTan X.21 system:

- it should have implemented the complete physical X.21 layer down to the extent, where outside equipment could not differ between McTan and a DCE from the Public Data Network. This would include both the X.21 call establishment protocol, the ISO 4903 physical connector and pin assignments, the X.24 logical pin definitions and the X.27 electrical characteristics.

- it should have implemented the complete HDLC protocol when data phase had been established by the physical layer. This included correct management of all types of HDLC frames, correct error detection and recovery and correct management of intrinsic timers and timeout conditions.

- it should have implemented the TP-monitor Path Control layer, as were it the SDC front-end system. This included control of the PPA administration and the link optimization facilities. Also it include the management of a real-time clock facility, as the front-end path control layer synchronizes the time-of-day clock in all attached TP2 systems.

- it should have implemented the End-to-End layer of the TP-monitor, as were it the SDC front-end. Only the STP portion of the EtE-layer was needed in the expected life cycle of this solution, so SEP was not considered.

Still this was not enough. The TP2 transaction formats are incompatible with TP1, so there was not much chance of making a realistic conversion into the old-type TP1 formats. Much of this is due to the fact, that the old TP1 transactions contain only ASCII data with fields separated by a special ASCII control character, whereas TP2 transactions contains both binary and ASCII numerical data and ASCII text data with no kind of separator between different fields. If the McTan should be able to decode these transaction, it should contain a record descriptor of every transaction and answer in TP2, and this was both unrealistic and impossible to maintain.

Instead, another approach was used. First the TP1 system only allows ASCII data to be sent over the loop lines, as information are sent as 7-bit ASCII plus parity. As TP2 HDLC link communication is completely transparent, TP2 data could not be sent directly over the loop line. To solve this, a special conversion algorithm was developed (see Bertelsen, 1981.4), which made an optimal conversion of binary data into ASCII-data, where needed. The converted TP2 transaction was then packed into the data field of a standard TP1 transaction format, and this transaction was then given a special identification by McTan.

When this transaction reached the old front-end system in SDC, it would be treated as any other TP1 transaction and send farther along to SDC's own telecommunications monitor, MONRAD, on the central mainframe. A change was made in MONRAD, which enabled it to recognize the special identification placed by McTan. If so, it would then extract the TP2 data field, deconvert it through the reverse algorithm, and pass the restructured data along to the various applications. What the central applications would see, was an exact replicate of what they would have gotten, had the transaction been sent the normal way through the public data network. Thus, no changes in the central applications were needed. The answer from the application would take the same route back. It would be catched by MONRAD, binary data converted, TP1 information added and then transmitted back to McTan over the TP1 front-end.

Again, the McTan would deconvert the data in the answer back to the original look and then transmit the data back to the Olivetti system. This system would not be aware of the changes made underway, hence no modification of application or system programs were needed.

The only visible effect would be an increase in the transaction response time compared to normal TP1 response times, primarily caused by the longer transaction length and the serial communication between McTan and the Olivetti system. These effects were sought minimized through the use of the conversion algorithm and by dropping invariant header information in the answers being returned to the McTan. Also, the delays were minimized by keeping a locally high transmission speed between McTan and Olivetti.

As such, the McTan X.21 system provided a transparent, alternate route for the communication between the centralized and decentralized applications.

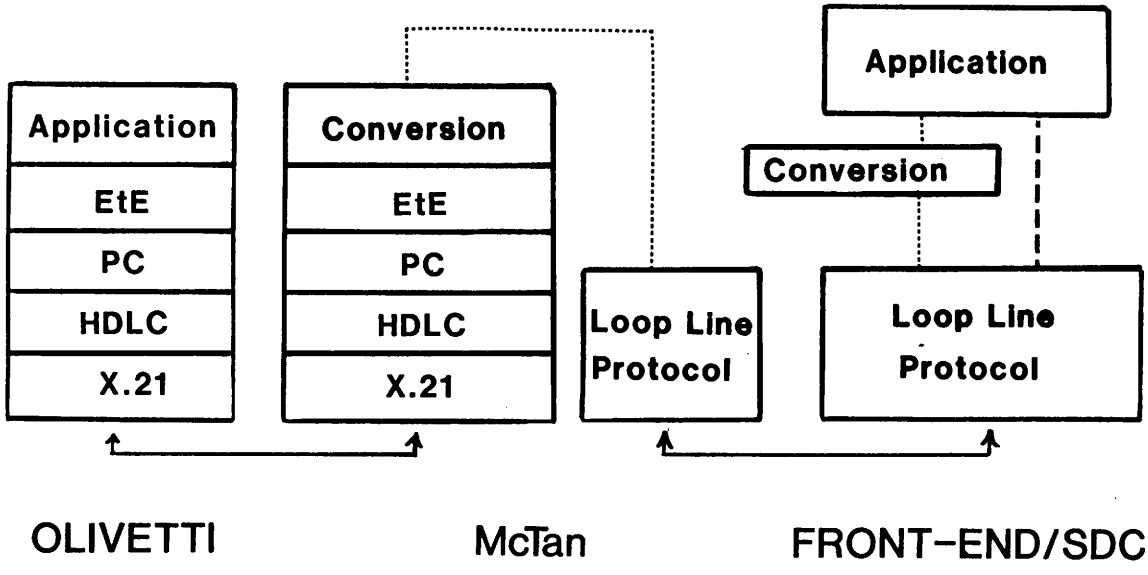


Figure 10.4: McTan participation in SDC network

10.6 Implementation details

This paragraph will discuss some of the implementation details of the McTan X.21 system and the background thereof.

10.6.1 Hardware

The McTan X.21 system would be designed around the existing McTan computer system. First of had this system proved to be reliable, and secondly, it was prepared for change. We saw the need for two new circuit boards or modules for the system:

- a X.21 interface module, which would contain the physical and electrical interfaces needed.

- a real time clock/calendar with power backup. This would be needed to maintain a valid time base for the path control layer.

10.6.2 McTan X.21 board

The X.21 board should be capable of the following: It should look like a DCE on the public data network, which included monitoring of DTE control signals and generation of bit- and byte-timing clocks for the DTE. X.21 uses synchronous communication for exchange of call information, so this had to be provided also. On the other hand, during the data phase of call, the HDLC link protocol uses bit-oriented communication over the same lines, so this would have to be implemented also.

These were no big problems, as we were already using the Zilog Z80 SIO serial communications controller chip in our previous design. The SIO is a multiprotocol chip, which can be programmed to carry out asynchronous, synchronous and bit-

oriented communication. Hence, this chip could support the data exchange functions of the line. The only problem in using such a chip is that it will be tightly connected with the overlying layers of software, which means, that the programmer must have a concise view of what is happening on a communication line and how the chip reacts to it.

However, the chip itself does not handle the X.21 signal protocol. This would have to take place by software monitoring and control of the X.21 control signals. Another problem was the X.21 byte timing signal. This is a clock pulse sent out from the DCE together with the last bit of a byte transmitted by the synchronous data line. The problem resides in the fact, that no modern systems need such a byte clock signal, because the actual protocol used on the data line ensures byte synchronization by itself. Thus a byte timing signal is not generated by any communication controller chip, nor is used, and no signals are available which tells the environment, when a specific bit is being outputted.

Generality is a virtue in system design. So even though the project was only supposed to develop a X.21 interface with DCE characteristics, we decided to design the X.21 board, so it could be used both as a DCE and a DTE. This imposed some further problems: the X.21 connector defined by ISO 4903 is such, that the same logical signals are on the same pins of the DTE and the DCE. This means, that a specific signal pin, that is being outputted at the DCE, should be treated as input by the DTE:

X.24	name	from DCE	from DTE
T	Transmit		X
R	Receive	X	
C	Control		X
I	Indication	X	
S	Signal element timing	X	
B	Byte timing	X	

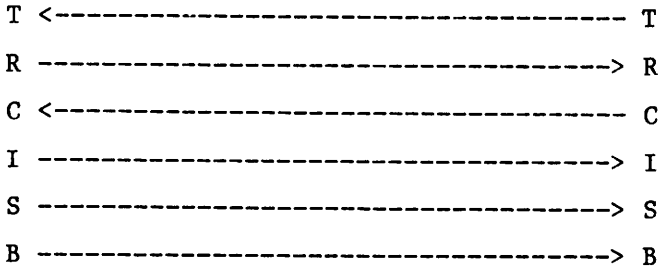


Figure 10.5: X.21 signal layouts

The result would be that the physical connector should be able to be configured differently with respect to signal paths dependent of whether a DCE or a DTE type of operation was needed. One requirement was, that any user should be unaware of this. We did not want people, who was going to install this system, to check diverse jumpers or straps, which could have a solution.

The problem was solved easily by using technology. We designed a programmable array logic chip, which managed to control the routing of the connector signals to the SIO signals on the board. If the board was to be used as a DCE, the X.21 T-signal would be connected to the SIO's receiver-input. If the board should be a DTE, the T-signal would be connected to the SIO's transmitter output. On each set of signals, both X.27 drivers and receivers were placed, where the PAL controlled whether these were active or disabled. The actual state of the PAL could be controlled by software, so now software would decide how to configure the board. Furthermore, we decided, that this unique feature would be no good if it did not also provide a self-testing feature of the board.

Thus the PAL was designed, so it could also make a loop-back connection of the X.24 signals (i.e. T connected to R, I connected to C), so the board could test itself. This loop back option could be done at the physical connector level or at the SIO level. This would provide a method to isolate failures in the X.27 drivers or receivers.

As the last option, the PAL was designed to provide for X.21 signal monitoring, i.e. all X.21 signals connected to receiver inputs at the SIO. This would enable, if needed, the system to be used as a X.21 line monitor.

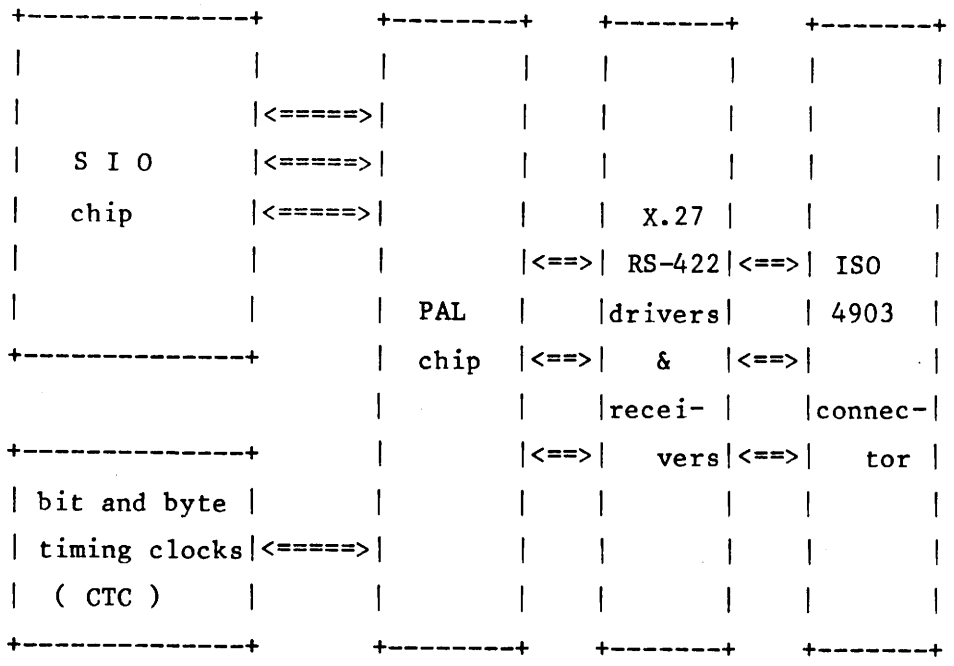


Figure 10.6: General structure of the McTan X.21 board

The time needed to design this board was one man-month from start to prototype on printed circuit board was ready. The electrical and physical design was primarily done by Claus Ringborg from Microlab.

10.6.3 McTan Real-Time Clock board

The real-time clock was solved by using a new clock/calendar chip from Motorola. This was new on the market, but was achieved through Anders Nielsen, who in the same period had been outstationed at the Collins company in Dallas to supervise the front-end development activities, so no further delays would occur. He took contact directly with Motorola in Dallas and got what they had of these chips.

The real-time clock chip was integrated with a new version of the system clock board of McTan. This board contains the circuits to generate the 4 MHz system clock, that the McTan system uses, besides power-on/reset logic. The board was also designed to include four light emitting diodes for testing and diagnostic purposes. This was needed, because the McTan X.21 system should be a self-contained "black box". All previous McTan systems, we had made, had been equipped with terminals or printers, where diagnostics could be outputted.

The basic hardware modules, we then had to design the system around, was:

- a processor module containing the Z80A processor and Z80A PIO for interface to the loop line board.
- a 32 Kbyte EPROM memory module
- a 48 Kbyte dynamic RAM module
- a dual channel RS-232C serial controller module
- the X.21 interface module
- the clock module
- a loop line interface module

These modules are sufficient to allow for McTan systems with different memory requirements and different I/O possibilities. But software is needed to get the benefit of such modules.

10.7 McTan X.21 software

Software is the major part of the McTan X.21 project. The project could not have been carried through, if the correct software tools had not been available.

Software for a communications oriented system like McTan is not like normal type of application software. The problem with data communication is, that several things happen at the same time. Transactions can be transmitted on one line, while other transactions are being received on another line. A program can prepare data to be sent, while an incoming call opens the line and starts data exchange. Timer constraints can make it necessary to perform data link handshakes at certain times. Data can be received faster, than the dedicated application can handle them.

The only way to design software for data communication systems is to use true concurrent processing, i.e. where several processes carry out different task independent of each other. By independent is meant, that the type of simulated concurrency implemented by coroutines in some systems are of no good.

10.7.1 The McTan multitasking operating system

We realized this several years ago and implemented a multitasking operating system for the McTan (see Bertelsen, 1980.3). This system allows several processes or tasks to execute concurrently, using time slicing. Also, the McTan multitasking operating system separates I/O from the OS in the sense, that individual I/O descriptors can be assigned to the process descriptors. Hence, I/O is not a part of the OS, but managed by a set of I/O drivers or I/O application services, which are build on top of the OS. Thus, individual I/O requirements for individual processes can be handled. The McTan

OS allows any process to create new processes in a dynamic fashion where convenient. As a result, it gives the programmer freedom to assign processes to individual functions, if this is found applicable.

Also, the McTan operating system imposes no specific requirements on applications development in the sense, that programmers must know the details of the operating system. As such, program modules can be made for single-user applications or multiple-user applications. One example is the loop line controller software (see Bertelsen, 1981.5), which can run in a traditionally sequential processing environment or in a multitasking environment, where several processes simultaneously access the loop line services.

Yet an unique feature of the McTan operating system is the concise view of semaphore notation and handling, where semaphores are divided into two distinct classes of passive and active semaphores. Passive semaphores are used for mutual exclusion of access to small critical regions of codes, i.e. to implement the monitor concept (see Hoare, 1974; Hansen, 1977), and they are controlled by the **lock** and **unlock** operations. The lock operation just performs a software loop, during which it tries to test-and-set the semaphore. As software regions guarded by lock/unlock operations are generally small, the waiting time spend in such a loop is minimal, compared to other approaches and hence this is an efficient way of implementing access control. Active semaphores, called so because they may alter the process state or scheduling and hence need significantly more software involvement than passive semaphores, are used for inter-process synchronization and they are controlled by the **wait** and **signal** operations (see Bertelsen, 1980.3). One should note, that the general advice, that is sometimes given, of just disabling interrupts, when executing critical regions of code, does not work in advanced microcomputer software design, as critical regions are a

concept applied at the process level which has nothing to do with the interrupt processing of the microcomputer. In fact, interrupt processing will take place simultaneously with the execution of code in critical regions at the process level.

10.7.2 The PLZ language family

Another factor is the tools for software design, including the software languages. Still today, many microprocessor-based system from many highly estimated suppliers are programmed in unreadable assembler code. Had we done that, the McTan X.21 project would never have kept its time schedule.

The McTan X.21 project is programmed in the PLZ language family. PLZ/SYS is the only member of the family, that are known to the rest of SDC, as it is being used for the TP2 application design (see Snook, 1978; Conway, 1979). PLZ/SYS is also the prime language used for the McTan system, but PLZ is more than a language.

PLZ is a language family concept, in which the language skeleton is common for all members. In fact, it is the only language available, that has these features. By skeleton is meant the language constructs used to structure a given program and the data declaration constructs. This includes what is known as procedure declarations, if-then-else statements, loops etc. What is between these constructs is the actual application dependent code, i.e. computational statements and assignment statements.

PLZ as it comes from Zilog has two members, PLZ/SYS and PLZ/ASM. Both languages are using the same control and declaration structure, but PLZ/SYS has high level statements between these constructs and PLZ/ASM has assembler statements.

As PLZ is a goto-free language, it implies, that the use of PLZ/ASM will produce goto-free assembler programs, and this is the reason for the large productivity improvement, that PLZ/ASM gives. As the control structure is the same for both languages, design of assembler programs is almost as fast as the design of high-level programs, and debugging and maintenance are dramatically minimized. In fact, more than 70 % of the McTan X.21 software has been designed error-free in the first try.

A contributing factor is the modularity of PLZ and the way it is implemented by Zilog. PLZ encourages the program design to be broken up into smaller modules, each containing its own intrinsic functions. Also, PLZ directly allows for making reentrant and recursive procedures without any need for special notification, which is a must in concurrent programming. Secondly, the procedure call interface is well documented and a master example of how a reliable programming language should look. Procedure calls are general in the sense, that there is no such oddities like subroutines or functions. Procedures can have null, one or several input parameters, and the same is true of output parameters. Input and output parameters are cleanly separated, which means, that no output parameters occur in the input parameter list. This again makes program design more reliable and more reusable. Also, it facilitates an extremely simple and reliable implementation of procedure calls with multiple parameters to the assembler level. In Microlab, we have designed a set of interface macros for the Zilog assembler, which makes it natural to interface PLZ/SYS procedures to the assembler level (see Zilog, 1979).

Although the code size and execution time penalty by using a high level language is minimal with PLZ/SYS and hence no need for special optimization seldom occurs, the language family concept however facilitates an unique approach to module optimization. As the declaration and control constructs are the

same for both PLZ/SYS and PLZ/ASM, any procedure in PLZ/SYS, which has been found time consuming, can be converted to assembler by just re-editing the computational statements, but keeping the control structure.

Most of the McTan software are written in PLZ/SYS. Interrupt routines and low level I/O-drivers are written in PLZ/ASM. As such, all McTan software (except for the old loop line controller interrupt routines dating back to 1978) is designed goto-free without a single user-written jump-instruction.

The McTan multitasking operating system is implemented as a set of PLZ procedures, which can be called from any other module.

10.7.3 McTan X.21 software design

The general concept of the McTan X.21 software design was to assign a process to each task, that could be collected into a functional unit. An example of this could be a receive task for loop line transactions or a task responsible for performing memory diagnostics.

Besides such identifiable task, which executes concurrently in the system, another type of execution must be considered. This is the interrupt type of execution, where an external event signals to the CPU, that processing is needed. Such events can be timer interrupts or serial controller interrupts when a byte has been sent or received. These interrupts must be handled fast and managed, so errors due to racing conditions in timing will not have effect.

The interrupt structure of the Z80, as mentioned before, is the ideal candidate for fast interrupt processing, especially when it is combined with the ability of the Zilog peripheral chips to identify the source of the interrupt. The SIO can as an example identify, whether the interrupt comes from a receive or

a transmit condition.

Fast interrupt handling are needed. Below is the typically peak interrupt rate in a McTan X.21 system connected to a 1200 bps loop line and a 9600 bps X.21 line:

loop line handler:	1200 interrupts per second
X.21 receive line:	1200 interrupts per second
X.21 transmit line:	1200 interrupts per second
byte time clock:	1200 interrupts per second
operating system clock:	250 interrupts per second
	5050 interrupts per second

This rate must be served by the processor, and still provide power enough for the rest of the software. The only way to do this is to program the interrupt routines in assembler. This may also be needed due to the fact, that interrupt routines typically access the chips directly through I/O instructions. However, the task of writing interrupt routines in assembler is no problem, if a suitable assembler language, like PLZ/ASM, is used.

10.7.4 McTan X.21 driver

The X.21 driver software in McTan is mainly interrupt routines. The X.21 standard is described by a state diagram, which indicates the actions to be taken, when certain events happen on the line. The interrupt routines keep control of that state and monitor its progress, until a final state has been reached. Also the interrupt routines initiate parallel running events to control the X.21 link, like sending valid information on the data line, initiating hardware timers to generate an interrupt, if timeout conditions occur, together with checking the integrity of the counterpart's usage of the link.

The interrupt relationship of the X.21 driver is complex. This is due to the fact, that the X.21 standard has some requirements, that are not handled by the serial controller chip. One example is the requirement, that any transition on the I or C signals must remain valid for at least 20 bit periods before it must be recognized by the X.21 driver. However, when one of these signals change value, it will be detected immediately by the SIO, which then generates an interrupt to the X.21 driver. The X.21 driver must then activate a hardware timer, which will interrupt 20 bit periods later. If it interrupts, the signal has remained stable, and further actions can be taken. If the signal change again before the 20 bit periods, the timer is restarted to monitor for the previous stable level.

Another tricky thing has been the synchronization of transmitted data with the byte timing signal required by X.21. This is done by having the DCE to generate the byte timing signal as a master clock. When X.21 data transmission has to be initiated, the X.21 driver will then get an interrupt from the byte timing clock, telling that now is the clock pulse active on the X.21 B-line. The interrupt routine then activates a hardware timer, which after 6 bit clock periods will interrupt again. This new interrupt will then activate the SIO chip for transmitting, and due to the internal delays within the SIO, which is dependent (but not documented) on the bit clock, the first synchronization byte, that the SIO will send, will actually come so delayed, that it will be synchronized with the free-running B-signal. As the SIO during the X.21 operation uses a byte-synchronous operation, all succeeding bytes will also be synchronized with the B-signal.

The X.21 driver is one piece of software, which has all the code needed to support both the DCE and DTE type of operation. X.21 also allows for two different types of call set-up, one for circuit switched lines (as on the public network) and one for leased lines (e.g. in a X.25 network, where X.25 is responsible for the actual routing). The McTan X.21 driver supports both types of operation, and the DCE version can, if needed, automatically sense, whether a call is being made from a circuit switched or leased line DTE.

Originally, it was suggested that the leased line type of operation should be used, because this would minimize the time needed for call setup. This agreement had furthermore been confirmed by Olivetti, as they had a type of operation of the LC, which was called "fixed line". When it came to testing, however, it turned out, that Olivetti's fixed line had nothing to do with the X.21 standard and could not operate with McTan. One reason was, that they did not control the C-signal, which is used as a handshake signal in the X.21 leased line type of operation, to control transitions from a ready state to a data transfer state and back. Furthermore, we found that the Olivetti software people had no understanding of the basic principles of X.21. When we discovered, that the upper layers of their communication software was totally dependent on which type of X.21 operation, that were used, and that HDLC and path control protocols were not identical for their circuit switched type of X.21 and their fixed line type, we stopped further discussion and used the X.21 circuit switched type of operation for the rest of the project. A layered architecture, where layers are independent, has no sense, if it only exists on paper.

10.7.5 McTan HDLC driver

Once the data transfer phase has been reached in the X.21 call protocol, the logical link layer is activated.

The logical link uses HDLC as the protocol. Hence, the SIO chip on the McTan X.21 board is reprogrammed to perform bit-oriented communication, when the X.21 driver detects the steady data transfer state.

The HDLC layer is more complex than the X.21 layer. It is controlled by a set of interrupt routines handling receiver and transmitter interrupts from the SIO. Furthermore it is managed by three concurrent processes handling receive and transmit buffers. It is accessed by a set of general routines, which can be called from any other process.

HDLC is using the concept of windowing, which is the fact, that several frames can be transmitted without waiting for acknowledge. Up to seven frames can be outstanding, and a set of timer rules are employed to ensure how acknowledge are performed within a given time frame. Together with the acknowledging comes the sequencing of frames, where a whole set of frames will be retransmitted if one of the first in a sequence are received incorrectly.

This causes the use of a set of receive and transmit buffers for HDLC operation, as the McTan system must be able to hold at least 7 frames from an Olivetti system before they are acknowledged. 8 transmit buffers are available for transmission and 12 receive buffers are used to handle received frames. This is necessary, as the HDLC sequencing control is reinitialized whenever the link is established. It is then possible for the Olivetti system to send 7 frames, get an acknowledge, close the link, call up again and start transmitting more frames, while

the previous received frames still are present in the receivers buffer pool, waiting for processing.

Applications accessing the HDLC layer are not aware of the actual physical layer. The physical layer can be X.21 circuit switched or leased line, or it can be a general RS-232C modem line. Likewise the HDLC layer itself uses the same type of protocol handshaking independent of the type of X.21 layer. Applications do however get a status code back, when they try to send a frame, which describe what kind of possible problems can be the cause of a refusal to send the frame, including link opening failures. The HDLC layer will itself try to open the X.21 link, if an application above request a frame to be transmitted. As such, applications above the HDLC layer are only concerned with their own higher level protocol and views the HDLC layer as a disposal box for messages, which have to be send.

One self-contained process is kept busy controlling the transmission of information frames and the handling of HDLC supervisory and unnumbered frames. This process also checks for link recovery conditions, timeouts, acknowledging and has been designed to solve almost any conceivable error condition.

Two processes are responsible for checking the receive buffer pool. One process checks the buffer pool for received path control frames and for other HDLC frame, that has an information content, like the HDLC XID, FRMR and TEST frames. The other process checks the buffer pool for received frames different from those above. These frames should be End-to-End frames.

The reason for having split the receive buffer handling into two processes is, that when an End-to-End frame is encountered, it is passed to the receiving part of the End-to-End layer. This layer will check the frame, and if it contains

information, that must be send to SDC, the contents are converted and a loop line transmission is requested. The loop line speed is slower than the X.21 line. Futhermore, several installations share one loop line, whereas the X.21 connection from McTan is only connected to one Olivetti system. This means, that delays can occur on the loop line before a transaction can be submitted, and this blocks the End-to-End layer, which just waits until the loop line is ready. Thus, sending a sequence of transactions to McTan may create a queue of unprocessed frames in McTan.

This is no problem, when the queue only contains End-to-End frames, which under all circumstances must be processed in sequence, but if a path control frame is added to the queue, problems can arise, since path control is responsible for the link optimization and certain recovery actions. Link optimization is done by sending a path control frame called RELREQ (Release Request). If the system receiving this frame has nothing to send, it should disconnect from the line. Otherwise it should send a RELREJ (Release Reject) frame. If these types of frames are not handled more or less immediately after they have been received, link recovery goes into action at the sending counterpart and normally an abnormal link closure will occur. This problem is solved by having a separate process to check for this.

10.7.6 McTan Path Control Layer

The path control layer is just a procedure, which is called from the receiving process checking for path control frames. There is not much code in this layer, and what it basically does is to send an appropriate answer in response to every frame received together with setting and checking for some status flags.

Link optimization, although a part of the Olivetti path control layer, is also implemented in McTan. As the path control layer described above is mainly a simple procedure called irregularly, the link optimization is handled by a separate process, whose major task is to blink the two green light emitting diodes on the McTan back panel, so an outside user can check whether the operation system is working or not. Link optimization is performed by arming a timer every time the links opens or every time a frame is being sent. If this timer runs out, a RELREQ path control frame is sent to the counterpart. Also another timer is being armed at this point. It will run out if the link has not been closed within a certain number of seconds unless a RELREJ frame has been received, and then it will force an abnormal link closure.

10.7.7 McTan End-to-End layer

The End-to-End layer is more complicated. It is divided in two parts. The receiving part is a procedure, which is called by the HDLC receive process, that checks the receive queue for End-to-End-like frames. This procedure contains all the administration of associations and protocol exchange. As such, this procedure will transmit back to Olivetti all the basic End-to-End handshake frames or status frames simulating a Collins error status if the loop line is inactive.

Besides that, a separate process handles the receiving of frames from the loop line. It determines, if a received answer is intended for the Olivetti system, deconverts it and uses the information in the association table generated by the End-to-End procedure to build a valid End-to-End response frame, which is then sent to Olivetti.

The End-to-End layer manages the mapping between the TP2 transaction addressing and the old TP1 transaction addressing. In TP2, transaction flow occurs between applications through the association concept, and addressing is performed by allocating a set of network port numbers dynamically to associations. In TP1, transaction flow is performed between the central application and a fixed terminal on the Datasaab system. The terminal address (TA) is a part of TP1 link level protocol, whereas association addresses in TP2 is a part of the end-to-end protocol, and has nothing to do with the link level.

Another problem is that TP1 only allows a limited number of terminals per installation with fixed addresses, whereas TP2 theoretically allows any number of associations to be created within a system, with no predetermined allocation of association addresses. This is solved by the McTan End-to-End layer through the use of an association table, where the TP1 TA-value is used as an index. When a request for establishing an association is received, a free entry in this table is found, and the association will then cause subsequent transactions for this association to be equipped with the corresponding index as a TA-value, when the transaction is send farther along the loop line. Likewise, when a loop line answer is received, the TA-value from the answer is used to perform a look-up in the table, from which the End-to-End address information then can be retrieved to construct the correct End-to-End answer back to Olivetti.

A sixth process is responsible for handling memory diagnostics. Concurrent with the other processes executing in the McTan, all RAM and PROM memory are checked for failure and consistency. This is done transparent for all other processes and interrupt routines.

This way of managing the flow of information within the system is indeed simple. No advanced concepts have been used. The major benefit of this approach is that it is easy to change and augment. This was needed to test the system.

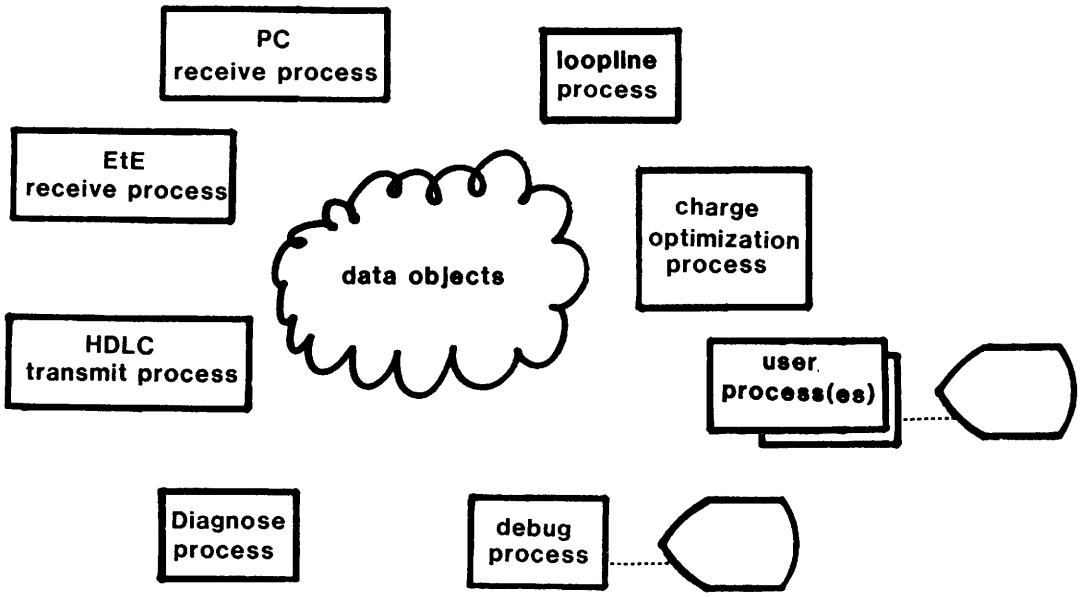


Figure 10.7: McTan X.21 process structure

10.8 McTan X.21 test

Testing had to be performed of the McTan X.21 system. During the development, we found that no useful type of test instruments was available. An expensive line monitor, bought by others in SDC, turned out to be a complete disaster, which could not handle the SDC type of communication. Also it was impossible to test against any Olivetti system, as they had trouble with their software and no chance of detecting errors in the communication.

Although not a part of the project, we decided to develop our own test instrument. In fact, this was easily done as we just added another process to the system. This process would control a terminal and a menu application program was written, which enabled us to configure and control the link directly from the screen. Futhermore, as the McTan operating system controls process oriented input/output through a set of io-descriptors, one common io-descriptor can be attached to every process, so all processes can use the screen for writing. It this way it is an easy task to add the needed statements for debugging, e.g. the processes responsible for checking the receive buffer can print a received frame on the screen.

One should note, that traditional type of debugging are of no use in a system like the McTan, where several processes interact and extensive interrupt processing goes on. Debugging cannot be performed at the language level for the same reasons, and not all types of procedures can survive, if debugging statements or breakpoints are inserted. Generally speaking, these things are forbidden in McTan design. Another point to be aware of, is that the type of communication software in McTan may contain certain areas, where software racing conditions can occur, especially at the interrupt level. These racing conditions may never occur during testing, but the software must be designed, so it has a built-in handling of all extreme situations. As such, recovery from these conditions may be untestable and one must simply rely on the faith in these recovery mechanisms.

By having a self-contained process containing the needed user interaction, two things arise. First, the user interface will (or should) automatically be more friendly, and when testing has completed, the process can just be removed or disabled. Secondly, it allows the debugging process to participate in the use of software services as an equal partner with other

processes, and this can be utilized to produce an extra load on the system or an illegal use of some procedures or data concurrent with the normal execution.

Several types of these testing facilities were implemented on the McTan systems during the design phase. They enabled us to develop and test the entire implementation of the software without the need for an Olivetti system or a special line monitor.

As such, we had free hands to select our own type of testing and diagnostic facilities. As the McTan is modular, and the PLZ language allows for modular software design, it was no problem to connect a sufficient advanced display terminal to the system, and use the capabilities of such a terminal to make a user-friendly interface to the software under test. This interface was designed to provide menu selection and intelligent prompting for setting up the various configurations of the software and keeping trace of the communication. Such facilities were unknown on the Olivetti systems. Another important design issue was the capability of generating a high load on the software by repeatedly sending HDLC frames or repeatedly making calls on the X.21 line. Another facility would permit us to impact different types of link errors on the network line.

These facilities proved vital during the official acceptance test with the Olivetti equipment. This acceptance phase of the project turned out to need much more time than anticipated. Several reasons caused this delay:

- The first reason was the Olivetti organization, where the people responsible for the different layers had no fundamental knowledge of other layers in the system. This made it necessary to divide the acceptance phase in several independent tests. When problems occurred in

the HDLC layer during a path control test, we were inhibited in discussing the problems as the people responsible for the path control layer knew nothing about HDLC. Actually, this phenomenon may be natural in a large organization like Olivetti, where several project groups must be managed in a more strict and defined manner than the one used in Microlab, but it does have disadvantages.

In this way, we found, that they actually broke protocol conventions in some cases. One example was when the Olivetti system received the path control command CLOSE_PPA. Documentation told us, that this should generate a CLOSE_PPA response. What the Olivetti system did, was to close down the X.21 link immediately after the reception of the frame with this command, without any concern for the HDLC layer, which never would acknowledge the frame. This would then generate an error condition in McTan, which would initiate the proper recovery and try to open the link again for retransmitting the frame. This immediate re-establishment of the call could then make the Olivetti system go down.

- Another reason was the Olivetti tools to carry out the test. These tools were slow and whenever frames were received from McTan, the Olivetti test people had to read hexadecimal dumps to figure out what had been received, which caused long periods of waiting. On the McTan system, this information was immediately visible in readable form on the display.

- Finally, the test made with the Olivetti system were static, due to the slow interface to the Olivetti system. This made it impossible to test any dynamic errors, which might occur when the McTan system was presented with a high load or frequent X.21 calls. We had to do such tests ourselves between two McTan systems.

One example of the effect of this static type of acceptance testing in TP2 has been the link problems present in the TP2 system this spring. Here, systems have gone down due to the fact, that they cannot recover from the situation, where both the Collins system and the Olivetti system tries to establish a call at the same time. This will happen, if a transaction takes longer than 5 seconds to be processed within the SDC mainframes, so the link is closed. If a new transaction is generated from another teller on the Olivetti system at the same time as the front-end system is ready to send the delayed answer, the collision condition may occur, and it actually does this in some per thousands of the calls.

This collision phenomenon is properly handled by the McTan X.21 driver, because it was spotted in our own dynamic test sequences.

Other minor problems were imposed on the McTan software, as it turned out that the TP-monitor documentation was not consistent with the actual Olivetti implementation, and where these problems sometimes were first discovered during the actual acceptance tests. Further changes were also made in the path control and end-to-end layer during the development phase, but it was easy to adapt to these.

One general and interesting effect can be noticed from the testing phase of the X.21 and link control software. From the point in time, where the software could pass the static tests, to the point in time, where the software would pass all our own conceivable dynamic tests, the size of the driver software nearly doubled. The different forms, in which a failure needing special recovery can occur, is many, especially when several processes are cooperating to control the line. However, the development in this increase in software is well justified in the absence of troubles, that might later occur.

10.9 McTan X.21 production

The first prototype of the McTan X.21 system was made in Microlab, SDC. During the autumn 1981, we discussed various ways to get the final version of the system produced. The problem was, that we could not estimate the need for systems, as these were dependent on the success of the completion of the Collins front-end. In the best case we would need about 20 systems and in the worst case maybe 500 or even more.

We decided to be responsible for the production ourselves, in the sense that we managed the purchasing and procurement of components in SDC and had the assembly of systems done by the independent electronic assembly company, Dansk Elektronik Montage. Integrated circuits was before assembly screened and tested at Elektronikcentralen.

This concept proved to work without many problems. The final system test was made in SDC with the built-in diagnostics of the X.21 software. We produced a prototype batch of 10 systems, which was needed for further development of the software and for acceptance testing. Our first real batch was of 20 systems, most of which were quickly taken into usage in SDC. Early

spring 1982, we decided to initiate the next batch of 40 systems.

As such, 70 systems were produced for this project. Luckily enough (or sadly), the Olivetti share of the TP2 project was delayed due to several reasons, so when the first part of the series installation was started, the Collins front-end system was ready to take over.

Nevertheless, the McTan X.21 systems had done their duty by connecting all of the internal TP2 systems within SDC to the host, and by connecting the prototype installations in the branch offices to the loop line network.

10.10 Project manpower

It is rather difficult to separate the different activities in a project like this, as many activities has been carried out in parallel by the same or few persons. The following figures should anyhow give a proper estimate in the design time for this project:

first analysis	-	1 week
preinvestigation	-	2.5 manmonths
X.21 controller board	-	1 manmonth
X.21 software driver	-	1 manmonth
HDLC software driver	-	1 manmonth
Path Control Layer	-	0.5 manmonth
End-to-End layer	-	2 manmonth
Loop line software	-	1.5 manmonth
Acceptance tests	-	2 manmonths
Production preparation	-	3 manmonths

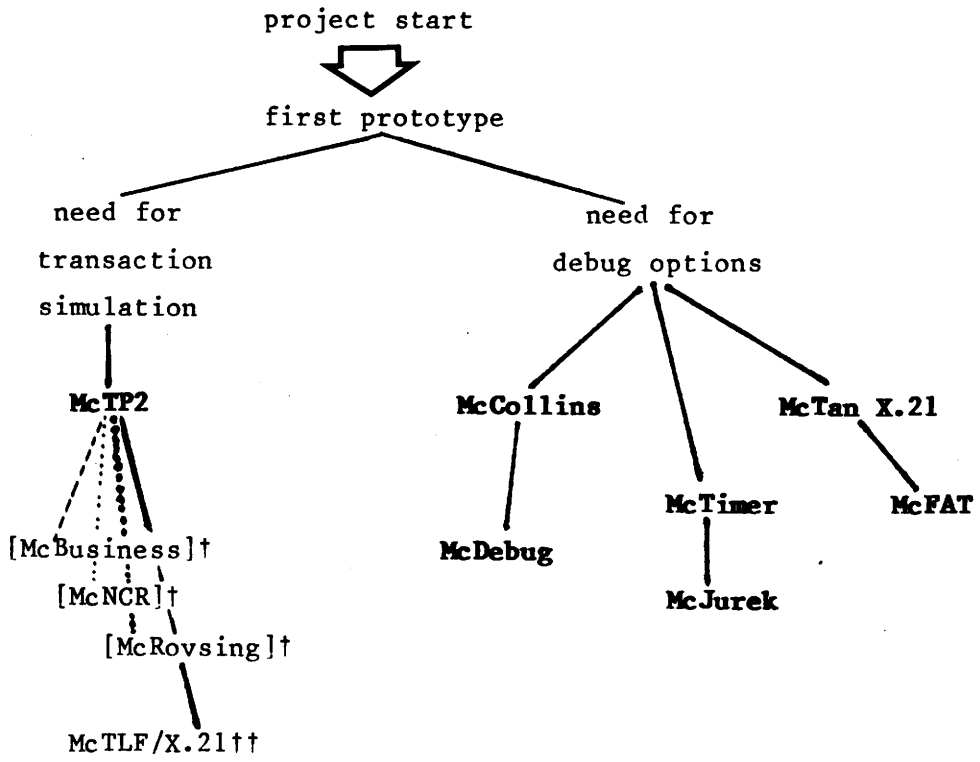
These figures show, that the amount of work needed is not staggering and can easily be carried out within a company like SDC or even smaller companies. The figures does however not show the entire truth, as they do not include the already made software for other purposes, like the loop line driver software and the multitasking operating system, as well as the figures are also a reflection of our know-how within 8-bit microprocessor applications, PLZ/SYS and PLZ/ASM interaction, concurrent programming in PLZ and our general overview on SDC communication. Using a non-modular language or just pure assembler instead of PLZ/ASM might well cause these figures to multiply by a factor of 4 or more.

It should be noted, that several other activities were done, like project administration, project activity scheduling, evaluation and design of test facilities, installation planning, which also involved other resources in SDC at various points in time, but which can be difficult to assess.

10.11 McTan X.21 derived products

The development of the McTan X.21 system has caused some derived products to be designed within SDC. These have been caused by differents needs for interfaces and needs for debugging.

The product history is as follows:



(†: design proposals)
(††: currently in design for SDC loop line network)

The McTan X.21 denotes the system primarily used within SDC and in the pilot-installations in the branch offices in the savings banks. The other product names used in SDC are described below:

McTP2 is a system with a X.21 DTE interface. It is equipped with a screen and printer and has built-in application software, that simulates all standard STP transactions from the Olivetti TP2 system. The system has been used by the design group in SDC, which is responsible for implementing the centralized TP2 applications. By having McTP2, they were able to test out central applications before the corresponding decentralized TP2 applications on the Olivetti system had been tested. It also showed, that a small computer actually can function as a workstation directly coupled to the SDC network.

McCollins is an enhanced X.21 tester. It was designed, when people from Collins in U.S.A. saw the initial McTan testing facilities for X.21. The problem in U.S.A. is, that a X.21 network does not exist, and no testing facilities are available. Two McCollins systems were sent to Dallas and contributed to the non-delayed delivery of the Collins system. When the first system arrived in Dallas and was powered on, it found 17 hitherto undetected errors the first day (see Hansen, 1982).

McFAT is a system much like the standard McTan X.21, but equipped with a matrix printer for debugging of sent and received frames. It has been used as a part of the acceptance test of TP2 software from Olivetti.

McTimer is a X.21 line monitor system, capable of performing response time measurements. It is connected as a monitor to a X.21 line between an Olivetti system and SDC, and has an monitoring interface for a keyboard and printer attached to a TC. In this way, time can be measured from a certain keystroke on the Olivetti keyboard, to the transaction occurs on the X.21 line, to the answer coming back and until the answer is printed on the TC journal printer.

The McTimer system has a timer resolution of 1 millisecond, and has been used to verify if response times were in accordance with the requirements put in the contract between Olivetti and SDC. It has also established a concise definition of events, on which the response time measurements are based.

McJurek is a device similar to McTimer, but designed to perform repeatedly simple response time measurements on the X.21/HDLC line without operator involvement.

McDebug is a new application, where the X.21 interface is used in DTE mode. The McDebug system contains a list of network DCE-numbers and can be instructed to perform a set of automatic tests on these numbers. The prime need is to test all the ports from the network to the Collins system, as the Collins system has no way of easily detecting whether a port is in operation or not. McDebug can perform these tests automatically without operator intervention and report statistics and error types on a printer.

McBusiness is a design concept for a business or customer computer with access to the SDC network. This computer will use a standard operating system, freeing SDC from any responsibility of providing application programs for the business customer. Also, this design concept is an example of the type of systems, that eventually may connect to the ASA nation-wide level.

McNCR is a design proposal for connecting NCR automatic teller machines to the SDC network.

McRovsing is a design proposal, which has been discussed with the Danish PTT. It should act as a concentrator, which would concentrate traffic from the proposed Dankort-terminals onto the X.21 public data network.

McTLF is a distributed processor system currently being designed for use on the loop lines, but which will be transferred on the public data network, when the TP2 installation is complete and traffic from this system is known.

McTLF provides a dialog-oriented application system, which communicates with printer-telephones placed in small businesses. Also it will support access from normal terminals, standard microcomputer systems as well as Teledata televisions. Through the dialog system, the user request information from the central system services at SDC.

10.12 Conclusions

The McTan X.21 project was an example of the importance of having in-house know-how and experience on microprocessor technology in a company like SDC. The prime objectives of SDC are not to design all hardware ourselves, but the type of know-how accumulated within Microlab, which made the X.21 project possible, cannot be acquired from pure theory and paper reading. Know-how in the microprocessor field must be kept valid, as the changes in technology are going on. By keeping this know-how up-to-date, SDC will be a qualified design partner for outside vendors, and it will enable SDC to implement solutions, which may not be in the standard product catalog of the vendors, but which will ensure the internal SDC standards to be followed. The organizationally concept of a microcomputer laboratory like Microlab is a brilliant idea for future research and developments in the large process of enhancing the entire SDC spectrum of know-how.

This page intentionally left blank

APPENDICES

11.1 Supplements

The following supplements are enclosed with this report:

- TUBUS Specification; EF-57 report
- The Z8000 TUBUS design; EF-57 report
- McTan X.21 projekt rapport

  - Udredning vedr. etablering af en datatransmissionsvej mellem Olivetti TP2 udstyr og SDC's centrale applikationer via de eksistende ringnet.

- McTimer technical manual

11.2 List of P896 participants

- as of november 25, 1981.

see the following pages

November 25, 1981

P896 MAILING LIST

Please check your address and telephone number for accuracy.
(We lack telephone numbers for a few of you.)

Mailing List	Telephone	Country
Andrew Allison Chairman, P896 Committee 27360 Natoma Road Los Altos Hills, CA 94022	(415) 941-6065	U. S.
Prof. J-D Nicoud EPF-DE-LPRL Bellerive 16, CH-1007 Lausanne SWITZERLAND	(41)(21) 47-26-42	Switzerland
Leon Adams 12403 Spring Grove Houston, TX 77099	(713) 778-5724	U. S.
Jean C. Ballegeer Thompson CSF/LCR Domaine de Corberville 91401 Orsay, FRANCE	(33)(6) 941-82-40	France
Tue Bertelsen Sparekassernes Datacenter Borup Vang 1, DC-2750 Ballerup, DENMARK	(45)(2) 657-11	Denmark
N. Bock Forschung Seigersdurf Lenaugasse 10, A-1080 Wien, AUSTRIA	(43)(225) 480-22-59	Austria
Paul Borrill Mullard Space Science Lab Holmbury St. Mary Gb-Dorking RH5 6NT, ENGLAND	(44)(30) 670-292	England
Keith Britton Blastmasters, Inc. 325 Beverly Ave. San Leandro, CA 94577	(415) 635-5144	U. S.
Bob Davis Summit Computer Systems 22685 Summit Road Los Gatos, CA 95030	(408) 353-2706	U. S.

Dante Delcorso Politecnico TET, Duca Abruzzi 24, 10129 Torino, ITALY	(39)(11) 51-64-54	Italy
Jim Flournoy 548 N. Darlington St. Rosemead, CA 91770	(213) 280-4365	U. S.
J. U. Gander Borer Electronics CH-4500 Solothurn, SWITZERLAND		Switzerland
Dana Grubb Room A216, Bldg. 225 National Bureau of Standards Washington, DC 20234	(301) 921-3427	U. S.
David Gustavson SLAC, Bin 88 P. O. Box 4349 Stanford, CA 94305	(415) 854-3300 x2803	U. S.
Dick Hodgman Intel Corp 3065 Bowers Ave. Santa Clara, CA 95051	(408) 987-7100	U. S.
Steve Kerman 3131 Homestead Road #9D Santa Clara, CA 95051		
Hubert Kirrmann BBC, Forschungszentrum 5405 Baden-Daettwil SWITZERLAND	(41)(56) 84-80-71 (41)(56) 83-19-03	Switzerland (home)
John Levy 651 Tennyson Ave. Palo Alto, CA 94301	(408) 973-3406	U. S.
Richard Lyman 10716 Stevens Canyon, #2 Cupertino, CA 95014	(408) 257-8397	U. S.
Hal Massey 1270 Coronado Drive #10 Sunnyvale, CA 94086		
Jean-Claude Mathon Eurotechnique, Box 2 BP2-Z1 Peynier Rousset 13790 Rousset, FRANCE	(33)(42) 23-98-01	France

Andy McMillan Xycom 750 North Maple Road Saline, MI 48176	(313) 429-4971	U. S.
Wolfgang Mahr Phonixweg 8 8408 Winterthur SWITZERLAND		Switzerland
brian McGann Assoc. Computer Consultants 210 Highland Drive Aptos, CA 95003	(408) 425-0937	U. S.
Savas Mirci Rockwell International Box 3669, MS 022/BC-22 Anaheim, CA 92803	(714) 632-1209	U. S.
Jonn Morgan Morgan Data Systems Box 3008 Ft. Walton Beach, FL 32548	(904) 863-2124	U. S.
Prof. R. Patzelt Technische Universitat Wien, Gusshausstr. 25, A1040 Wien, AUSTRIA	(43) (222) 65-76-41	Austria
Mike Rothan Post Office Telecommunications TD 913, Room 613 207 Old Street London ECIV 9PS, ENGLAND	(44) 01739-3464 x7631	England
Jim Spackman Geophysical Service Inc. Box 225621, M/S 3977 Dallas, TX 75265		U. S.
Mike Smolin, Co-Chm CS MSC Synertek, Inc. Box 552, MS 39 Santa Clara, CA 95052	(408) 988-5777	U. S.
David Stevenson Zilog Corp. 10460 Bubb Road Cupertino, CA 95014	(408) 446-4666	U. S.
Francis Stevenson Kongsberg Vaapenfabrikk Dept. of Computer System Dev. Postbox 25, N-3601 Kongsberg, NORWAY		Norway

Bob Stewart 1658 Belvoir Drive Los Altos, CA 94022	(415) 941-8099	U. S.
John Theus Tektronix Inc. Box 500 MS92-803 Beaverton, OR 97077		
Gary Wood 2467 Wosswood Lane Santa Clara, CA 95051	(408) 727-6289	U. S.
Friedrich Zahorka, BBC Dep. ESL-23, 5300 Turgi SWITZERLAND	(41)(56) 29-94-70	Switzerland

PDP is a trademark of Digital Equipment Corp.

Pascal/MT+ is a trademark of Digital Research, Inc.

Proofreader is a registered trademark of Aspen Software Company

SASI is a trademark of Shugart Associates

UCSD p-System is a trademark of The Regents of the University of California

UNET is a trademark of 3Com Corp.

UNIX is a trademark of Bell Laboratories

VAX is a trademark of Digital Equipment Corp.

VERSAbus, VMEbus are trademarks of Motorola, Inc.

WordStar is a trademark of MicroPro International Corp.

Xerox is a registered trademark of Xerox Corporation

Z80 is a registered trademark of Zilog, Inc.

Z8000, Z80000, ZBI are trademarks of Zilog, Inc.

iAPX432, iRMX are trademarks of Intel Corporation

11.4 McTan price list

The price list which come on the following pages are the official sales prices for sale of McTan products to the savings banks.

PRISLISTE
for
McTan Microcomputer System

ID	Beskrivelse	PRIS
CPU	Processorkort med 4 MHz Z80A CPU, Z80A PIO, clock driver, buffers og interrupt logik.	1995,-
CLOCK	4 MHz System clock kort, forsynet med CMOS real-time clock/calendar/RAM kreds med batteri-backup samt diagnostic display option.	2180,-
SIO	Serielt datakommunikationsmodul med to fuld-duplex kanaler, asynkron/synkron/-bitorienteret kommunikation. Indeholder Z80A SIO/0, Z80A CTC. To RS-232-C kompatible udgange.	3575,-
X21	Datakommunikationsmodul til det offentlige datanet. Opfylder CCITT X.21 standard. Elektrisk kompatibel med X.27. Mekanisk tilslutning i overensstemmelse med ISO 4903. Kortet kan programmæssigt styres til at se ud som DTE, DCE eller X.21 dual channel monitor.	3550,-
MODEM I	Modem interface modul til ringnetsmodem.	1485,-
MODEM II	Intelligent Modem interface modul til ringnetsmodem. Tillader seriekobling af flere McTan´s eller D5´er på samme ringnetsmodem.	2180,-
EPROM	32 Kbyte UV-EPROM kort. Indeholder plads til 8 prommer af typen 2732, 2532, 2716, 2516. Wait-cycle logik er integreret på kortet, så prommer med lang accesstid kan bruges.	2725,-
STATIC-4	Hukommelseskort med 4 Kbyte statisk RAM.	2095,-
DRAM-16	Hukommelseskort med 16 Kbyte dynamisk RAM.	2725,-

DRAM-32	Hukommelseskort med 32 Kbyte dynamisk RAM.	3225,-
DRAM-48	Hukommelseskort med 48 Kbyte dynamisk RAM.	3725,-
CMOS-16	Hukommelseskort med 16 Kbyte CMOS statisk RAM. Indeholder batteri-backup, der kan gemme informationen ved strømsvigt i 8 dage.	3905,-
INTLOC	Interrupt logik modul, der forøger eksekveringstiden for Z80 RETI instruktioner, således at Daisy Chain ripple problemer i systermer med mange I/O-kort minimeres.	985,-
POWER-5	Switching Power Supply modul, 5 Volt 5 Ampere. Indeholder specielt netfilter til dæmpning af støj på lysnettet, sikkerhedsindkapsling, integreret sikringsholder, kapacitet for hold-up ved strømudfald, 12 VAC trafo til ventilator. Isolationsspænding 2000 V. Kan køre på 110-240 V, 50/60 Hz.	4670,-
KABINET	Aluminiumskabinet med sort svøb. Indeholder system backplane, lysdiode for 5 Volt power, samt ventilator.	3695,-

Prislisten er gældende fra 13. august 1982.
Priser er DKr. ekskusiv moms.

Der tages forbehold for ændringer i priser og specifikationer.

Microlab 82.08.13
THB/AN

B I B L I O G R A P H Y

[AMD, 1980]:

Advanced Micro Devices: The AmZ8000 BLOC-BUS Interface; Confidential AMD material covered by non-disclosure agreement, 1980;

[Abraham, 1983]:

Robert Abraham & Ron Munro: Microfloppies battle for preeminence; Computer Design, January 1983, p.119-126; ISSN 0010-4566

[Adams, 1981]:

Leon Adams: IEEE-P896 electrical considerations; P896 document, July 8, 1981;

[Akkoyunlu, 1974]:

Eralp Akkoyunlu, Arthur Bernstein, Richard Schantz: Interprocess communications facilities for network operating systems; Computer, Vol. 7, No. 6, June 1974; ISSN 0018-9162

[Alexandridris, 1980]:

N.A. Alexandridris: Microprocessor systems - architecture and engineering; Infotech State of the Art Report, Microelectronics, series 8, no. 2, p.3-41; ISBN 8553-9630-X

[Allan, 1983]:

Roger Allan: Network developments taking different routes; Electronic Design, June 9, 1983, p.49-50; ISSN 0013-4872

[Allison, 1981]:

Andrew A. Allison: Status report on the P896 backplane bus; IEEE Micro, February 1981, p.67-82; ISSN 0272-1732

[Alpert, 1983]:

Don Alpert, Dean Carberry, Mike Yamamura, Ying Chow & Phil Mak: 32-bit processor chip integrates major system functions; Electronics, July 14, 1983, p.113-119; ISSN 0013-5070

[Anderson, 1981]:

D.A. Anderson: Operating systems; Computer, June 1981, p.69-82; ISSN 0018-9162

[Anderson, 1982]:

Gordon E. Anderson & Kenneth C. Shumate: Selecting a programming language, compiler and support enviroment: method & exp; Computer, August 1982, p.29-36; ISSN 0018-9162

[Andrews, 1983]:

Gregory R. Andrews & Fred B. Schneider: Concepts and notations for concurrent programming; Computing Surveys, Vol. 15, No.1, 1983, p.3-43; ISSN 0010-4892

[Announce, 1983]:

<unknown>: Software for Ethernet LANs reaches transport level; Computer Design, January 1983, p. 162; ISSN-0010-4566

[Announce, 1983.2]:

<unknown>: Hardware interface links RS-232 ports to Ethernet networks; Computer Design, February 1983, p.54-55; ISSN 0010-4566

[Arora, 1982]:

R.K. Arora & N.K. Shama: On the design of a distributed operating system using a HL distributed P.L.; Microprocessing and microprogramming, Vol 10, no. 4, 1982, p.247-254; ISSN 0165-6074

[Arst, 1982]:

Philip L. Arst, David T. Yeh, Charles Gopen & Robert Galin: 2-chip controller set drives down Ethernet hookup costs; Data Communications, October 1982, p.163-172; ISSN 0363-6399

[Aseo, 1983]:

Joseph Aseo: Ethernet software handles file transfer, remote program execution; Computer Design, February 1983, p.39-40; ISSN 0010-4566

[Ashkenazi, 1981]:

David Ashkenazi: Hardware comes to the aid of modular high-level languages; Electronics, April 21, 1981, p.175-177; ISSN 0013-5070

[Bachman, 1982]:

Charles W. Bachman & Ronald G. Ross: Toward a more complete reference model of computer-based information systems; Computer Networks, No. 6, 1982, p.331-343; ISSN 0376-5075

[Bal, 1982]:

Subhash Bal et al.: The NS16000 family - advances in architecture and hardware; Computer, June 1982, p.58-66; ISSN 0018-9162

[Banning, 1979]:

John Banning: Z-bus and peripheral support packages tie distributed computer systems together; Electronic Design, Vol. 27, No. 24, 1979; ISSN 0013-4872

[Barker, 1983]:

Barry Barker: Interface Intelligence; Systems International, March 1983, p.21-22; ISSN 0309-1171

[Barnes, 1983]:

George H. Barnes & Stephen F. Lundstrom: Design & validation of a connection network for many-processor multipr. systems; Computer, December 1981, p.31-41; ISSN 0018-9162

[Basart, 1982]:

Ed Basart, David Folger & Bill Shellooe: Pipelining and new OS boost mini to 8 MIPS; Mini-Micro Systems, September 1982, p.258-272; ISSN-0364-9342

[Bass, 1980]:

Charlie Bass: A comparison of networks vs. trad. comm. in distributed systems architecture; Proceedings of the ACM Pacific-80 conference, 1980, p.68-69;

[Bass, 1980.2]:

Charlie Bass, Joseph S. Kennedy & John M. Davidson: Local network gives new flexibility to distributed processing; Electronics, September 25, 1980, p.114-122; ISSN 0013-5070

[Beach, 1982]:

Robert Beach et al: System-level functions enchance controller IC; Electronics, October 6, 1982, p.95-97; ISSN 0013-5070

[Beauvais, 1980]:

Linda Beauvais & Dale H. Grit: A protocol for a ring network of heterogeneous computers; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.77-85; IEEE 80CH1542-0

[Becker, 1980]:

Hal B. Becker: Information management strategies in the 1980-s; Proceedings of the ACM Pacific-80 conference, 1980;

[Bell, 1983]:

Alan E. Bell: Optical data storage technology status and prospects; Computer Design, January 1983, p.133-146; ISSN 0010-4566

[Bender, 1980]:

Linda Bender: ZBI: a system bus for the Z8000; Mini-Micro Systems, June 1980, p.67-75;

[Bender, 1982]:

Michael Bender: Distributed databases: Needs and solutions; Mini-Micro Systems, October 1982, p.229-235; ISSN 0364-9342

[Beresford, 1983]:

Roderic Beresford & J. Robert Lineback: Opening salvos launched in 256-K battle; Electronics, June 30, 1983, p.107-108; ISSN 0013-5070

[Bertelsen, 1979]:

Tue Bertelsen: Micros expand the lifetime of old protocols; Proceedings of the MIMI'79 8th international symposium, ACTA Press, 1979, p.13-19; ISBN 3-7153 0000 6

[Bertelsen, 1979.2]:

Tue Bertelsen & Anders Nielsen: Implementing distributed processes in centralized systems; Proceedings of the MIMI'79 8th international symposium, ACTA Press, 1979, p.94-98; ISBN 3-7153 0000 6

[Bertelsen, 1979.3]:

Tue Bertelsen & Anders Nielsen: Implementing distributed processes in centralized systems; Kiver Communications S.A.: Proceedings of the IMMM-79 conference, June 19-21, 1979, p.301-306;

[Bertelsen, 1980]:

Tue Bertelsen: TUBUS Specification; EF-57 report, June 1980;

[Bertelsen, 1980.2]:

Tue Bertelsen: The Z8000 TUBUS design; EF-57 report, May 1980;

[Bertelsen, 1980.3]:

Tue Bertelsen: SDC multitask scheduling facilities - User's guide; Microlab, SDC, Rev. 1.0, 1980;

[Bertelsen, 1980.4]:

Tue Bertelsen & Anders Nielsen: Draft proposal for the communications protocol between Olivetti UPLC and McTan; Microlab, SDC, May 1, 1980; SDC internal document

[Bertelsen, 1980.5]:

Tue Bertelsen: SDC's views on computer technology and its impact on future branchoffice system; Paper presented at the OL/SDC future product committee meeting, Firenze, 1980;

[Bertelsen, 1981]:

Tue Bertelsen: McBusiness - Systemarkitekturen for Sparekassernes Erhvervsterminal; Foreloebig design rapport, SDC, december 1981;

[Bertelsen, 1981.2]:

Tue Bertelsen: Report of the EF-57 study tour covering visits to major companies in USA; EF-57 project report, January 1981;

[Bertelsen, 1981.3]:

Tue Bertelsen & Claus Ringborg: McTan X.21 PDN/Loop line interface - Notat vedr. TP2 access til SDC online-sys.; SDC, Juli 1981;

[Bertelsen, 1981.4]:

Tue Bertelsen et al.: McTan X.21 projekt - Udredning vedr. etablering af en datatransmissionsvej ....; SDC, 26 august 1981;

[Bertelsen, 1981.5]:

Tue Bertelsen: McTan loop interface utility. Release 5.5. User's guide; Microlab, SDC, January, 1981;

[Bertelsen, 1981.6]:

Tue Bertelsen: TP2 og McTan; SDC informerer, No. 6, November 1981;

[Bertelsen, 1982]:

Tue Bertelsen: Naar Data skal kommunikeres - og kommunikation skal programmeres!; Proceedings of the Programmering '82 conference, 1982;

[Bertelsen, 1982.2]:

Tue Bertelsen: McTan er i luften; SDC informerer, Juni 1982, p.9-10;

[Bertelsen, 1982.3]:

Tue Bertelsen: Preliminary proposal for TP2 user data formatting; Microlab, SDC, March 23, 1982;

[Beuchat, 1981]:

Rene Beuchat & Rene Sommer: A local network interface on P896; IEEE P896 Design Course Documentation, June 28, 1981;

[Bigelow, 1983]:

Bert Bigelow: Dual processor tackles complex control tasks; Electronic Design, June 9, 1983, p.161-166; ISSN 0013-4872

[Black, 1983]:

George Black: CP/M is now tuned in to C; Computer Weekly, April 28, 1983, p.5;

[Blackett, 1983]:

R. Kent Blackett: UNIX-based system runs real-time applications; Mini-Micro Systems, May 1983, p.249-255; ISSN 0364-9342

[Blasgen, 1979]:

Mike Blasgen et al: The Convoy phenomenon; IBM San Jose Research Laboratory, 1979;

[Bliss, 1983]:

John Bliss & Dave Stevenson: Effectively link microcomputers with fiber optics; Computer Design, January 1983, p.75-80; ISSN 0010-4566

[Bloom, 1979]:

Toby Bloom: Evaluating synchronization mechanisms; Proceedings of the seventh symposium on operating systems principles, ACM No. 534790; 1979, p.24-32; ISBN 0-89791-009-5

[Blundell, 1982]:

Greggory S. Blundell: Microcomputer market soars on all fronts; Data Communications, December 1982, p.89-94; ISSN 0363-6399

[Boberg, 1980]:

Richard W. Boberg: Propose microcomputer system 796 bus standard, IEEE Task P796/D2; Computer, October 1980, p.89-105; ISSN 0018-9162

[Boberg, 1983]:

Boberg, Rich: Major standardization issues of the proposed IEEE 796 bus - Multibus; Microprocessors and Microsystems, Vol. 6, no. 9, p.471; ISSN 0141-9331

[Bochmann, 1979]:

Gregor v. Bochmann: Architecture of distributed computer systems; Springer Verlag, 1979; ISBN 3-540-09723-6

[Boehm, 1983]:

Barry W. Boehm: The hardware/software cost ratio: Is it a myth?; Computer, March 1983, p.78-80; ISSN 0018-9162

[Bolan, 1983]:

Michael Bolan: RAM dons lithium-cell hat for convenient nonvolatility; Electronics, June 30, 1983, p.147-150; ISSN 0013-5070

[Bonci, 1980]:

F. Bonci: Nuova Linea Sistemi: Il bus di sistima; Olivetti, GID, Progetto Sistimi, 1980;

[Borill, 1981]:

Paul L. Borill: Microprocessor bus structures and standards; IEEE Micro, February 1981, p.84-95; ISSN 0272-1732

[Borrill, 1980]:

Paul L. Borrill: P896-PLB-17 electrical and pinout considerations; IEEE P896 working document, June 4, 1980;

[Borrill, 1980.2]:

Paul L. Borrill: Proposal for Fastbus style protocol; IEEE P896 Design Course Documentation, June 16, 1980;

[Borrill, 1981]:

Paul L. Borrill: Backplane transmission line and crosstalk behaviour, Viewgraphs of presentation; IEEE P896 Design Course Documentation, June 28, 1981;

[Borrill, 1981.2]:

Paul L. Borrill: P896 electrical & mechanical requirements. Detailed specifications; IEEE P896 Design Course Documentation, June 28, 1981;

[Borrill, 1981.3]:

Paul L. Borrill: Results of arbiter test. A preliminary report to the P896 committee; IEEE P896 Design Course Documentation, may 31, 1981; P896-PLB-18

[Borrill, 1981.4]:

Paul L. Borrill: Implementation of task management on the parallel bus, general arbiter operat.; IEEE P896 Design Course Documentation, June 28, 1981;

[Borrill, 1983]:

Borrill, Paul: Backplane bus standards - why we need them .....; Microprocessors and Microsystems, Vol. 6, no. 9, p.450; ISSN 0141-9331

[Borrill, 1983.2]:

Borrill, Paul: IEEE P896 - the Futurebus project; Microprocessors and Microsystems, Vol. 6, no. 9, p.489; ISSN 0141-9331

[Branscomb, 1982]:

Lewis M. Branscomb: Bringing computing to the people: the broadening challenge; Computer, July 1982, p.68-75; ISSN 0018-9162

[Brender, 1981]:

Ronald F. Brender & Isaac R. Nassi: What is ADA?; Computer, June 1981, p.17-24; ISSN 0018-9162

[Bucci, 1982]:

Giacomo Bucci & Dario Maio: Merging Performance and Cost-benefit Analysis in Computer System Evaluation; Computer, September, 1982, p.23-31; ISSN 0018-9162

[Burkhardt, 1981]:

Heinz J. Burkhardt & Sigram Schindler: Structuring principles of the communication architecture of open systems; Computer Networks, No. 5, 1981, p.157-166; ISSN 0376-5075

[Bursky, 1979]:

Dave Bursky: LSI peripherals: uP's helping hands are strong and getting stronger; Electronic Design, Vol. 27, No. 24, 1979, p.118-126; ISSN 0013-4872

[Bursky, 1983]:

Dave Bursky: 32-bit CPU works with 4-Gbyte address; Electronic Design, July 21, 1983, p.42-43; ISSN 0013-4872

[Canning, 1980]:

EDP Analyzer: In your future: Local computer networks; Canning Publications, Inc., June, 1980, Vol. 18, no. 6;

[Canning, 1981]:

Richard G. Canning, editor: A new approach for local networks; EDP Analyzer, Vol. 19, No. 11, 1981; ISSN 0012-7523

[Canning, 1983]:

Richard G. Canning: Plan now for work stations; EDP Analyzer, February 1983, Vol.21 no.2; ISSN 0012-7523

[Canning, 1983.2]:

Richard G. Canning, Editor: Planning your distributed systems; EDP analyzer, June 1983; ISSN 0012-7523

[Card, 1983]:

Chuck Card, Donald Prigge, Josephine Walkowicz & Majorie Hill: The world of standards; Byte, February 1983, p.130-142; ISSN 0360-5280

[Card, 1983.2]:

Chuck Card: A proposed floppy disk format standard; Byte, February 1983, p.182-190; ISSN 0360-5280

[Carlson, 1982]:

Carlson, William E. & Fisher, David. A.: First complete ADA compiler runs on a micro; Mini-Micro Systems, September 1982, p.207-219; ISSN-0364-9342

[Carter, 1983]:

William Carter, Jackson Hu, Frank Lynch, David Stevenson: 8- and 16-bit processor family keep pace with fast RAMs; Electronic Design, April 28, 1983, p. 215-221;

[Castillo, 1980]:

Xavier Castillo: Workload, performance and reliability of digital computing systems; Department of Computer Science, Carnegie-Mellon University, December 1, 1980; CMU CS-81-104

[Ceferin, 1983]:

Jack V. Ceferin: Modular protocols improve industrial network control; Computer Design, January 1983, p.59-62; ISSN-0010-4566

[Chattergy, 1979]:

R. Chattergy & U.W. Pooch: A distributed function computer with dedicated processors; Computer Journal, Vol. 22, No. 1, 1979, p.37-40;

[Cheriton, 1979]:

David R. Cheriton et al: Thoth, a portable real-time operating system; Communications of the ACM, Vol. 22, No. 2, February 1979;

[Civera, 1982]:

P. Civera, G. Conte, D. Del Corso, F. Gregoretti & E. Pasero: The U* project: an experience with a multimicroprocessor system; IEEE Micro, May 1982, p.38-50; ISSN 0272-1732

[Clary, 1979]:

J.B. Clary & R.A. Sacane: Self-testing computers; Computer, Vol. 12, No. 10, 1979; ISSN 0018-9162

[Cody, 1981]:

W. J. Cody: Analysis of proposals for the floating-point standard; Computer, March 1981, p.63-68; ISSN 0018-9162

[Cohen, 1983]:

Kenneth I. Cohen: Multiprocessor architecture tunes in to transaction processing; Electronics, January 27, 1983, p.94-97; ISSN 0013-5070

[Coleman, 1982]:

Coleman, Vern: Ethernet and the PBX - a beneficial partnership; Computer Design, September 1982, p.183-188; ISSN-0010-4566

[Coleman, 1982.2]:

Vernon Coleman: Siliconizing the local area network; Computer Networks, Vol 6. No 4., 1982, p.245-254; ISSN 0376-5075

[Coleman, 1982.3]:

Vernon Coleman, Thomas Ermlovich & James Vittera: Controller chip shares tasks in buffer, net management; Electronics, October 6, 1983, p.92-94; ISSN 0013-5070

[Collins, 1982]:

John P. Collins & Miles M. Lewitt: An object oriented operating system for microcomputers; Computer Design, June 1982, p.165-172; ISSN 0010-4566

[Conrath, 1981]:

David W. Conrath, Chris Higgins, Cherian Thachenkary & Willian Wright: The electronic office and organizational behavior - measuring office activities; Computer Networks, Vol. 5, no. 6, 1981, p.401-410; ISSN 0376-5075

[Conte, 1980]:

Gianni Conte & Francesco Gregoretti: Software development and debug aids for the U* multimicroprocessor system; Reprint, IET, Torino, 1980;

[Conway, 1977]:

John W. Conway: Level 6 Inter System Link overview; P896 orientation document, March 29, 1977;

[Conway, 1979]:

Richard Conway, David Gries, Michael Fay & Charlie Bass: Introduction to microprocessor programming using PLZ; Winthrop, 1979; ISBN 0-87626-403-8

[Cooper, 1982]:

Edward Cooper: 13 often-asked question about broadband; Data Communications, April 1982, p.137-142; ISSN-0363-6399

[Cooper, 1983]:

Edward B. Cooper: Broadband Network Design: issues and answers; Computer Design, March 1983, p.209-216; ISSN-0010-4566

[Cooper, 1983.2]:

Edward B. Cooper & Philip K. Edholm: Design issues in broadband local networks; Data Communications, February 1983, p.109-122; ISSN 0363-6399

[Cotton, 1980]:

Ira W. Cotton: Technologies for local area computer networks; Computer Networks, Vol. 4, No. 5, 1980, p.197-208; ISSN 0376-5075

[Courtois, 1981]:

B. Courtois, M. Marinescu & J.F. Pons: SKALP, Skeleton architecture for fault-tolerant distributed computing; Microprocessing and Microprogramming, No. 7, 1981, p.312-225;

[Cragon, 1982]:

H.G. Cragon: The myth of the hardware-software cost ratio; Computer, December 1982, p.100-101; ISSN 0016-9162

[Crook, 1980]:

C. Crook: Microcomputer architecture - a survey report; Infotech State of the Art Report, Microelectronics, series 8, no. 2, p.83-112; ISBN 8553-9630-X

[DRN, 1983]:

<unknown>: CP/M-68K initiates portability strategy based on C; Digital Research News, Vol.3, no.2. 1983, p.2;

[Dahod, 1983]:

Ashraf M. Dahod: Local network standards: no utopia; Data Communications, March 1983, p.173-180; ISSN 0363-6399

[Datacomm, 1981]:

<unknown>: Broadband at base of Wang's far-reaching local network; Data Communications, July 1981, p.30; ISSN-0363-6399

[Datacomm, 1981.2]:

<unknown>: The Cambridge ring is still making the rounds; Data Communications, July 1981, p.32; ISSN-0363-6399

[Datacomm, 1982]:

Viewpoint: The IEEE 802 Committee states its case concerning its LAN standards efforts; Data Communications, April 1982, p.13; ISSN-0363-6399

[Datacomm, 1982.2]:

Newfront: In a sudden reversal, IEEE embraces Ethernet; Data Communications, April 1982, p.35-38; ISSN-0363-6399

[Davies, 1979]:

D.W. Davies, D.L.A. Barber, W.L. Price & C.M. Solomonides: Computer networks and their protocols; Wiley, 1979; ISBN 0-471-99750-1

[Davis, 1982]:

Dwight B. Davis: Datacomm's presence grows in small-computer markets; Mini-Micro Systems, March 1982, p.121-124; ISSN 0364-9342

[DeBock, 1983]:

Rich deBock: VERSAbus - a multiprocessor bus standard - and VMEbus - its Eurocard counter; Microprocessors and Microsystems, Vol. 6, no. 9, p.475; ISSN 0141-9331

[DeJoung, 1983]:

Ron DeJoung: 68000-based supermicro uses distributed architecture; Mini-Micro Systems, May 1983, p.227-231; ISSN 0364-9342

[DeSousa, 1981]:

M.R. DeSousa: Electronic information interchange in an office environment; IBM Systems Journal, Vol. 20, No. 1, p.4-22; ISSN 0018-8670

[Dean, 1983]:

Joseph Dean: Eurocard microcomputers a rugged alternative; Design News, February 7, 1983, p.102-103;

[Deiss, 1981]:

S.R. Deiss, R. Downing, D.B. Davidson, R. Larsen, C. Logg, L. Paffrath: Applicability of the Fastbus standard to distributed control; IEEE P896 Design Course Documentation, March 1981; SLAC-PUB-2703

[DelCorso, 1979]:

Dante Del Corso: A test technique for microprocessor based machines; IEEE P896 Design Course Documentation, February, 1979 (Italian magazine);

[DelCorso, 1980]:

Dante Del Corso, F. Gregoretti & P. Osella: Proposta preliminare, TOMP-80; Progretto MuMicro, Unita' di Torino, Febbraio 1980;

[DelCorso, 1980.2]:

Dante Del Corso: TOMP-80 multimicroprocessor system - preliminary specification; IET, Torino, II Revision, July 1st, 1980;

[DelCorso, 1981]:

Dante Del Corso: Description and classification of processor/peripheral interfaces; IEEE P896 Design Course Documentation, June 28, 1981;

[DelCorso, 1981.2]:

Dante Del Corso: Bus design choices for efficiency and ease of test; IEEE P896 Design Course Documentation, June 28, 1981;

[DelCorso, 1981.3]:

Dante Del Corso: M3BUS - Modular Multi Micro Bus - preliminary specification; IEEE P896 Design Course Documentation, March 3, 1981;

[Devlin, 1983]:

Phil Devlin: LSI chips ease design of hard-disk controller; Electronic Design, April 28, 1983, p. 189-194;

[Dhas, 1980]:

C. Retna Dhas & David C. Rine: Design of a ring-based local area network for microcomputers; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.65-68; IEEE 80CH1542-0

[Dias, 1983]:

Daniel M. Dias & J. Robert Jump: Packet switching interconnection networks for modular systems; Computer, December 1981, p.42-52; ISSN 0018-9162

[Digital, 1982]:

Unknown: Introduction to local area networks; Digital Equipment Corporation, 1982; EB 22714-18

[Dijkstra, 1974]:

Edsger W. Dijkstra: Self-stabilizing systems in spite of distributed control; Communications of the ACM, Vol. 17, No. 11, 1974, p.643-644;

[Donnelley, 1979]:

Jed Donnelley: Components of a network operating system; Computer Networks, December 1979, p.389-399; ISSN 0376-5075

[Donnelley, 1980]:

J.E. Donnelley & J.G. Fletcher: Resource access control in a network operating system; Proceedings of the ACM Pacific-80 conference, 1980, p.115-125;

[Dumoulin, 1981]:

D. Dumoulin: Supervision of data transfers on P896; IEEE P896 Design Course Documentation, June 28, 1981;

[EDISG, 1981]:

Multiprocessor/Multitask Subgroup of the Euro. Distr. Intel. Study Grp.: Microprocessor bus evaluation; IEEE P896 Design Course Documentation, June 28, 1981;

[Eimers, 1982]:

Eimers, Garth W.: Network, Heal Thyself: a diagnostic primer; Computer Design, September 1982, p.213-220; ISSN-0010-4566

[El-Ayat, 1979]:

K. A. El-Ayat: The Intel 8089: an integrated I/O processor; Computer, Vol.12, No. 6, June 1979; ISSN 0018-9162

[Electronics, 1983]:

<unknown>: A plethora of projects in the U.S. try data-flow and other architectures; Electronics, June 16, 1983, p.107-110; ISSN 0013-5070

[Electronics, 1983.2]:

<unknown>: Western Europe looks to parallel processing for future computers; Electronics, June 16, 1983, p.111-113; ISSN 0013-5070

[Electronics, 1983.3]:

<unknown>: Japan is busy trying to make manufacturable data-flow computers; Electronics, June 16, 1983, p.114; ISSN 0013-5070

[Ellis, 1980]:

Clarence A. Ellis & Gary J. Nutt: Office information systems and computer science; Computing Surveys, Vol. 12, No. 1, March 1980, p.27-60;

[Elphick, 1983]:

Michael Elphick & Richard Parker: Winchester disk technologies spins into new orbits; Computer Design, January 1983, p.89-102; ISSN 0010-4566

[Elsmore, 1983]:

Elsmore, Tim: Standard bus for 8-bit microprocessor systems; Microprocessors and Microsystems, Vol. 6, no. 9, p.455; ISSN 0141-9331

[Enslow, 1974]:

Philip H. Enslow: Multiprocessors and parallel processing; John Wiley & Sons, 1974; ISBN 0-471-16735-5

[Enslow, 1978]:

Philip H. Enslow: What is a distributed data processing system?; Computer, January 1978, p.13-21; ISSN 0018-9162

[Ether, 1980]:

The blue book: The Ethernet: A local area network. Data link layer and physical layer specs.; Digital, Intel, Xerox, Version 1.0, september 30, 1980;

[Ether, 1982]:

New products: Ethernet comes to desktop stations (IBM PC); Electronics, November 3, 1982, p.162; ISSN 0013-5070

[Evanczuk, 1983]:

Stephen Evanczuk: Designers tune UNIX for real-time use; Electronics, March 24, 1983, p.111-114; ISSN 0013-5070

[Evanczuk, 1983.2]:

Stephen Evanczuk: Chips come to aid of embedded systems; Electronics, March 24, 1983, p.114-115; ISSN 0013-5070

[Farber, 1974]:

David J. Farber: Software considerations in distributed architectures; Computer, Vol.7, No.3, March 1974; ISSN 0018-9162

[Fathi, 1983]:

Eli T. Fathi & Moshe Krieger: Multiple microprocessor systems: What, Why, and When; Computer, March 1983, p.23-32; ISSN-0018-9162

[Feldman, 1979]:

Jerome A. Feldman: High level programming for distributed computing; Communications of the ACM, Vol. 22, No. 6, June 1979, p.353-367;

[Felix, 1980]:

Robert G. Felix: Organization and policy issues in the implementation of distributed systems; Proceedings of the ACM Pacific-80 conference, 1980, p.30-37;

[Feng, 1981]:

Tse-yun Feng: A survey of interconnection networks; Computer, December 1981, p.12-27; ISSN 0018-9162

[Flemming, 1983]:

Jim Flemming & William Frezza: NAPLPS: a new standard for text and graphics; Byte, February 1983, p.203-254; ISSN 0360-5280

[Folkes, 1983]:

Don Folkes, Alan Gant & Denise Burrows: Built-in I/O support beefs up 16-bit uP; Electronic Design, May 26, 1983, p.153-160; ISSN 0013-4872

[Folts, 1979]:

Harold C. Folts: Status report on new standards for DTE/DCE interface protocols; Computer, Vol.12, No. 9, September 1979; ISSN 0018-9162

[Folts, 1981]:

Harold C. Folts: Coming of Age: A long-awaited standard for heterogeneous nets; Data Communications, January 1981, p.63-73; ISSN 0363-6399

[Frank, 1983]:

Werner L. Frank: The history of myth no. 1; Datamation, May 1983, p.252-256; ISSN 0011-6963

[Freedman, 1983]:

Freedman, David: Portable operating systems fight for 16-bit machines; Mini-Micro Systems, September 1982, p.237-249; ISSN-0364-9342

[Freedman, 1983.2]:

Roy S. Freedman: The common sense of object oriented languages; Computer Design, February 1983, p.111-118; ISSN 0010-4566

[Fuller, 1978]:

Samuel H. Fuller et al: Multi-microprocessors: an overview and working example; Proceedings of the IEEE, Vol. 66, No. 2, 1978, p.216-228;

[GKS, 1982]:

Dradt International Standard ISO/DIS 7942 version 7.02: Graphical Kernel System (GKS) - functional description; American National Standards Institute, Inc., August 9, 1982;

[GSG, 1982]:

General Systems Group, Inc.: Review of the system performance of the TP2 system; , March 2, 1982;

[Garetz, 1983]:

Garetz, Mark: P696/S100 - a bus which supports a wide range of 8- and 16-bit processors; Microprocessors and Microsystems, Vol. 6, no. 9, p.467; ISSN 0141-9331

[Garetz, 1983.2]:

Mark Garetz: The IEEE standard for the S-100 bus; Byte, February 1983, p.272-298; ISSN 0360-5280

[Geisler, 1981]:

Pamela Geisler: Design approach for 16 bit; Systems International, November 1981, p.24-28; ISSN 0309-1171

[Gibson, 1982]:

Ronald W. Gibson: Comparing features aids selecting broadband local net; Data Communications, April 1982, p.127-135; ISSN-0363-6399

[Gibson, 1982.2]:

Ronald W. Gibson: A primer for evaluating and purchasing today's local networks; Data Communications, June 1982, p.147-156; ISSN-0363-6399

[Gibson, 1983]:

Ronald W. Gibson: Local net user relates trials, tribulations; Data Communications, March 1983, p.121-129; ISSN 0363-6399

[Gifford, 1981]:

David K. Gifford: Violet, an experimental decentralized system; Computer Networks, Vol. 5, no. 6, 1981, p.423-433; ISSN 0376-5075

[Goldberg, 1982]:

Jeff Goldberg: Unix-like system standards; Computer, July 1982, p.79-80; ISSN 0018-9162

[Goldstein, 1980]:

Barry C. Goldstein: Inversion of the memory hierarchy; Proceedings of the ACM Pacific-80 conference, 1980, p.154-157;

[Gooding, 1983]:

Claire Gooding & George Black: The Pick micro race is underway; Computer Weekly, April 8, 1983, p.8;

[Goodrich, 1982]:

Allen B. Goodrich: Link-controller IC combines versatility and flexibility; Electronics, October 6, 1982, p.101-103; ISSN 0013-5070

[Gordon, 1979]:

R.L. Gordon, W.W. Farr & P. Levine: Ringnet: a packet switched local network with decentralized control; Computer Networks, December 1979, p.373-379; ISSN 0376-5075

[Gordon, 1980]:

R.L. Gordon: Perspectives on the evolution of commercial local networking; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.3-7; IEEE 80CH1542-0

[Gordon, 1982]:

Michael F. Gordon & Stephen V. Cope: Coprocessing to ease the graphics burden; Computer Design, July 1982, p.147-152; ISSN 0010-4566

[Grant, 1982]:

Alastair Grant & David Hutchison: X25 protocols and local area networks; Computer Networks, Vol 6. No 4., 1982, p.255-262; ISSN 0376-5075

[Graube, 1982]:

Maris Graube: Local area nets: a pair of standards; IEEE Spectrum, June 1982, p.60-64; ISSN 0018-9235

[Gupta, 1983]:

Amar Gupta & Hoo-min D. Toong: An architectural comparison of 32-bit microprocessors; IEEE Micro, February 1983, p. 9-22; ISSN-0272-1732

[Gustavson, 1979]:

David B. Gustavson: An introduction to the Fastbus; IEEE P896 Design Course Documentation, August 1979; SLAC-PUB-2378

[Gustavson, 1981]:

David B. Gustavson: Comments on supervisors versus snoops; IEEE P896 Design Course Documentation, June 7, 1981;

[Gustavson, 1981.2]:

David B. Gustavson: A summary of the Fastbus protocol for P896; IEEE P896 Design Course Documentation, May 14, 1981;

[Gustavston, 1981]:

David B. Gustavson: The Wire-Or glitch: implications for P896; IEEE P896 Design Course Documentation, May 12, 1981;

[Guyod, 1983]:

Gerald Guyod & Dick Vink: OASIS, PICK and UNIX: OS for OEMs; Datamation, April 1983, p. 282-27 - 282-38; ISSN 0011-6963

[Halfant, 1983]:

Matthew Halfant: The UNIX C-compiler in a CP/M environment; BYTE, August 1983, p.243-267; ISSN 0360-5280

[Hansen, 1973]:

Per Brinch Hansen: Operating system principles; Prentice-Hall, 1973; ISBN 0-13-637843-9

[Hansen, 1977]:

Per Brinch Hansen: The architecture of concurrent programs; Prentice-Hall, 1977; ISBN 0-13-0044628-9

[Hansen, 1979]:

Per Brinch Hansen: Multiprocessor architectures for concurrent programs; Computer Architecture News, Vol. 7, No. 4, 1979, p.4-23;

[Hansen, 1980]:

Bent Hansen: McTan er en ny mikrodatamat der er udviklet i SDC; SDC informerer, April 1980, p.9-12;

[Harakal, 1981]:

Joseph P. Harakal: Putting Ethernet onboard the MULTIBUS; Computer Design, September 1981, p.183-186; ISSN 0010-4566

[Harrison, 1983]:

Nigel Harrison: Evolution of the Multibus; Systems International, March 1983, p.47-49; ISSN 0309-1171

[Heckel, 1977]:

Paul G. Heckel & Butler W. Lampson: A terminal-oriented communication system; Communications of the ACM, Vol. 20, No. 7, July 1977;

[Heider, 1982]:

Heider, George: Let operating systems aid in component designs; Computer Design, September 1982, p.151-160; ISSN-0010-4566

[Held, 1983]:

Gilbert Held: Strategies and concepts for linking today's personal computers; Data Communications, May 1983, p.211-219; ISSN 0363-6399

[Hemenway, 1981]:

Jack Hemenway & Robert Grappel: Understand the newest processor to avoid future shock; Electronic Design News, April 29, 1981, p.129-136;

[Hennessy, 1980]:

John Hennessy: Pascal* : A Pascal based systems programming language; Computer Systems Laboratory, Stanford University, Technical Report 174, July 1980;

[Herald, 1983]:

Robert F. Herald & Marvin E. Prahl: Sure and steady 4-inch drive keeps weight and cost down. How? Simplification; Electronic Design, June 9, 1983, p.221-224; ISSN 0013-4872

[Hersch, 1981]:

R.D. Hersch & D. Dumoulin: L896 - bus interface design for 68000 uP systems; IEEE P896 Design Course Documentation, June 28, 1981;

[Highleyman, 1980]:

Wilbur H. Highleyman: Survivable systems; ComputerWorld - In Depth series, Februar/March 1980;

[Hill, 1983]:

Hill, John: Eurobus - the UK-designed general-purpose backplane bus; Microprocessors and Microsystems, Vol. 6, no. 9, p.483; ISSN 0141-9331

[Hindin, 1982]:

Harvey J. Hindin: Dual-chip sets forge vital link for Ethernet local-network scheme; Electronics, October 6, 1983, p.89-91; ISSN 0013-5070

[Hindin, 1983]:

Harvey J. Hindin: Local-net standardization gains; Electronics, March 24, 1983, p.98-99; ISSN 0013-5070

[Hoare, 1974]:

C.A.R. Hoare: Monitors: an operating system structuring concept; Communications of the ACM, Vol. 17, No. 10, 1974, p.549-557;

[Hobbs, 1970]:

L.C. Hobbs: Parallel processor systems, technologies and applications; Spartan Books, 1970; ISBN 0-87671-161-1

[Horovitz, 1982]:

Danny Horovitz, Frank Holsworth & John Wharton: Concurrency: key to 16-bit operating system efficiency; Computer Design, June 1982, p.183-192; ISSN 0010-4566

[Hough, 1981]:

David Hough: Applications of the proposed IEEE 754 standard for floating-point arithmetic; Computer, March 1981, p.70-74; ISSN 0018-9162

[Houston, 1983]:

Jerry Houston, Jim Brodrick & Les Kent: Comparing C compilers for CP/M-86; BYTE, August 1983, p.82-106; ISSN 0360-5280

[Hsi, 1980]:

Peter Hsi & Tsvi Lissack: Local networks' consensus: high speed; Data Communications, December, 1980, p.56-66; ISSN 0363-6399

[Huie, 1983]:

Jonathan Huie, Richard Lowenthal & Steven Blank: MegaFrame aims at mainframe performance; Mini-Micro Systems, April 1983, p.157-162; ISSN 0364-9342

[Hunt, 1979]:

V. Bruce Hunt & Pier Carlo Ravasio: Preliminary protocol architecture for the Olivetti local network system; SRI International, Client Private Technical Report, SRI Project 8874, December 1979;

[Hunter, 1983]:

Bruce Hunter: A comparison of C compilers; Lifelines, May 1983, p.15-21; ISSN 0279-2575

[Hutchison, 1982]:

David Hutchison & Peter Corcoran: Building local area networks to Ethernet specification; Computer Communications, Vol. 5, No. 1, 1982, p.12-16; ISSN 0140-3664

[Hwang, 1982]:

Kai Hwang et al: A UNIX-based local computer network with load balancing; Computer, April 1982, p.55-66; ISSN 0018-9162

[I896, 1982]:

I896 Working Group of EWICS - TC10(2): A proposed standard backplane bus specicication for advanced microcomputer sys.; I896 Draft 5.1, November 24, 1982;

[IEEE, 1982]:

Local Network Standards Committee: IEEE Project 802 - a status report; IEEE Computer Society, October 1981;

[IEEE, 1982.2]:

IEEE project 802 / Local area network standards: Draft IEEE standard 802.2 - Logical Link Control; IEEE Computer Society, Draft D, November 1982; UNAPproved draft published for comment

[IEEE, 1982.3]:

IEEE project 802 / Local area network standards: Draft IEEE standard 802.3 - CSMA/CD Access Method & Physical Layer Specs.; IEEE Computer Society, Draft D, December 1982; UNAPproved draft published for comment

[IEEE, 1982.4]:

IEEE project 802 / Local area network standards: Draft IEEE standard 802.4 - Token-Passing Bus Access Method & Physical Layer S.; IEEE Computer Society, Draft D, December 1982; UNAPproved draft published for comment

[Inselberg, 1983]:

Armond D. Inselberg: Multiprocessor architecture ensures fault-tolerant transaction processing; Mini-Micro Systems, April 1983, p.165-172; ISSN 0364-9342

[Intel, 1981]:

: The micromainframe computer; Solutions, Intel European edition, july/august 1981, p.2-5;

[Intel, 1981.2]:

: Local area networks; Solutions, Intel European edition, September/october 1981, p.2-5;

[Intel, 1982]:

Unknown: High performance microprocessor opens new doors for system designers; Solutions, Intel European edition, May/June 1982, p. 2-5;

[Intel, 1982.2]:

Unknown: Intel's first integrated microsystem is based on VLSI standards; Solutions, Intel European edition, February, 1982, p.2-5;

[Intel, 1982.3]:

Intel Ethernet product specs.: The complete VLSI LAN solution; Intel, 1982;

[Intel, 1983]:

Unknown: Ethernet chips cut LAN connection costs; Solutions, Intel European edition, Januray/February 1983, p.14-15;

[Intel, 1983.2]:

Unknown: Components and software for the iAPX 286; Solutions, Intel European edition, Januray/February 1983, p.18-19;

[Intel, 1983.3]:

Intel technical manual: 82586 Local Communications Controller Reference Manual; 210891-001, January 1983;

[Isaak, 1982]:

Jim Isaak: Squeezing the most out of the 68000; Mini-Micro Systems, October 1982, p.193-202; ISSN 0364-9342

[Jackson, 1983]:

Paul Jackson: UNIX variant opens a path to managing multiprocessor systems; Electronics, July 28, 1983, p.118-124; ISSN 0013-5070

[Jacob, 1983]:

Larry Jacob: Tailor the winchester to the system; Computer Design, January 1983, p.107-112; ISSN 0010-4566

[Jarrett, 1983]:

Tom Jarrett & Bill Rhodes: Microfloppy-disk drive adopts 3-1/2 inch diskette endorsed by industry; Electronic Design, June 9, 1983, p.213-217; ISSN 0013-4872

[Jaworsk, 1983]:

Joseph V. Jaworsk: Flexible controller mates with popular Winchester drives; Electronic Design, April 28, 1983, p. 175-182;

[Jessen, 1982]:

Stan Jessen & Knox Rannill: Device independent color graphics language for micros; Computer Design, July 1982, p.137-141; ISSN 0010-4566

[Jones, 1978]:

Anita K. Jones et al: Programming issues raised by a multiprocessor; Proceedings of the IEEE, Vol. 66, No. 2, 1978, p.229-237;

[Jones, 1982]:

Thomas C. Jones: Paving the way for universal document interchange; Data Communications, July 1982, p.123-131; ISSN 0363-6399

[Jones, 1982.2]:

Keith Jones: Racal-Milgo announces token-passing network; Mini-Micro Systems, March 1982, p.77-78; ISSN 0364-9342

[Joseph, 1974]:

Earl C. Joseph: Innovations in heterogeneous and homogeneous distributed-function architectures; Computer, Vol.7, No.3, March 1974; ISSN 0018-9162

[Joy, 1983]:

Bill Joy: Berkeley 4.2 gives UNIX operating system network support; Electronics, July 28, 1983, p.114-118; ISSN 0013-5070

[Jurich, 1981]:

Dale Jurich: 432 Packet bus protocol supports multiple processors; Solutions, Intel European edition, july/august 1981, p.6-9;

[Kahn, 1978]:

Kevin C. Kahn: A small-scale operating system foundation for microprocessor applications; Proceedings of the IEEE, Vol. 66, No. 2, 1978, p.209-216;

[Kayaler, 1983]:

Peter Kayaler & Alan Greenspan: At-net - A new UNIX based transparent networking system; Altos Computer Systems, June 1, 1983;

[Kenealy, 1982]:

Kenealy, Patrick: Database software packages for micros; Mini-Micro Systems, September 1982, p.193-202; ISSN-0364-9342

[Kern, 1983]:

Christopher O. Kern: Five C compilers for CP/M-80; BYTE, August 1983, p.110-130; ISSN 0360-5280

[Kernighan, 1981]:

Brian W. Kernighan & John R. Mashey: The UNIX programming environment; Computer, April 1981, p.12-24; ISSN-0018-9162

[Killen, 1983]:

Michael Killen: The microcomputer connection to local networks; Data Communications, December 1983, p.97-113; ISSN 0363-6399

[King, 1983]:

Kenneth J. King & Fred J. Maryanski: Information management trends in office automation; Proceedings of the IEEE, Vol. 71, no.4, 1983, p.519-528; ISSN 0018-9219

[Kirrmann, 1981]:

Hubert Kirrmann: A serial interprocessor link for multiprocessor management in the P896 bus; IEEE P896 Design Course Documentation, June 28, 1981;

[Kirrmann, 1981.2]:

Hubert Kirrmann: Proposals for the P896 protocol; IEEE P896 Design Course Documentation, June 2, 1981;

[Knuth, 1974]:

Donald E. Knuth: Computer programming as an art; Communications of the ACM, Vol. 17, No. 12, 1974, p.667-673;

[Kobayashi, 1981]:

Koji Kobayashi: Computer, Communications and Man: The integration of C&C. with man as an axis; Computer Networks, 1981, p.237-250; ISSN 0376-5075

[Koehler, 1982]:

Koehler, Stephen C. & Franks, William P: Native Code teams with pseudocode to hasten emulation; Computer Design, September 1982, p.139-144; ISSN-0010-4566

[Kopec, 1982]:

Stan Kopec & Dane Elliot: Design latitude broadens link-controller chip's appeal; Electronics, October 6, 1982, p.98-100; ISSN 0013-5070

[Kumar, 1979]:

B. Kumar & Timothy A. Gonsalves: Modelling and analysis of distributed software systems; Proceedings of the seventh symposium on operating systems principles, ACM No. 534790; 1979, p.2-8; ISBN 0-89791-009-5

[LOCALNetter, 1982]:

Architecture Technology Corporation: the LOCALNetter Designer's Handbook; , 1982 Edition;

[Lampson, 1976]:

Butler W. Lampson & Howard E. Sturgis: Reflections on an operating system design; Communications of the ACM, may 1976, p.251-265;

[Landwehr, 1983]:

Carl E. Landwehr: The best available technologies for computer security; Computer, July 1983, p.86-100; ISSN 0018-9162

[Langhorst, 1982]:

Fred. E. Langhorst: Working towards standards in graphics; Computer Design, July 1982, p.177-182; ISSN 0010-4566

[Langhorst, 1983]:

Fred E. Langhorst & Thomas B. Clarkson III: Realizing graphics standards for microcomputers; Byte, February 1983, p.256-268; ISSN 0360-5280

[Lantz, 1979]:

Keith A. Lantz & Richard F. Rashid: Virtual terminal management in a multiple process environment; Proceedings of the seventh symposium on operating systems principles, ACM No. 534790; 1979, p.86-97; ISBN 0-89791-009-5

[Lantz, 1980]:

Keith A. Lantz: RIG, an architecture for distributed systems; Computer systems laboratory, Stanford University, 1980;

[Lantz, 1980.2]:

Keith A. Lantz: Uniform Interfaces for Distributed Systems; Computer Science Department, University of Rochester, May 1980;

[Larson, 1983]:

Kenneth N. Larson & W. Roy Chestnut: Adding another layer to the ISO net architecture reduces cost; Data Communications, March 1983, p.215-222; ISSN 0363-6399

[Lauer, 1979]:

Hugh C. Lauer & Poger M. Needham: On the duality of operating system structures; Proc. of the second international syposium on operating systems. IRIA, October, 1979;

[Lavi, 1980]:

Yohav Lavi et al: 16-bit microprocessor enters virtual memory domain; Electronics, April 24, 1980, p.123-129;

[Lax, 1983]:

Leo Lax & Mark Olsen: NAPLPS standard graphics and the microcomputer; BYTE, July 1983, p.82-92; ISSN 0360-5280

[Lettieri, 1982]:

Larry Lettieri: Software-in-Silicon boots system performance, cuts programming time; Mini-Micro Systems, March 1982, p.93-95; ISSN 0364-9342

[Levy, 1978]:

John V. Levy: Buses, the skeleton of computer structures; Computer Engineering: a DEC view of hardware systems design, 1978; P896 document

[Levy, 1982]:

Walther A. Levy & Harriet Mehl: Taking a new look at matrix-switched systems; Mini-Micro Systems, March 1982, p.147-156; ISSN 0364-9342

[Lewis, 1982]:

Lewis, Geoff: Pick jockeys for position in standard operating-system race; Mini-Micro Systems, September 1982, p.32-42; ISSN-0364-9342

[Lewis, 1982.2]:

Geoff Lewis: UNIX and the supermicrocomputer: a marriage of convenience; Mini-Micro Systems, November 1982, p.17-22; ISSN 0364-9342

[Lewis, 1982.3]:

Andrea Lewis: 16-bit operating systems, a whole new ball game; Computer Design, June 1982, p.197-204; ISSN 0010-4566

[Liskov, 1979]:

Babara Liskov: Primitives for distributed computing; Proceedings of the seventh symposium on operating systems principles, ACM No. 534790; 1979, p.33-42; ISBN 0-89791-009-5

[Lochovsky, 1983]:

Fred H. Lochovsky: Improving office productivity: a technology perspective; Proceedings of the IEEE, Vol. 71, no.4, 1983, p.512-518; ISSN 0018-9219

[Logg, 1981]:

C.A. Logg, L. Paffrath, B. Bertolucci & D. Horelick: Fastbus introduction and demonstration; IEEE P896 Design Course Documentation, May 1981;

[Lorin, 1980]:

Harold Lorin & Barry C. Goldstein: Operating system structures for polymorphic hardware; Proceeding of the Pacific-80 ACM conference, 1980;

[Lowenthal, 1982]:

Eugene Lowenthal: Multiuser microprocessor systems get a data-base manager; Electronics, June 30, 1983, p.113-117; ISSN 0013-5070

[Lyman, 1981]:

Rich Lyman: Comparison of bus arbitrations; IEEE P896 Design Course Documentation, June 28, 1981;

[Lyman, 1981.2]:

Rich Lyman: P986 bus arbitration proposal; IEEE P896 Design Course Documentation, May 10, 1981;

[Lyman, 1981.3]:

Richard Lyman: P896 Specification Structure; P896 committee document, June 20, 1981;

[MacKenna, 1981]:

Graig MacKenna: Amended P896 supervisory handshake proposal; IEEE P896 Design Course Documentation, June 9, 1981;

[Madnick, 1974]:

Stuart E. Madnick & John D. Donovan: Operating Systems; McGraw-Hill, 1974; ISBN 0-07-039455-5

[Maine, 1981]:

Paul A. D. de Maine: A programming language for networks; Data Communications, July 1981, p.70-76; ISSN-0363-6399

[Manison, 1983]:

Manison, Andrew: Software Transportability; Systems International, April 1983, p.65 - 66;

[Manuel, 1983]:

Tom Manuel: Special processor handles connection to local networks; Electronics, January 27, 1983, p.47-48; ISSN 0013-5070

[Manuel, 1983.2]:

Tom Manuel: Advanced parallel architectures get attention as way to faster computing; Electronics, June 16, 1983, p.105-106; ISSN 0013-5070

[Marglaris, 1980]:

Basil Marglaris & Tsvi Lissack: An integrated broadband local network architecture; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.31-37; IEEE 80CH1542-0

[Marsan, 1979]:

M. Ajmone Marsan, G. Conto, D. Del Corso & F. Gregoretti: Architecture, communication procedures and performance eval. of the U* system; 1st Conf. on distributed computer systems, IEEE, October, 1979, p.106-115;

[Maryanski, 1980]:

Fred. J. Maryanski: Backend database systems; Computing Surveys, Vol. 12, No. 1, March 1980, p.3-25;

[Mason, 1982]:

Alan Mason: Criteria for Unix-like systems; Computer, September, 1982, p.94-96; ISSN 0018-9162

[Mateosian, 1983]:

Richard Mateosian: 16-bit uPs get a boost from demand-paged MMU; Electronic Design, May 26, 1983, p.179-184; ISSN 0013-4872

[McKevitt, 1979]:

James McKevitt & John Bayliss: New options from big chips; IEEE Spectrum, Vol. 16, No. 3, 1979, p.28-34;

[McLenning, 1983]:

McLening, Maggie: Bridging the gap (16-bit operating systems); International Systems, April 1983, p. 68-69;

[McNamara, 1978]:

J. E. NcNamara: Technical aspects of data communication; Digital Press, Digital Equipment Corporation, 1978;

[McQuillan, 1980]:

John M. McQuillan: Local network technology and the lessons of history; Computer Networks, Vol. 4, No. 5, 1980, p.235-238; ISSN 0376-5075

[Metcalfe, 1976]:

Robert M. Metcalfe & David R. Boggs: Ethernet: Distributed packet switching for local computer networks; Communications of the ACM, July 1976, p. 395-404;

[Microdata, 1976]:

Microdata Corporation: 3200 microprogramming reference manual (S1000); Proprietary information, February, 1976; 98800-76-1020A

[Microdata, 1978]:

Microdata Corporation: Express III software instruction set (S1000); Proprietary information, September 2, 1978; 52936

[Mier, 1982]:

Edwin E. Mier: High level protocols, standards, and the OSI reference model; Data Communications, July 1982, p.71-101; ISSN 0363-6399

[Mier, 1982.2]:

Edwin E. Mier: Patent snafu hits 802; Data Communications, August 1982, p.38-39; ISSN 0363-6399

[Miller, 1982]:

C. Kenneth Miller & David M. Thompson: Making a case for token passing in local networks; Data Communications, March 1982, p.79-88; ISSN 0363-6399

[Miller, 1983]:

Darby Miller: Videotex - Science fiction or reality?; BYTE, July 1983, p.42-56; ISSN 0360-5280

[Mitchell, 1983]:

Peter Mitchell: The narrowing supermini gap; Systems International, March 1983, p.55-57; ISSN 0309-1171

[Moldow, 1981]:

Bert D. Moldow: Reality and the proposed OSI standard; Data Communications, June 1981, p.77-80; ISSN 0363-6399

[Molina, 1982]:

Hector Garcia-Molina: Reliability issues for fully replicated distributed databases; Computer, September, 1982, p.34-42; ISSN 0018-9162

[Mor, 1983]:

S. Mor, H. Hingarh, M. Vora, D. Wilnay, D. Maxwell & T.Longo: 16-bit bipolar microprocessor marches to standard instruction set; Electronics, April 7, 1983, p.134-139; ISSN 0013-5070

[Morse, 1980]:

Stephen P. Morse, Bruce W. Ravenel, Stanley Mazor & William B. Pohlman: Intel microprocessors - 8008 to 8086; Computer, October 1980, p.42-60; ISSN 0018-9162

[Muehlemann, 1981]:

K. Muehlemann: Real time operating system approaches for multiprocessors; IEEE P896 Design Course Documentation, June 28, 1981;

[Murphy, 1983]:

John A. Murphy: Integrated office-automation systems; Mini-Micro Systems, May 1983, p.181-188; ISSN 0364-9342

[Muston, 1979]:

John Muston: Trends in 16-bit architecture; Microprocessors and Microsystems, April 1979, p.129-133; ISSN 0141-9331

[Myers, 1981]:

Ware Myers: Computer graphics: the need for graphics design, part one; Computer, June 1981, p.86-92; ISSN 0018-9162

[Myers, 1982]:

Ware Myers: Toward a local network standard; IEEE Micro, August 1982, p.28-45; ISSN 0272-1732

[Myers, 1982.2]:

Curtis Myers & Grant Munsey: A multiprocessor minicomputer designed for UNIX; Computer Design, February 1982, p.87-96; ISSN 0010-4566

[NCR, 1983]:

<unknown>: The NCR/32 processor family; NCR, 1983; MD 601 0882

[NCR, 1983.2]:

<unknown>: NCR 9300 32-bit mainframe processor... A new generation; NCR, 1983; SP 852 0283

[Naffah, 1981]:

Najah Naffah: Communication protocols for integrated office systems; Computer Networks, Vol. 5, no. 6, 1981, p.445-454; ISSN 0376-5075

[Nelson, 1981]:

Bruce Jay Nelson: Remote procedure call; Department of Computer Science, Carnegie-Mellon University, May 1981; CMU CS-81-119

[Nickens, 1980]:

Don O. Nickens, Thomas B. Genduso & Stanley Y. W. Su: The architecture and hardware implementation of a prototype MICRONET; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.56-64; IEEE 80CH1542-0

[Nicoud, 1981]:

Jean-Daniel Nicoud: Data transfers in P896; IEEE P896 Design Course Documentation, april 29, 1981;

[Noice, 1983]:

Judy Noice: CP/M on a chip; Solutions, Intel European edition, Januray/February 1983, p.14;

[Norman, 1983]:

Norman, Stephen: HP's PC Strategy; Systems International, April 1983, p.62-63;

[Noyce, 1981]:

Robert N. Noyce & Marcian E. Hoff: A history of microprocessor development at Intel; IEEE Micro, February 1981, p.8-21; ISSN 0272-1732

[OL, 1977]:

unknown: Monobus Specification; Olivetti, 1977;

[OL, 1979]:

unknown: Sample of instruction execution time (S1000); Olivetti, July 3, 1979;

[OL, 1981]:

Olivetti: TC 1808, Theory of operation; SDC documentation, December 1981; GP Code 3978990 Z (1)

[OL, 1981.2]:

Olivetti: System 1000, System Summary; SDC documentation, March 1981; GS 3947640 D (1)

[OL, 1981.3]:

Olivetti: TC 1808, TCOSMOS - concepts and facilities; SDC documentation, August 1981; GU Code 3978940 (2)

[OL, 1981.4]:

Olivetti: SDC System, APMAN - Concepts and facilities; SDC documentation, 1981; LU Code 3979250 J (2)

[OL, 1981.5]:

Olivetti: SDC System, SBPL - User Guide; SDC documentation, May 1981; LP Code 3979260 K (1)

[OL, 1981.6]:

Olivetti: System 1000 - Theory of operation; SDC documentation, November 1981; GP Code 3947660

[OL, 1982]:

Olivetti: SDC System, Application environment operating procedures; SDC documentation, February 1982; LO Code 3981450 D (1)

[OL, 1982.2]:

Olivetti: SDC System, TP-monitor - Concepts and facilities; SDC documentation, 3rd edition, May 1982; LP Code 3979320 R (2)

[OL, 1982.3]:

Olivetti, Ivrea: Transfer rate in a 'filesave' operation using floppy disk (single file, 700 KB); Attachment to SDC/OL tape backup meeting, February 18, 1982; SDC K2.60.4.77

[OL, 1982.4]:

Olivetti: SDC System: System Overview and Network Components; SDC documentation, January 1982; LS Code 3947680 R (1)

[OL, 1982.5]:

Olivetti: S1000 service manual; SDC documentation, September 1982; Code 3962310 Z (0)

[Ousterhout, 1980]:

John K. Ousterhout, Donald A. Scelza & Pradeep S. Sindhu: Medusa: an experiment in distributed operating system structure; Communications of the ACM, Vol.23, No.2, 1980, p.92-117; ISSN 0001-0782

[Overgaard, 1982]:

Overgaard, Mark: Implementing the UCSD p-System; Mini-Micro Systems, September 1982, p.183-190; ISSN-0364-9342

[P896, 1980]:

P896 Backplane Bus Subcommittee: Standard specification for microcomputer system backplanes; IEEE Computer Society Microprocessor Standards Committee, July 18, 1980; DRAFT P896/D1

[P896, 1981]:

P896 committee: Proposed standard specification for P896/D 3.4 advanced mc. system backplane; IEEE Computer Society Micriprocessor Standards Committee, April 13, 1981;

[P896, 1981.2]:

P896 Backplane Subcommittee: Proposed standard specification for P896/D 3.4.1 advanced uP system backplane; IEEE Computer Society Microprocessor Standards Committee, May 26, 1981;

[P896, 1981.3]:

P896 backplane subcommittee: Proposed standard specification for P896/D 4. advanced mc. system backplane; temporary document, November 3, 1981;

[Palmer, 1980]:

David F. Palmer: Resource requirements definition in distributed system design; Proceedings of the ACM Pacific-80 conference, 1980, p.70-76;

[Parker, 1983]:

Parker, Richard: Committees push to standardize disk I/O; Computer Design, March 1983, p.30-34; ISSN-0010-4566

[Parker, 1983.2]:

Parker, Richard & Shapiro, Sydney: Untangling Local Area Networks; Computer Design, March 1983, p.159-172; ISSN-0010-4566

[Patstone, 1981]:

W. Patstone: 16-bit uP benchmark - an update with explanations; Electronic Design News, September 16, 1981, p. 169-184;

[Pearson, 1983]:

Gregory Pearson: Anatomy of a microcomputer protocol; Data Communications, March 1983, p.231-239; ISSN 0363-6399

[Peterson, 1983]:

C.B. Peterson et al: Two Chips endow 32-bit processor with fault-tolerant architecture; Electronics, April 7, 1983, p.159-164; 0013-5070

[Peuto, 1979]:

B.L. Peuto: Architecture of a new microprocessor; Computer, February 1979, p.10-21; ISSN 0018-9162

[Pfister, 1982]:

George M. Pfister & Bradley V. O'Brien: Comparing the CBX to the local network - and the winner is?; Data Communications, July 1982, p.103-113; ISSN 0363-6399

[Phraner, 1983]:

Ralph A. Phraner: Nine C compilers for the IBM PC; BYTE, August 1983, p.134-168; ISSN 0360-5280

[Piatkowski, 1980]:

Thomas F. Piatkowski: The ISO-ANSI open systems reference model - a proposal for a systems approach; Computer Networks, No. 4, 1980, p.111-124; ISSN 0376-5075

[Piney, 1980]:

C. Piney, C. Parkman & F. Fluckiger: Endpoint interconnection of high-speed packet LAN in a virtual call environment; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.41-50; IEEE 80CH1542-0

[Popek, 1974]:

Gerald J. Popek: Protection structures; Computer, Vol. 7, No. 6, June 1974; ISSN 0018-9162

[Posa, 1980]:

John G. Posa & Bruce LeBoss: Intel takes aim at the '80s; Electronics, February 28, 1980, p.89-95; ISSN 0013-5070

[Prycker, 1983]:

Martin De Prycker: A performance comparison of three contemporary 16-bit microprocessors; IEEE Micro, April 1983, p.26-37; ISSN 0272-1732

[Queyssac, 1979]:

Daniel Queyssac: Projecting VLSI's impact on microprocessors; IEEE Spectrum, Vol. 16, No. 6, May 1979, p.38-41;

[Ranko, 1981]:

Raymond R. Ranko: Facing basic issues in office automation; Computer Networks, Vol. 5, no. 6, 1981, p.391-399; ISSN 0376-5075

[Rao, 1980]:

Ram Rao: Design and evaluation of distributed communication primitives; Proceedings of the ACM Pacific-80 conference, 1980, p.14-23;

[Rash, 1981]:

Bill rash: Getting started with the numeric data processor; Intel application note, February 1981; AP 113

[Rauch-Hindin, 1982]:

Wendy Rauch-Hindin: Managing networks with distributed database technology; Data Communications, August 1982, p.125-127; ISSN 0363-6399

[Rauch-Hindin, 1982.2]:

Wendy Rauch-Hindin: Burroughs times its entry into the IEEE local-net fray with token scheme; Data Communications, June 1982, p.61; ISSN-0363-6399

[Reghizzi, 1980]:

Stefano Crepsi-Reghizzi, Pierluigi Corti & Alberto Dapra: A survey of microprocessor languages; Computer, January 1980, p.48-66; ISSN 0018-9162

[Richer, 1981]:

Ira Richer, Marianne Steiner & Masama Sengoku: Office communication and the digital PBX; Computer Networks, Vol. 5, no. 6, 1981, p.411-422; ISSN 0376-5075

[Ripps, 1983]:

David L. Ripps: Multitasking OS manages a team of processors; Electronic Design, July 21, 1983, p.139-146; ISSN 0013-4872

[Ritchie, 1974]:

Dennis M. Ritchie & ken Thomson: The UNIX time-sharing system; Communications of the ACM, Vol. 17, No. 7, 1974, p.365-375;

[Rolander, 1982]:

Thomas A. Rolander, Randall Baird & John Wharton: Network software borrows design from microcomputer operating system; Data Communications, December 1983, p. 123-133; ISSN-0363-6399

[Rolander, 1982.2]:

Thomas A. Rolander: Microcomputer software meshes with local nets; Electronics, January 27, 1982, p.96-99; ISSN 0013-5070

[Roloff, 1980]:

Jeffrey J. Roloff: Managing memory to unloose the full power of microprocessors; Electronics, April 4, 1980, p.130-134; ISSN 0013-5070

[Rom, 1981]:

Raphael Rom & Fouad A. Tobagi: Message-based priority functions in local multiaccess communication systems; Computer Networks, 1981, p.273-286; ISSN 0376-5075

[Rumste, 1983]:

M. van Rumste: The iAPX 432, a next generation microprocessor; Microprocessing and Microprogramming, 1983, vol. 11, no. 2, p.69-106; ISSN 0165-6074

[Rushby, 1983]:

John Rushby & Brian Randell: A distributed secure system; Computer, July 1983, p.55-67; ISSN 0018-9162

[Ryan, 1981]:

Robert Ryan, George D. Marshall, Robert Beach & Steven R. Kerman: Intel local network architecture; IEEE Micro, November 1981, p.26-41; ISSN 0272-1732

[Ryer, 1982]:

Michael Ryer: Developing an ADA programming support environment; Mini-Micro Systems, September 1982, p.223-226; ISSN-0364-9342

[SDC, 1978]:

Sparekassernes Datacenter: Sparekassernes fremtidige terminaludstyr. Redegoerelse vedr. udbudsgrundlaget; Internal SDC material, July 1978;

[SDC, 1981]:

Sparekassernes Datacenter: Langtidsplan for SDC, 1982-1986; , September, 1981;

[Salwen, 1983]:

Howard C. Salwen: In praise of ring architecture for local area networks; Computer Design, March 1983, p.183-192; ISSN-0010-4566

[Sanford, 1983]:

Curtis Sanford & David Walden: Developing programs with UNIX; Mini-Micro Systems, May 1983, p.279-282; ISSN 0364-9342

[Sauer, 1980]:

Anton M. Sauer, Heinz G. Schwartzel & Hans. H. Bellm: A local microprocessor network for the office environment with an optical bus; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.17-20; IEEE 80CH1542-0

[Schanning, 1980]:

Brian P. Schanning, Susan A. Powers & John Kowalchuk: MEMO: Privacy and authentication for the automated office; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.21-30; IEEE 80CH1542-0

[Schell, 1983]:

Roger R. Schell: A security kernel for a multiprocessor microcomputer; Computer, July 1983, p.47-53; ISSN 0018-9162

[Schicker, 1981]:

Peter Schicker: The computer based mail environment - an overview; Computer Networks, Vol. 5, no. 6, 1981, p.435-443; ISSN 0376-5075

[Schindler, 1982]:

Sigram Schindler, Thomas Luckenbach & Michael Steinacker: X.21 as a universal digital service access interface; Computer Communications, Vol. 5, No. 6, December 1982, p. 298-307; ISSN 0140-3664

[Schneider, 1979]:

G. Michael Schneider: Computer network protocols: a hierarchical viewpoint; Computer, Vol.12, No. 9, September 1979; ISSN 0018-9162

[Schultz, 1983]:

Gaymond Schultz: An intelligent-workstation approach to office automation; Mini-Micro Systems, May 1983, p.193-200; ISSN 0364-9342

[Schwans, 1982]:

Karsten Schwans: Tailoring software for multiple processor systems; Computer Science Department, Carnegie-Mellon University, October 1, 1982; CMU CS-82-137

[Scott, 1983]:

Brian L. Scott: Voice recognition systems and strategies; Computer Design, January 1983, p.67-70; ISSN 0010-4566

[Serlin, 1983]:

Omri Serlin: Selecting superchips and supermicros; Datamation, February 1983, p. 177-3 - 177-6; ISSN 0011-6963

[Sharp, 1981]:

Steven A. Sharp: A building block I/O processor subsystem; Computer Design, December 1981, p.183-185; ISSN 0010-4566

[Shaw, 1974]:

Alan C. Shaw: The logical design of operating systems; Prentice-Hall, 1974; ISBN 0-13-540112-7

[Shima, 1978]:

Masathoshi Shima: Two versions of 16-bit chip span microprocessor, microcomputer needs; Electronics, 27 december 1978, p.81-88; ISSN 0013-5070

[Shima, 1979]:

Masatoshi Shima: Demystifying microprocessor design; IEEE Spectrum, Vol. 16, No. 7, 1979, p.22-30;

[Shoch, 1982]:

John F. Shoch, Yogen K. Dalal, David D. Redell & Ronald C. Crane: Evolution of the Ethernet local computer network; Computer, August 1982, p.10-27; ISSN 0018-9162

[Shoemaker, 1983]:

Kenneth Shoemaker: Microprocessor chip integrates a host of peripheral functions to simply systems; Electronics, May 5, 1983, p.139-145; ISSN 0013-5070

[Shonim, 1981]:

Jacob Slonim et al.: NDX-100: an electronic filing machine for the office of the future; Computer, May 1981, p.24-36; ISSN 0018-9162

[Shrehlo, 1982]:

Kevin B. Shrehlo: ANSI graphics standards coalesce around international kernel; Mini-Micro Systems, November 1982, p.175-186; ISSN 0364-9342

[Sims, 1982]:

John Sims: Building I/O into the VME bus; Systems International, December, 1982, p.48-50; ISSN 0309-1171

[Smith, 1977]:

Alan Jay Smith: Multiprocessor memory organization and memory interference; Communications of the ACM, Vol. 20, No. 10, October 1977;

[Smura, 1981]:

Edwin J. Smura: Structures for advanced information systems; Computer, September 1981, p.60-73; ISSN 0018-9162

[Snook, 1978]:

Tod Snook, Charlie Bass, Janet Roberts, Armen Nehapetian & Mike Fay: Report on the programming language PLZ/SYS; Springer Verlag, 1978; ISBN 0-387-90374-7

[Solomon, 1979]:

Marvin H. Solomon & Raphael A. Finkel: The Roscoe distributed operating system; Proceedings of the seventh symposium on operating systems principles, ACM No. 534790; 1979, p.108-114; ISBN 0-89791-009-5

[Soltis, 1981]:

Frank G. Soltis: Design of a small business data processing system; Computer, September 1981, p.77-93; ISSN 0018-9162

[Spector, 1980]:

Alfred Z. Spector: Extending LAN interfaces to provide more efficient interprocessor comm. fac.; Proceedings of the ACM Pacific-80 conference, 1980, p.6-13;

[Stahlman, 1982]:

Mark Stahlman: Inside Wangs local net architecture; Data Communications, January 1982, p.85-90; ISSN 0363-6399

[Standards, 1982]:

Conference Report: The new ISO standards for communications and office automation; Computer Networks, Vol 6. No 4., 1982, p.291-298; ISSN 0376-5075

[Stankovic, 1982]:

John A. Stankovic: Software communication mechanisms: Procedure calls versus Messages; Computer, April 1982, p.19-25; ISSN 0018-9162

[Stenning, 1981]:

Vic Stenning et al.: The ADA environment: a perspective; Computer, June 1981, p.26-36; ISSN 0018-9162

[Stevenson, 1981]:

David Stevenson: A proposed standard for binary floating-point arithmetic; Computer, March 1981, p.51-62; ISSN 0018-9162

[Stewart, 1983]:

Robert G. Stewart: The good news - P696, P796 now IEEE standards; IEEE Micro, February 1983, p.69-71; ISSN 0272-1732

[Stieglitz, 1981]:

Mark Stieglitz: Local network access tradeoffs; Computer Design, October 1981, p.163-168; ISSN 0010-4566

[Stieglitz, 1982]:

Mark Stieglitz: Token-access controller minimizes network complexity; Mini-Micro Systems, March 1982, p.171-180; ISSN 0364-9342

[Stockton, 1982]:

John F. Stockton: A virtual breakthrough for micros; Computer Design, August 1982, p.153-162; ISSN 0010-4566

[Stockton, 1983]:

John F. Stockton: Growth of processor family boots systems options; Computer Design, February 1983, p.71-80; ISSN 0010-4566

[Strehlo, 1982]:

Kevin Strehlo: Support for graphics standards swells to 15 vendors; Mini-Micro Systems, September 1982, p.47; ISSN-0364-9342

[Stuck, 1983]:

Bart W. Stuck: Calculating the maximum mean data rate in local area networks; Computer, May 1983, p.72-76; ISSN 0018-9162

[Stuck, 1983.2]:

Bart Stuck: Which local net bus access is most sensitive to traffic congestion; Data Communications, January 1983, p.107-120; ISSN 0363-6399

[Sugarman, 1979]:

Robert Sugarman: Computers: our 'microuniverse' expands; IEEE Spectrum, Vol. 16, No. 1, 1979, p.32-37;

[Tannenbaum, 1981]:

Andrew S. Tannenbaum: Network protocols; Computing Surveys, Vol. 13, No. 4, December 1981, p. 453-489;

[Taylor, 1982]:

Richard Taylor & Pete Wilson: Process-oriented language meets demands of distributed processing; Electronics, November 30, 1982, p.89-95; ISSN 0013-5070

[Teger, 1983]:

Sandra L. Teger: Factors impacting the evolution of office automation; Proceedings of the IEEE, Vol. 71, no.4, 1983, p.503-511; ISSN 0018-9219

[Thomae, 1980]:

Irving H. Thomae: 8-bit microprocessor harbors 16-bit performance; Electronics, January 3, 1980, p.163-167; ISSN 0013-5070

[Thurber, 1982]:

Kenneth J. Thurber & Harvey A. Freedman: Many makers unloose a flood of local nets; Electronics, January 27, 1982, p.90-95; ISSN 0013-5070

[Tilborg, 1983]:

Andre M. van Tilborg & Larry D. Wittie: Operating systems for the Micronet network computer; IEEE Micro, April 1983, p.38-47; ISSN 0272-1732

[Timmons, 1981]:

Michael L. Timmons: Distributed communication architecture forms framework for network design; Computer Design, February 1981, p.121-125; ISSN 0010-4566

[Tobagi, 1980]:

Fouad A. Tobagi & V. Bruce Hunt: Performance analysis of Carrier Sense Muliple Access with Collision Detection; Computer Networks, Vol. 4, No. 5, 1980, p.245-259; ISSN 0376-5075

[Toong, 1980]:

Hoo-min D. Toong et al: Parallel processing capabilities of advanced 16-bit microprocessors; Proceedings of the ACM Pacific-80 conference, 1980, p.130-141;

[Treleaven, 1979]:

Philip C. Treleaven: Exploiting program concurrency in computing systems; Computer, Vol.12, No.1, January 1979; ISSN 0018-9162

[Treleaven, 1982]:

Philip C. Treleaven: VLSI processor architectures; Computer, June 1982, p.33-45; ISSN 0018-9162

[Uhlig, 1981]:

Ronald P. Uhlig, David J. Farber & James H. Bair: The office of the future; North-Holland Publishing Company, 1981; ISBN 0-444-85336-7

[Unger, 1982]:

Brian Unger et al.: An OASIS simulation of the ZNET microcomputer network; IEEE Micro, August 1982, p.70-84; ISSN 0272-1732

[Unger, 1982.2]:

Brian W. Unger & Don S. Bidulock: The design and simulation of a multi-computer network message processor; Computer Networks, Vol 6. No 4., 1982, p.263-277; ISSN 0376-5075

[Walden, 1979]:

David C. Walden & Alexander A. McKenzie: The evolution of host-to-host protocol technology; Computer, Vol. 12, No.9, 1979; ISSN 0018-9162

[Waller, 1983]:

Larry Waller: ADA compiler yields source code in C, operates on both 16- and 32-bit computers; Electronics, July 14, 1983, p.49-50; ISSN 0013-5070

[Ward, 1980]:

S. Ward: The NU personal computer: NuBus specification; IEEE P896 Design Course Documentation, November 9, 1980;

[Warman, 1983]:

Ernest Warman: Graphics - a European Standard; Systems International, April 1983, p.51 - 53;

[Warner, 1980]:

Clifford Warner: Connecting local networks to long haul networks: issues in protocol design; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.71-76; IEEE 80CH1542-0

[Warren, 1983]:

Carl Warren: Rigid-disk drives: capacity, performance mount as size shrinks; Electronic Design, April 28, 1983, p. 139-150;

[Warren, 1983.2]:

Carl Warren: Computer system design: Supermicrocomputers; Electronic Design, May 26, 1983, p.97-106; ISSN 0013-4872

[Warren, 1983.3]:

Carl Warren: UNIX system 5 goes generic in a standard environment; Electronic Design, July 7, 1983, p.51-52; ISSN 0013-4872

[Watson, 1979]:

Bruce Watson: The traffic dependent performance of a prioritized, CSMA broadcast network; Computer Networks, December 1979, p.427-434; ISSN 0376-5075

[Watson, 1980]:

W. Bruce Watson: Simulation study of the configuration dependent performance of a CMSA network; Proceedings of the 5th conference on local computer networks, October 6-7, 1980, p.95-103; IEEE 80CH1542-0

[Wecker, 1979]:

Stuart Wecker: Computer network architectures; Computer, Vol. 12, No.9, 1979; ISSN 0018-9162

[Weiss, 1982]:

Paul F. Weiss: The mini-interface: Its prospects and problems; Data Communications, June 1982, p.133-145; ISSN-0363-6399

[Weissberger, 1983]:

Allan J. Weissberger: Bit oriented data link controls; Computer Design, March 1983, p.195-206; ISSN-0010-4566

[Weissman, 1982]:

Esti Weismann, Louis H. Phillips & John Wallace : Bank validates transactions with home-grown data authentication; Data Communications, July 1982, p.117-120; ISSN 0363-6399

[Weitzman, 1980]:

Cay Weitzman: Distributed Micro/Minicomputer systems; Prentice-Hall, 1980; ISBN 0-13-216481-7

[Whitcomb, 1982]:

Roger Whitcomb: On-chip memory management comes to 8-bit microprocessor; Electronic Design, October 14, 1982; ISSN 0272-1732

[Whitworth, 1979]:

I. R. Whitworth: Designing flexibility into memory systems; Microprocessors and Microsystems, Vol. 3, No. 10, December 1979; ISSN 0141-9331

[Whitworth, 1980]:

Ian Whitworth: Developments in 16-bit microprocessors; Microprocessors and Microsystems, January 1980, p.19-22; ISSN 0141-9331

[Wiley, 1982]:

Joe M. Wiley: Barriers to integrating voice and data; Data Communications, March 1983, p. 109-113; ISSN 0363-6399

[Williams, 1983]:

Gregg Williams: The LISA computer system; Byte, February 1983, p.33-50; ISSN 0360-5280

[Wilson, 1983]:

Pete Wilson: Programming system builds multiprocessor software; Electronic Design, July 21, 1983, p.129-134; ISSN 0013-4872

[Wirth, 1977]:

Nicklaus Wirth: Towards a discipline of real-time programming; Communications of the ACM, Vol. 20, No. 8, August 1977;

[Witten, 1983]:

Ian H. Witten: Welcome to the standards jungle; Byte, February 1983, p.146-178; ISSN 0360-5280

[Wolfe, 1981]:

Martin I. Wolfe et al.: The ADA language system; Computer, June 1981, p.37-45; ISSN 0018-9162

[Yoram, 1983]:

Cedar Yoram & Meir Ben-Nun: Improving computational throughput; Computer Design, March 1983, p.143-146; ISSN-0010-4566

[Zahorka, 1981]:

F. Zahorka: A possible handshake scheme on a single line; P896 Data Transfer Protocol proposal, May 6, 1981;

[Zahorka, 1981.2]:

F. Zahorka: A P896 level without the parallel bus; P896 Serial only level proposal, May 6, 1981;

[Zbits, 1982]:

<unknown>: Figures from Dataquest show the Z80 leading the 8-bit microprocessor race; Zilog Zbit Newsletter, Issue 3, 1982;

[Zeigler, 1981]:

Stephen Zeigler et al.: Ada for the Intel 432 microcomputer; Computer, June 1981, p.47-56; ISSN 0018-9162

[Zilog, 1979]:

Tue Bertelsen, Anders Nielsen & Peter Laessoe: Macros for interfacing assembly language modules with PLZ/SYS; Zilog Application Note No. 102G, Software Series, September 1979;

[Zilog, 1983]:

Zilog technical manual: Z80000 CPU, product specification; 00-2071-01, preliminary version for limited distribution, July 1983;

[Zimmermann, 1980]:

Hubert Zimmermann: OSI reference model - the ISO model of architecture for open systems intercon.; IEEE Transactions on Communication, Vol. Com-28, No.4, 1980; ISSN 0090-6778

[Zloof, 1981]:

Moshe M. Zloof: QBE/OBE: a language for office and business automation; Computer, May 1981, p.13-22; ISSN 0018-9162

[Zoccoli, 1981]:

Mario P. Zoccoli & Arthur C. Sanderson: RAPID bus multiprocessor system; Computer Design, November 1981, p.189-200; ISSN 0010-4566

TITLE CROSS-REFERENCE OF BIBLIOGRAPHY

A distributed secure system --> **[Rushby, 1983]**

A history of microprocessor development at Intel --> **[Noyce, 1981]**

A local microprocessor network for the office environment with an optical bus --> **[Sauer, 1980]**

A local network interface on P896 --> **[Beuchat, 1981]**

A multiprocessor minicomputer designed for UNIX --> **[Myers, 1982.2]**

A new approach for local networks --> **[Canning, 1981]**

A P896 level without the parallel bus --> **[Zahorka, 1981.2]**

A performance comparison of three contemporary 16-bit microprocessors --> **[Prycker, 1983]**

A plethora of projects in the U.S. try data-flow and other architectures --> **[Electronics, 1983]**

A possible handshake scheme on a single line --> **[Zahorka, 1981]**

A primer for evaluating and purchasing today's local networks --> **[Gibson, 1982.2]**

A programming language for networks --> **[Maine, 1981]**

A proposed floppy disk format standard --> **[Card, 1983.2]**

A proposed standard backplane bus specicication for advanced microcomputer sys. --> **[I896, 1982]**

A proposed standard for binary floating-point arithmetic --> **[Stevenson, 1981]**

A protocol for a ring network of heterogeneous computers --> **[Beauvais, 1980]**

A security kernel for a multiprocessor microcomputer --> **[Schell, 1983]**

A serial interprocessor link for multiprocessor management in the P896 bus --> **[Kirrmann, 1981]**

A small-scale operating system foundation for microprocessor applications --> **[Kahn, 1978]**

A summary of the Fastbus protocol for P896 --> **[Gustavson, 1981.2]**

A survey of interconnection networks --> **[Feng, 1981]**

A survey of microprocessor languages --> **[Reghizzi, 1980]**

A terminal-oriented communication system --> **[Heckel, 1977]**

A test technique for microprocessor based machines --> **[DelCorso, 1979]**

A UNIX-based local computer network with load balancing --> **[Hwang, 1982]**

A virtual breakthrough for micros --> **[Stockton, 1982]**

ADA compiler yields source code in C, operates on both 16- and 32-bit computers --> **[Waller, 1983]**

Ada for the Intel 432 microcomputer --> **[Zeigler, 1981]**

Adding another layer to the ISO net architecture reduces cost --> **[Larson, 1983]**

Advanced parallel architectures get attention as way to faster computing --> **[Manuel, 1983.2]**

Amended P896 supervisory handshake proposal --> **[MacKenna, 1981]**

An architectural comparison of 32-bit microprocessors --> **[Gupta, 1983]**

An integrated broadband local network architecture --> **[Marglaris, 1980]**

An intelligent-workstation approach to office automation --> **[Schultz, 1983]**

An introduction to the Fastbus --> **[Gustavson, 1979]**

An OASIS simulation of the ZNET microcomputer network --> **[Unger, 1982]**

An object oriented operating system for microcomputers --> **[Collins, 1982]**

Analysis of proposals for the floating-point standard --> **[Cody, 1981]**

Anatomy of a microcomputer protocol --> **[Pearson, 1983]**

ANSI graphics standards coalesce around international kernel --> **[Shrehlo, 1982]**

Applicability of the Fastbus standard to distributed control --> **[Deiss, 1981]**

Applications of the proposed IEEE 754 standard for floating-point arithmetic --> **[Hough, 1981]**

Architecture of a new microprocessor --> **[Peuto, 1979]**

Architecture of distributed computer systems --> **[Bochmann, 1979]**

Architecture, communication procedures and performance eval. of the U* system --> **[Marsan, 1979]**

At-net - A new UNIX based transparent networking system --> **[Kayaler, 1983]**

Backend database systems --> **[Maryanski, 1980]**

Backplane bus standards - why we need them ..... --> **[Borrill, 1983]**

Backplane transmission line and crosstalk behaviour, Viewgraphs of presentation --> **[Borrill, 1981]**

Bank validates transactions with home-grown data authentication --> **[Weissman, 1982]**

Barriers to integrating voice and data --> **[Wiley, 1982]**

Berkeley 4.2 gives UNIX operating system network support --> **[Joy, 1983]**

Bit oriented data link controls --> **[Weissberger, 1983]**

Bridging the gap (16-bit operating systems) --> **[McLenning, 1983]**

Bringing computing to the people: the broadening challenge --> **[Branscomb, 1982]**

Broadband at base of Wang's far-reaching local network --> **[Datacomm, 1981]**

Broadband Network Design: issues and answers --> **[Cooper, 1983]**

Building I/O into the VME bus --> **[Sims, 1982]**

Building local area networks to Ethernet specification --> **[Hutchison, 1982]**

Built-in I/O support beefs up 16-bit uP --> **[Folkes, 1983]**

Burroughs times its entry into the IEEE local-net fray with token scheme --> **[Rauch-Hindin, 1982.2]**

Bus design choices for efficiency and ease of test --> **[DelCorso, 1981.2]**

Buses, the skeleton of computer structures --> **[Levy, 1978]**

Calculating the maximum mean data rate in local area networks --> **[Stuck, 1983]**

Chips come to aid of embedded systems --> **[Evanczuk, 1983.2]**

Coming of Age: A long-awaited standard for heterogeneous nets --> **[Folts, 1981]**

Comments on supervisors versus snoops --> **[Gustavson, 1981]**

Committees push to standardize disk I/O --> **[Parker, 1983]**

Communication protocols for integrated office systems --> **[Naffah, 1981]**

Comparing C compilers for CP/M-86 --> **[Houston, 1983]**

Comparing features aids selecting broadband local net --> **[Gibson, 1982]**

Comparing the CBX to the local network - and the winner is? --> **[Pfister, 1982]**

Comparison of bus arbitrations --> **[Lyman, 1981]**

Components and software for the iAPX 286 --> **[Intel, 1983.2]**

Components of a network operating system --> **[Donnelley, 1979]**

Computer graphics: the need for graphics design, part one --> **[Myers, 1981]**

Computer network architectures --> **[Wecker, 1979]**

Computer network protocols: a hierarchical viewpoint --> **[Schneider, 1979]**

Computer networks and their protocols --> **[Davies, 1979]**

Computer programming as an art --> **[Knuth, 1974]**

Computer system design: Supermicrocomputers --> **[Warren, 1983.2]**

Computer, Communications and Man: The integration of C&C. with man as an axis --> **[Kobayashi, 1981]**

Computers: our 'microuniverse' expands --> **[Sugarman, 1979]**

Concepts and notations for concurrent programming --> **[Andrews, 1983]**

Concurrency: key to 16-bit operating system efficiency --> **[Horovitz, 1982]**

Connecting local networks to long haul networks: issues in protocol design --> **[Warner, 1980]**

Controller chip shares tasks in buffer, net management --> **[Coleman, 1982.3]**

Coprocessing to ease the graphics burden --> **[Gordon, 1982]**

CP/M is now tuned in to C --> **[Black, 1983]**

Facing basic issues in office automation --> **[Ranko, 1981]**

Factors impacting the evolution of office automation --> **[Teger, 1983]**

Fastbus introduction and demonstration --> **[Logg, 1981]**

Figures from Dataquest show the Z80 leading the 8-bit microprocessor race --> **[Zbits, 1982]**

First complete ADA compiler runs on a micro --> **[Carlson, 1982]**

Five C compilers for CP/M-80 --> **[Kern, 1983]**

Flexible controller mates with popular Winchester drives --> **[Jaworsk, 1983]**

Getting started with the numeric data processor --> **[Rash, 1981]**

Graphical Kernel System (GKS) - functional description --> **[GKS, 1982]**

Graphics - a European Standard --> **[Warman, 1983]**

Growth of processor family boots systems options --> **[Stockton, 1983]**

Hardware comes to the aid of modular high-level languages --> **[Ashkenazi, 1981]**

Hardware interface links RS-232 ports to Ethernet networks --> **[Announce, 1983.2]**

High level programming for distributed computing --> **[Feldman, 1979]**

High level protocols, standards, and the OSI reference model --> **[Mier, 1982]**

High performance microprocessor opens new doors for system designers --> **[Intel, 1982]**

HP's PC Strategy --> **[Norman, 1983]**

IEEE P896 - the Futurebus project --> **[Borrill, 1983.2]**

IEEE Project 802 - a status report --> **[IEEE, 1982]**

IEEE-P896 electrical considerations --> **[Adams, 1981]**

Implementation of task management on the parallel bus, general arbiter operat. --> **[Borrill, 1981.4]**

Implementing distributed processes in centralized systems --> **[Bertelsen, 1979.2]**

Implementing distributed processes in centralized systems --> **[Bertelsen, 1979.3]**

Implementing the UCSD p-System --> **[Overgaard, 1982]**

Improving computational throughput --> **[Yoram, 1983]**

Improving office productivity: a technology perspective --> **[Lochovsky, 1983]**

In a sudden reversal, IEEE embraces Ethernet --> **[Datacomm, 1982.2]**

In praise of ring architecture for local area networks --> **[Salwen, 1983]**

In your future: Local computer networks --> **[Canning, 1980]**

Information management strategies in the 1980-s --> **[Becker, 1980]**

Information management trends in office automation --> **[King, 1983]**

Innovations in heterogeneous and homogeneous distributed-function architectures --> **[Joseph, 1974]**

Inside Wangs local net architecture --> **[Stahlman, 1982]**

Integrated office-automation systems --> **[Murphy, 1983]**

Intel local network architecture --> **[Ryan, 1981]**

Intel microprocessors - 8008 to 8086 --> **[Morse, 1980]**

Intel takes aim at the '80s --> **[Posa, 1980]**

Intel's first integrated microsystem is based on VLSI standards --> **[Intel, 1982.2]**

Interface Intelligence --> **[Barker, 1983]**

Interprocess communications facilities for network operating systems --> **[Akkoyunlu, 1974]**

Introduction to local area networks --> **[Digital, 1982]**

Introduction to microprocessor programming using PLZ --> **[Conway, 1979]**

Inversion of the memory hierarchy --> **[Goldstein, 1980]**

Japan is busy trying to make manufacturable data-flow computers --> **[Electronics, 1983.3]**

L896 - bus interface design for 68000 uP systems --> **[Hersch, 1981]**

Langtidsplan for SDC, 1982-1986 --> **[SDC, 1981]**

Let operating systems aid in component designs --> **[Heider, 1982]**

Level 6 Inter System Link overview --> **[Conway, 1977]**

Link-controller IC combines versatility and flexibility --> **[Goodrich, 1982]**

Local area nets: a pair of standards --> **[Graube, 1982]**

Local area networks --> **[Intel, 1981.2]**

Local net user relates trials, tribulations --> **[Gibson, 1983]**

Local network access tradeoffs --> **[Stieglitz, 1981]**

Local network gives new flexibility to distributed processing --> **[Bass, 1980.2]**

Local network standards: no utopia --> **[Dahod, 1983]**

Local network technology and the lessons of history --> **[McQuillan, 1980]**

Local networks' consensus: high speed --> **[Hsi, 1980]**

Local-net standardization gains --> **[Hindin, 1983]**

LSI chips ease design of hard-disk controller --> **[Devlin, 1983]**

LSI peripherals: uP's helping hands are strong and getting stronger --> **[Bursky, 1979]**

M3BUS - Modular Multi Micro Bus - preliminary specification --> **[DelCorso, 1981.3]**

Macros for interfacing assembly language modules with PLZ/SYS --> **[Zilog, 1979]**

Major standardization issues of the proposed IEEE 796 bus - Multibus --> **[Boberg, 1983]**

Making a case for token passing in local networks --> **[Miller, 1982]**

Managing memory to unloose the full power of microprocessors --> **[Roloff, 1980]**

Managing networks with distributed database technology --> **[Rauch-Hindin, 1982]**

Many makers unloose a flood of local nets --> **[Thurber, 1982]**

McBusiness - Systemarkitekturen for Sparekassernes Erhvervsterminal --> **[Bertelsen, 1981]**

McTan er en ny mikrodatamat der er udviklet i SDC --> **[Hansen, 1980]**

McTan er i luften --> **[Bertelsen, 1982.2]**

McTan loop interface utility. Release 5.5. User's guide --> **[Bertelsen, 1981.5]**

McTan X.21 PDN/Loop line interface - Notat vedr. TP2 access til SDC online-sys. --> **[Bertelsen, 1981.3]**

McTan X.21 projekt - Udredning vedr. etablering af en datatransmissionsvej .... --> **[Bertelsen, 1981.4]**

Medusa: an experiment in distributed operating system structure --> **[Ousterhout, 1980]**

MegaFrame aims at mainframe performance --> **[Huie, 1983]**

MEMO: Privacy and authentication for the automated office --> **[Schanning, 1980]**

Merging Performance and Cost-benefit Analysis in Computer System Evaluation --> **[Bucci, 1982]**

Message-based priority functions in local multiaccess communication systems --> **[Rom, 1981]**

Microcomputer architecture - a survey report --> **[Crook, 1980]**

Microcomputer market soars on all fronts --> **[Blundell, 1982]**

Microcomputer software meshes with local nets --> **[Rolander, 1982.2]**

Microfloppies battle for preeminence --> **[Abraham, 1983]**

Microfloppy-disk drive adopts 3-1/2 inch diskette endorsed by industry --> **[Jarrett, 1983]**

Microprocessor bus evaluation --> **[EDISG, 1981]**

Microprocessor bus structures and standards --> **[Borill, 1981]**

Microprocessor chip integrates a host of peripheral functions to simply systems --> **[Shoemaker, 1983]**

Microprocessor systems - architecture and engineering --> **[Alexandridris, 1980]**

Micros expand the lifetime of old protocols --> **[Bertelsen, 1979]**

Modelling and analysis of distributed software systems --> **[Kumar, 1979]**

Modular protocols improve industrial network control --> **[Ceferin, 1983]**

Monitors: an operating system structuring concept --> **[Hoare, 1974]**

Monobus Specification --> **[OL, 1977]**

Multi-microprocessors: an overview and working example --> **[Fuller, 1978]**

Multiple microprocessor systems: What, Why, and When --> **[Fathi, 1983]**

Multiprocessor architecture ensures fault-tolerant transaction processing --> **[Inselberg, 1983]**

Multiprocessor architecture tunes in to transaction processing --> **[Cohen, 1983]**

Multiprocessor architectures for concurrent programs --> **[Hansen, 1979]**

Multiprocessor memory organization and memory interference --> **[Smith, 1977]**

Multiprocessors and parallel processing --> **[Enslow, 1974]**

Multitasking OS manages a team of processors --> **[Ripps, 1983]**

Multiuser microprocessor systems get a data-base manager --> **[Lowenthal, 1982]**

Naar Data skal kommunikeres - og kommunikation skal programmeres! --> **[Bertelsen, 1982]**

NAPLPS standard graphics and the microcomputer --> **[Lax, 1983]**

NAPLPS: a new standard for text and graphics --> **[Flemming, 1983]**

Native Code teams with pseudocode to hasten emulation --> **[Koehler, 1982]**

NCR 9300 32-bit mainframe processor... A new generation --> **[NCR, 1983.2]**

NDX-100: an electronic filing machine for the office of the future --> **[Shonim, 1981]**

Network developments taking different routes --> **[Allan, 1983]**

Network protocols --> **[Tannenbaum, 1981]**

Network software borrows design from microcomputer operating system --> **[Rolander, 1982]**

Network, Heal Thyself: a diagnostic primer --> **[Eimers, 1982]**

New options from big chips --> **[McKevitt, 1979]**

Nine C compilers for the IBM PC --> **[Phraner, 1983]**

Nuova Linea Sistemi: Il bus di sistima --> **[Bonci, 1980]**

OASIS, PICK and UNIX: OS for OEMs --> **[Guyod, 1983]**

Office communication and the digital PBX --> **[Richer, 1981]**

Office information systems and computer science --> **[Ellis, 1980]**

On the design of a distributed operating system using a HL distributed P.L. --> **[Arora, 1982]**

On the duality of operating system structures --> **[Lauer, 1979]**

On-chip memory management comes to 8-bit microprocessor --> **[Whitcomb, 1982]**

Opening salvos launched in 256-K battle --> **[Beresford, 1983]**

Operating system principles --> **[Hansen, 1973]**

Operating system structures for polymorphic hardware --> **[Lorin, 1980]**

Operating Systems --> **[Madnick, 1974]**

Operating systems --> **[Anderson, 1981]**

Operating systems for the Micronet network computer --> **[Tilborg, 1983]**

Optical data storage technology status and prospects --> **[Bell, 1983]**

Organization and policy issues in the implementation of distributed systems --> **[Felix, 1980]**

OSI reference model - the ISO model of architecture for open systems intercon. --> **[Zimmermann, 1980]**

P696/S100 - a bus which supports a wide range of 8- and 16-bit processors --> **[Garetz, 1983]**

P896 electrical & mechanical requirements. Detailed specifications --> **[Borrill, 1981.2]**

P896 Specification Structure --> **[Lyman, 1981.3]**

P896-PLB-17 electrical and pinout considerations --> **[Borrill, 1980]**

P986 bus arbitration proposal --> **[Lyman, 1981.2]**

Packet switching interconnection networks for modular systems --> **[Dias, 1983]**

Parallel processing capabilities of advanced 16-bit microprocessors --> **[Toong, 1980]**

Parallel processor systems, technologies and applications --> **[Hobbs, 1970]**

Pascal* : A Pascal based systems programming language --> **[Hennessy, 1980]**

Patent snafu hits 802 --> **[Mier, 1982.2]**

Paving the way for universal document interchange --> **[Jones, 1982]**

Performance analysis of Carrier Sense Muliple Access with Collision Detection --> **[Tobagi, 1980]**

Perspectives on the evolution of commercial local networking --> **[Gordon, 1980]**

Pick jockeys for position in standard operating-system race --> **[Lewis, 1982]**

Pipelining and new OS boost mini to 8 MIPS --> **[Basart, 1982]**

Plan now for work stations --> **[Canning, 1983]**

Planning your distributed systems --> **[Canning, 1983.2]**

Portable operating systems fight for 16-bit machines --> **[Freedman, 1983]**

Preliminary proposal for TP2 user data formatting --> **[Bertelsen, 1982.3]**

Preliminary protocol architecture for the Olivetti local network system --> **[Hunt, 1979]**

Primitives for distributed computing --> **[Liskov, 1979]**

Process-oriented language meets demands of distributed processing --> **[Taylor, 1982]**

Programming issues raised by a multiprocessor --> **[Jones, 1978]**

Programming system builds multiprocessor software --> **[Wilson, 1983]**

Projecting VLSI's impact on microprocessors --> **[Queyssac, 1979]**

Proposal for Fastbus style protocol --> **[Borrill, 1980.2]**

Proposals for the P896 protocol --> **[Kirrmann, 1981.2]**

Propose microcomputer system 796 bus standard, IEEE Task P796/D2 --> **[Boberg, 1980]**

Proposed standard specification for P896/D 3.4 advanced mc. system backplane --> **[P896, 1981]**

Proposed standard specification for P896/D 3.4.1 advanced uP system backplane --> **[P896, 1981.2]**

Proposed standard specification for P896/D 4. advanced mc. system backplane --> **[P896, 1981.3]**

Proposta preliminare, TOMP-80 --> **[DelCorso, 1980]**

Protection structures --> **[Popek, 1974]**

Putting Ethernet onboard the MULTIBUS --> **[Harakal, 1981]**

QBE/OBE: a language for office and business automation --> **[Zloof, 1981]**

Racal-Milgo announces token-passing network --> **[Jones, 1982.2]**

RAM dons lithium-cell hat for convenient nonvolatility --> **[Bolan, 1983]**

RAPID bus multiprocessor system --> **[Zoccoli, 1981]**

Real time operating system approaches for multiprocessors --> **[Muehlemann, 1981]**

Reality and the proposed OSI standard --> **[Moldow, 1981]**

Realizing graphics standards for microcomputers --> **[Langhorst, 1983]**

Reflections on an operating system design --> **[Lampson, 1976]**

Reliability issues for fully replicated distributed databases --> **[Molina, 1982]**

Remote procedure call --> **[Nelson, 1981]**

Report of the EF-57 study tour covering visits to major companies in USA --> **[Bertelsen, 1981.2]**

Report on the programming language PLZ/SYS --> **[Snook, 1978]**

Resource access control in a network operating system --> **[Donnelley, 1980]**

Resource requirements definition in distributed system design --> **[Palmer, 1980]**

Results of arbiter test. A preliminary report to the P896 committee --> **[Borrill, 1981.3]**

Review of the system performance of the TP2 system --> **[GSG, 1982]**

RIG, an architecture for distributed systems --> **[Lantz, 1980]**

Rigid-disk drives: capacity, performance mount as size shrinks --> **[Warren, 1983]**

Ringnet: a packet switched local network with decentralized control --> **[Gordon, 1979]**

S1000 service manual --> **[OL, 1982.5]**

Sample of instruction execution time (S1000) --> **[OL, 1979]**

SDC multitask scheduling facilities - User's guide --> **[Bertelsen, 1980.3]**

SDC System, APMAN - Concepts and facilities --> **[OL, 1981.4]**

SDC System, Application environment operating procedures --> **[OL, 1982]**

SDC System, SBPL - User Guide --> **[OL, 1981.5]**

SDC System, TP-monitor - Concepts and facilities --> **[OL, 1982.2]**

SDC System: System Overview and Network Components --> **[OL, 1982.4]**

SDC's views on computer technology and its impact on future branchoffice system --> **[Bertelsen, 1980.5]**

Selecting a programming language, compiler and support enviroment: method & exp --> **[Anderson, 1982]**

Selecting superchips and supermicros --> **[Serlin, 1983]**

Self-stabilizing systems in spite of distributed control --> **[Dijkstra, 1974]**

Self-testing computers --> **[Clary, 1979]**

Siliconizing the local area network --> **[Coleman, 1982.2]**

Simulation study of the configuration dependent performance of a CMSA network --> **[Watson, 1980]**

SKALP, Skeleton architecture for fault-tolerant distributed computing --> **[Courtois, 1981]**

Software communication mechanisms: Procedure calls versus Messages --> **[Stankovic, 1982]**

Software considerations in distributed architectures --> **[Farber, 1974]**

Software development and debug aids for the U* multimicroprocessor system --> **[Conte, 1980]**

Software for Ethernet LANs reaches transport level --> **[Announce, 1983]**

Software Transportability --> **[Manison, 1983]**

Software-in-Silicon boots system performance, cuts programming time --> **[Lettieri, 1982]**

Sparekassernes fremtidige terminaludstyr. Redegoerelse vedr. udbudsgrundlaget --> **[SDC, 1978]**

Special processor handles connection to local networks --> **[Manuel, 1983]**

Squeezing the most out of the 68000 --> **[Isaak, 1982]**

Standard bus for 8-bit microprocessor systems --> **[Elsmore, 1983]**

Standard specification for microcomputer system backplanes --> **[P896, 1980]**

Status report on new standards for DTE/DCE interface protocols --> **[Folts, 1979]**

Status report on the P896 backplane bus --> **[Allison, 1981]**

Strategies and concepts for linking today's personal computers --> **[Held, 1983]**

Structures for advanced information systems --> **[Smura, 1981]**

Structuring principles of the communication architecture of open systems --> **[Burkhardt, 1981]**

Supervision of data transfers on P896 --> **[Dumoulin, 1981]**

Support for graphics standards swells to 15 vendors --> **[Strehlo, 1982]**

Sure and steady 4-inch drive keeps weight and cost down. How? Simplification --> **[Herald, 1983]**

Survivable systems --> **[Highleyman, 1980]**

System 1000 - Theory of operation --> **[OL, 1981.6]**

System 1000, System Summary --> **[OL, 1981.2]**

System-level functions enchance controller IC --> **[Beach, 1982]**

Tailor the winchester to the system --> **[Jacob, 1983]**

Tailoring software for multiple processor systems --> **[Schwans, 1982]**

Taking a new look at matrix-switched systems --> **[Levy, 1982]**

TC 1808, TCOSMOS - concepts and facilities --> **[OL, 1981.3]**

TC 1808, Theory of operation --> **[OL, 1981]**

Technical aspects of data communication --> **[McNamara, 1978]**

Technologies for local area computer networks --> **[Cotton, 1980]**

The ADA environment: a perspective --> **[Stenning, 1981]**

The ADA language system --> **[Wolfe, 1981]**

The AmZ8000 BLOC-BUS Interface --> **[AMD, 1980]**

The architecture and hardware implementation of a prototype MICRONET --> **[Nickens, 1980]**

The architecture of concurrent programs --> **[Hansen, 1977]**

The best available technologies for computer security --> **[Landwehr, 1983]**

The Cambridge ring is still making the rounds --> **[Datacomm, 1981.2]**

The common sense of object oriented languages --> **[Freedman, 1983.2]**

The complete VLSI LAN solution --> **[Intel, 1982.3]**

The computer based mail environment - an overview --> **[Schicker, 1981]**

The Convoy phenomenon --> **[Blasgen, 1979]**

The design and simulation of a multi-computer network message processor --> **[Unger, 1982.2]**

The electronic office and organizational behavior - measuring office activities --> **[Conrath, 1981]**

The Ethernet: A local area network. Data link layer and physical layer specs. --> **[Ether, 1980]**

The evolution of host-to-host protocol technology --> **[Walden, 1979]**

The good news - P696, P796 now IEEE standards --> **[Stewart, 1983]**

The hardware/software cost ratio: Is it a myth? --> **[Boehm, 1983]**

The history of myth no. 1 --> **[Frank, 1983]**

The iAPX 432, a next generation microprocessor --> **[Rumste, 1983]**

The IEEE 802 Committee states its case concerning its LAN standards efforts --> **[Datacomm, 1982]**

The IEEE standard for the S-100 bus --> **[Garetz, 1983.2]**

The Intel 8089: an integrated I/O processor --> **[El-Ayat, 1979]**

The ISO-ANSI open systems reference model - a proposal for a systems approach --> **[Piatkowski, 1980]**

The LISA computer system --> **[Williams, 1983]**

the LOCALNetter Designer's Handbook --> **[LOCALNetter, 1982]**

The logical design of operating systems --> **[Shaw, 1974]**

The microcomputer connection to local networks --> **[Killen, 1983]**

The micromainframe computer --> **[Intel, 1981]**

The mini-interface: Its prospects and problems --> **[Weiss, 1982]**

The myth of the hardware-software cost ratio --> **[Cragon, 1982]**

The narrowing supermini gap --> **[Mitchell, 1983]**

The NCR/32 processor family --> **[NCR, 1983]**

The new ISO standards for communications and office automation --> **[Standards, 1982]**

The NS16000 family - advances in architecture and hardware --> **[Bal, 1982]**

The NU personal computer: NuBus specification --> **[Ward, 1980]**

The office of the future --> **[Uhlig, 1981]**

The Pick micro race is underway --> **[Gooding, 1983]**

The Roscoe distributed operating system --> **[Solomon, 1979]**

The traffic dependent performance of a prioritized, CSMA broadcast network --> **[Watson, 1979]**

The U* project: an experience with a multimicroprocessor system --> **[Civera, 1982]**

The UNIX C-compiler in a CP/M environment --> **[Halfant, 1983]**

The UNIX programming environment --> **[Kernighan, 1981]**

The UNIX time-sharing system --> **[Ritchie, 1974]**

The Wire-Or glitch: implications for P896 --> **[Gustavston, 1981]**

The world of standards --> **[Card, 1983]**

The Z8000 TUBUS design --> **[Bertelsen, 1980.2]**

Thoth, a portable real-time operating system --> **[Cheriton, 1979]**

Token-access controller minimizes network complexity --> **[Stieglitz, 1982]**

TOMP-80 multimicroprocessor system - preliminary specification --> **[DelCorso, 1980.2]**

Toward a local network standard --> **[Myers, 1982]**

Toward a more complete reference model of computer-based information systems --> **[Bachman, 1982]**

Towards a discipline of real-time programming --> **[Wirth, 1977]**

TP2 og McTan --> **[Bertelsen, 1981.6]**

Transfer rate in a 'filesave' operation using floppy disk (single file, 700 KB) --> **[OL, 1982.3]**

Trends in 16-bit architecture --> **[Muston, 1979]**

TUBUS Specification --> **[Bertelsen, 1980]**

Two Chips endow 32-bit processor with fault-tolerant architecture --> **[Peterson, 1983]**

Two versions of 16-bit chip span microprocessor, microcomputer needs --> **[Shima, 1978]**

Understand the newest processor to avoid future shock --> **[Hemenway, 1981]**

Uniform Interfaces for Distributed Systems --> **[Lantz, 1980.2]**

UNIX and the supermicrocomputer: a marriage of convenience --> **[Lewis, 1982.2]**

UNIX system 5 goes generic in a standard environment --> **[Warren, 1983.3]**

UNIX variant opens a path to managing multiprocessor systems --> **[Jackson, 1983]**

UNIX-based system runs real-time applications --> **[Blackett, 1983]**

Unix-like system standards --> **[Goldberg, 1982]**

Untangling Local Area Networks --> **[Parker, 1983.2]**

VERSAbus - a multiprocessor bus standard - and VMEbus - its Eurocard counter --> **[DeBock, 1983]**

Videotex - Science fiction or reality? --> **[Miller, 1983]**

Violet, an experimental decentralized system --> **[Gifford, 1981]**

Virtual terminal management in a multiple process environment --> **[Lantz, 1979]**

VLSI processor architectures --> **[Treleaven, 1982]**

Voice recognition systems and strategies --> **[Scott, 1983]**

Welcome to the standards jungle --> **[Witten, 1983]**

Western Europe looks to parallel processing for future computers --> **[Electronics, 1983.2]**

What is a distributed data processing system? --> **[Enslow, 1978]**

What is ADA? --> **[Brender, 1981]**

Which local net bus access is most sensitive to traffic congestion --> **[Stuck, 1983.2]**

Winchester disk technologies spins into new orbits --> **[Elphick, 1983]**

Working towards standards in graphics --> **[Langhorst, 1982]**

Workload, performance and reliability of digital computing systems --> **[Castillo, 1980]**

X.21 as a universal digital service access interface --> **[Schindler, 1982]**

X25 protocols and local area networks --> **[Grant, 1982]**

Z-bus and peripheral support packages tie distributed computer systems together --> **[Banning, 1979]**

Z80000 CPU, product specification --> **[Zilog, 1983]**

ZBI: a system bus for the Z8000 --> **[Bender, 1980]**

I N D E X