Flexible interfacing and TIRO B-channel encryption

Heutinck, Richard

Award date:
1991

Flexible Interfacing and TIRO B-Channel Encryption

Author : Richard Heutinck
Date : Mai 23, 1991
Version : Final
Supervisors : Ir. S. Born (RNL)
Ir. J.B. Roubos (RNL)
Ir. M.J.M. van Weert (TUE)
Prof. Ir. M.P.J. Stevens (TUE)

PTT Research Neher Laboratory (RNL)
Department of Communication Systems (CS)
Project group Digital Development (OTWD)

Technical University of Eindhoven (TUE)
Faculty of Informationtechnology
Group Digital Systems (EB)

Preface

This report describes my research activities at the PTT Research Neher Laboratory, Leidschendam. From August 1990 till Mai 1991 I have joined the project group Digital Developments (Ontwikkeling Digitaal) of the Department of Communication Systems.

Readers aimed at

This report primarily addresses readers familiar with the many aspects of telecommunication, especially communication protocols and the OSI and ISDN reference models. Caused by the two subjects discussed in this report, different groups of readers might be interested. The first part deals with the many aspects of flexible interfacing in attempting to move closer to create network-independency and thus addresses readers with interest in that direction. Readers interested in a hardware design with the use of a XILINX Field Programmable Gate Array (FPGA) which provides encipherment of ISDN user-data in an ISDN Oriented Modular (IOM/GCI) interface environment might like to read the second part of this report.

Remains the group of readers which are either interested in both subjects or completely not interested. I would like to advise the former group to read the entire report and the latter group to pass this report to someone of the former groups.

R.H.

Samenvatting

Doordat computers meer en meer onmisbaar worden in onze samenleving, stijgt de behoefte naar communicatie faciliteiten voor computers. Netwerk-onafhankelijkheid is een sleutelwoord in de communicatie-ontwikkelingen. Om deze netwerk-onafhankelijkheid te verwezenlijken zijn flexibele interfaces nodig vanwege de diversiteit in bestaande faciliteiten. In local area omgeving zijn al enkele interfaces ontwikkeld zoals Novell's Open Data-Link Interface en Microsoft/3Com's Network Driver Interface Specification. De ISDN Programmable Communication Interface (PCI) is een duits/frans initiatief, afgeleid van de COMMON-ISDN-API, om een flexibele interface voor het ISDN te creëren. Omdat het ISDN ook datacommunicatie services ondersteunt, die onder meer interworking met LANs mogelijk maken, is integratie van de "computer communicatie wereld" en de "telecommunicatie wereld" vereist. Deze integratie vergroot de complexiteit van de ISDN user-network interface aanzienlijk.

Het eerste deel van dit verslag behandelt de genoemde flexibele interfaces en eindigt met een poging om een antwoord te vinden op de vraag: 'Hoe ziet een interface, in een terminal die aan een ISDN verbonden is en toegang tot en gebruik van de services van dat ISDN vereist, eruit? Het resultaat daarvan is een mogelijk conceptueel model van de interface.

Het tweede deel van dit verslag behandelt het ontwerp en de implementatie van een TIRO_B-Channel_Encryption (TBCE) Module. Daar de gevoeligheid van informatie toeneemt, stijgt ook de behoefte naar beveiligings-services. Vercijfering van gebruikers-data kan een bruikbare methode zijn om meer geavanceerde beveiligingsservices te creëren. De TBCE Module is gemaakt om vercijfering van ISDN-gebruikers-data met behulp van de TIRO-ASIC, ontworpen door PTT Research, mogelijk te maken. De module is toepasbaar binnen een Siemens ISDN chipset en is geïmplementeerd met een ISDN Oriented Modular (IOM) Interface. De module is getest in IOM1 omgeving en werkt. De test in IOM2 omgeving is niet uitgevoerd vanwege de late beschikbaarheid van de IOM2 componenten.

Abstract

As computers become more and more indispensable in our society, the need for communication facilities increases. Network independency is a key-word in communication developments. To create this network independency, flexible interfaces are required, because of the existing diversity in resources. In LAN environment some interfaces are developed such as Novell's Open Data-Link Interface and Microsoft/3Com's Network Driver Interface Specification. The ISDN Programmable Communication Interface (PCI) is a german/french initiative derived from the COMMON-ISDN-API, to create a flexible interface for ISDN environment. Because of provision of datacommunication services, which allows for example interworking with LANs, an integration between the "computer communication" and the "telecommunication" is required, which increases the complexity of the ISDN interface.

The first part of this report deals with the described flexible interfaces and ends with an attempt to answer the question:"What does an interface inside a terminal, connected to an ISDN, demanding access to and use of the services an ISDN provides, look like?", by creating a possible conceptual model.

The second part of this report deals with the design and implementation of a TIRO_B-Channel_Encryption (TBCE) Module. As the sensitivity of information grows, the need for security services increases. Encipherment of user-data can be a useful method to create and provide sophisticated security services. The TBCE Module is created to allow encipherment of ISDN user data with the use of the TIRO-ASIC, designed by PTT Research. The Module is applicable in an Siemens ISDN chipset and is implemented with an ISDN Oriented Modular (IOM) Interface. The Module is tested in IOM1 environment and worked. The test in IOM2 environment is not performed caused by the late availability of the IOM2 components

Contents

Chapter 1

Introduction

Living in the age of information, a society without computers is inconceivable. More and more people have a computer at home and in almost every company they have become indispensable. Hence, information is a conception which plays an important role but in future society its importance shall only increase. As a consequence exchange of information, communication, tracks this development.

A lot of different communication-systems exist nowadays and still more are developed. The problem that arises more often lately is the inability of communicating between different kind of communication systems. Developments in this area show a trend which points at designing flexible systems and/or interfaces. This way communication is provided between systems which were incompatible without the flexible interface. For Local Area Environment some examples are Novell's Open Data Link Interface (ODLI) and Microsoft/3Com's Network Driver Interface Specification.

In the Public Network area the same trend can be recognized. The new Integrated Services Digital Network (ISDN) intends to provide at least all existing communication services and efforts are made to provide future communication services as well. The ISDN will evolve from the existing Public Switched Telephone Network (PSTN). At that time more sophisticated ways of communicating are possible.

The increase in possibilities also changes the need for services. For example the need to interconnect Local Area Networks. This need is caused by the fact that many manufacturers have their own Local Area Networks at their divisions. These Networks can be geographically scattered and still access to, or distribution of, information in the entire company is required. Since a lot of people have a computer at home, the need arises to have access to the facilities and information of their office in order to be able to work at home. This means they have to have access to their company's Local Area Network.

Future developments attempt to create complete independency of network-aspects. This means the user has to have no knowledge about the kind of network he is dealing with, whether this is a public network or a local area network or an arbitrary combination of both. In Local Network area network independency is evolving from independency of one kind of network to independency of different kind of networks by means of the flexible interfaces.

The provision of datacommunication services by an ISDN allows, among other services, more interworking with Local Area Networks. This provision causes, in the same manner as in local network area, a need for a user-network interface which hides the network dependant aspects. Next to the datacommunication services, the ISDN is supposed to provide other services, such as speech as well. This requires an integration between the two different areas in the communication world, the telecommunication area and the 'computer-communication' area.

It is this integration which increases the complexity of the user-network interface of an ISDN. A lot of large manufacturers, such as IBM, AT&T and Hayes occupy themselves with this interface. Also standardization organizations such as CCITT and ETSI perform studies in trying to create a model of the user-network interface in order to standardize it. Three german companies, together with the Deutsche Bundespost,have performed some developments in this area. They have developed an interface called the 'COMMON-ISDN API', where API stands for Application Programmer Interface. After obtaining the support of France, together they have contributed the concept, now called ISDN PCI (Programmable Communications Interface) to CCITT and ETSI.

The importance of information and access to it via a public network or transport of it across a public network gives rise to another kind of service. This service is called security. Abuse of information that is sensitive to an individual, a company or a society must be prevented. One service that is applicable in such situations is encipherment. This encipherment is also usable to create more sophisticated security measures.

This report consists of two parts. The first part deals with flexible interfaces in both LAN and ISDN environment, discussed in this introduction. It describes existing LAN interfaces but focuses to the ISDN environment. The main question behind it is :"What does an interface inside a terminal, connected to an ISDN, demanding access to and use of the services an ISDN provides, look like?". The second part consists of a description of a hardware realisation for offering one small part of security, the encipherment of an ISDN bearer service. In between it contains a chapter about some

security aspects of an ISDN. This chapter globally describes the complexity of the subject and only serves as an introduction to the background of the designed hardware.

The first part consists of chapter 2,3 and 4. Chapter 2 deals with the flexible interfaces in LAN environment. Novell's Open Data Link Interface and Microsoft/3Com's Network Driver Interface Specification are described. Chapter 3 describes the Flexible Interfaces in ISDN environment. It concentrates on the COMMON-ISDN API or ISDN PCI. Chapter 4 ends the first part and discusses the requirements of a flexible ISDN user-interface, trying to create a conceptual model of it using the knowledge of the previous chapters.

Chapter 5 forms the transition between the two parts and describes some aspects of security of an ISDN globally. Furthermore it describes the essential bearer services in an ISDN in which parts of the TIRO system can be integrated. This TIRO System, a system designed by the PTT Netherlands, can be used to secure end-to-end communication against certain threats is also described.

The second part handles the design of a hardware module called "TIRO_B-Channel_Encryption (TBCE) Module" and consists of two chapters. Chapter 6 describes the hardware environment in which the hardware module has to be applicable. The chapter contains descriptions of the surrounding Siemens IC's and their ISDN Oriented Modular (IOM) interface. Chapter 7 deals with the actual design of the TBCE Module. Design Rules are formulated and the design of functional blocks is worked out to a real implementation into a XILINX FPGA.

Chapter 8 finally contains conclusions and evaluations concerning the subjects that are dealt with in this report.

Chapter 2

Flexible Programmer Interfaces in LAN Environment

2.1 Introduction

This chapter describes the state of the art of the flexible interfaces inside the LAN environment. Presently a widespread variety of networks and protocols exists on the market. The standardization committees try to create recommendations which will assure compatibility among products of different manufacturers. Compatibility is a very hot item in the technical business. But the actual situation of different hard- and software packages remains.

A lot of elegant and powerful LANs exist but these networks stand or fall on their ability to service end-user software programs. Companies like Novell Inc, Microsoft Corp. and 3Com Corp. are actively searching to build greater support for user application software into their network operating systems [27]. Often, the route to better access to LAN resources is an application programmer interface (API), which defines the procedures that user programs must follow to gain access to low level LAN software routines.

The Open Systems Interconnection (OSI) seven-layer reference model [1] is used to model LANs. The APIs available today correspond in functionality to almost every layer of this model. This chapter focuses on the low level APIs. Low level APIs provide access to basic communications protocols, which developers can use to transfer data between clients and servers or between any two computers on a peer-to-peer basis.

The latest developments in this area concentrate on providing flexibility. Up to now most APIs were only applicable with one kind of network using one 'OSI protocol stack'. A flexible API or flexible interface offers access to 'configurable protocol stacks'. For example different low level communication protocols can be configured which allows communication between previous incompatible networks. Also different network-independent protocols are applicable on one physical LAN. Network-independent protocols in this chapter mean protocols on layer 3 and above on the OSI stack. Layer 3 protocols in general are not network-independent but considering only networks with the same 'structure', the term network-independent can be used. It more or less means 'network-hardware-independent' in this context.

This chapter deals with the interfaces of Novell (Open Data-Link Interface) and of Microsoft and 3Com (Network Driver Interface Specification). First standardization of the protocols on the different layers of the OSI reference model is explained. The relation between the products that are already on the market and in which way they can be placed in the OSI reference model is clarified.

2.2 Standardization

The OSI seven-layer reference model [1] can be used for modelling a Local Area Network. Most Local Area Networks are connectionless, which means that the protocols on the lower three layers are connectionless. The protocol on the transport layer (layer 4) is mostly a connection oriented protocol.

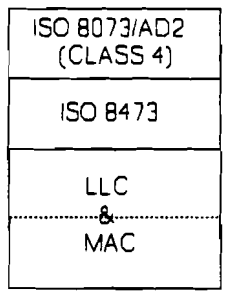


Figure 2.1: *Protocolstack for a connectionless LAN*

A MAC and a LLC layer cover layers 1 and 2 of the OSI reference model though it is not possible to make a clear distinction between these two layers according to the OSI reference model. MAC stands for Media Access Control and LLC stands for Link Level Control. Both LLC and MAC are described in the group IEEE Local Area

Network specifications and are taken over by the International Standardisation Organisation (ISO) in the ISO 8802 series [21].

ISO 8802.2 specifies the LLC part and the different media's are specified in ISO 8802.3 until ISO 8802.6. These specifications respectively stand for 'CSMA/CD', 'Token Bus', Token Ring' and 'DQDB'.

A Connection-Less Network Protocol (CLNP) is specified by OSI in ISO 8473 [20] for layer 3 in the reference model. This protocol is suitable to provides in the Network Service Definition for the CL-transmission, specified in ISO 8348ad1 [19]. This area is still for further study within the CCITT recommendations. Recommendation X.213 resembles ISO 8348 [18], but the CL-service is still for further study.

As layer 3 contains a CL-protocol, layer 4 mostly contains a CO-protocol. Standards are available for this protocol also. The standards for layer 4 contain several classes. Both the ISO standard and the CCITT recommendation contain 5 classes (also referred to as TP's). These classes increase in functionality but are not all compatible. Although there does exist a hierarchy of TP4 above TP2 and TP2 above TP0. The only usable class in CL-LANs is Class 4 or TP4 because the (minimal) service provided by layer 3 requires more functionality of the layer 4 protocol. The documents containing the specifications of the transportprotocol are ISO 8073 and CCITT X.224 [16]. ISO 8072 and CCITT X.214 [15] specify the service these protocols provide.

Inside the LAN-world several 'de facto' standards have arisen in the different protocolstacks. Mostly layer 1 and 2 are implemented according to the standards, but layer 3 and 4 are often implemented different. Several combinations such as Transport Control Protocol (TCP)/ Internet Protocol (IP) and Sequenced Packet Exchange protocol (SPX)/Internet Packet Exchange protocol (IPX) are used. This is possible but also constraining because a combination TCP/IPX or SPX/IP is incompatible without adjustments. Figure 2.2 shows the relation between the different de facto standards and the OSI reference model.

Incompatibility between several 'de facto' standards of figure 2.2 prevents communication between networks, which not use the same configuration. The desire to communicate with all networks, no matter their 'protocol configuration' gave rise to the design of flexible interfaces. A flexible interface allows communication between networks using different 'protocol configurations' provided that the network-independent protocols are corresponding at both sides. Using different network-independent protocols on the same physical LAN is another possibility. The rest of this chapter deals with two of those interfaces. Novell's Open Data Link Interface and Microsoft's Network Driver Interface Specification are discussed in the next sections.

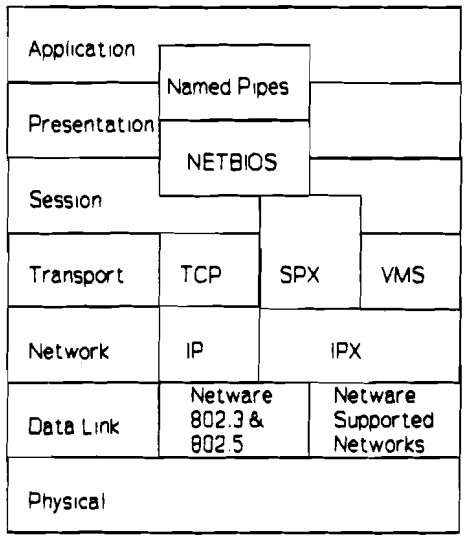


Figure 2.2: *Relation 'de facto' standards and OSI*

2.3 Open Data-Link Interface

2.3.1 Introduction

The Open Data-link Interface (ODLI), a specification jointly designed by Novell and Apple [24], allows many communication protocols to coexist, on the same physical network. Figure 2.3 shows two different communication protocols sharing the same media.

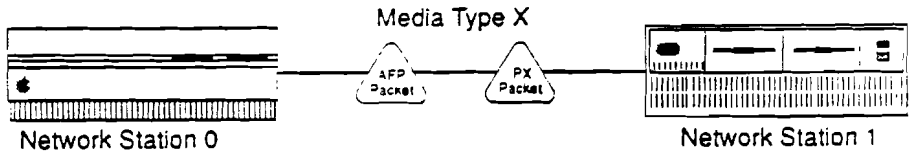


Figure 2.3: *Multiple communication protocols on the same media*

The ODLI allows different applications in the same network station to use different communication protocols on the same network. In figure 2.4, the IPX application in Network Station 0 is communicating with the IPX application in Network Station 1, and the AFP application in Station 0 is communicating with the AFP application in

Network Station 2. Some packets on the network are carrying IPX packets and the rest are carrying AFP packets.

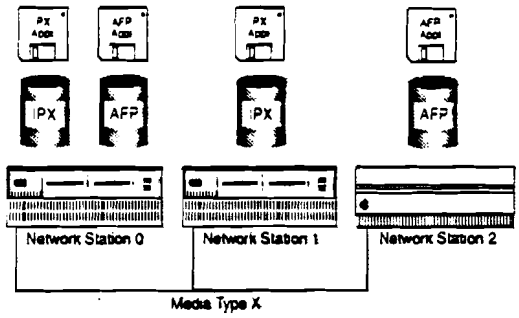


Figure 2.4: *Multiple protocols on one network station*

The next sections discuss the specification of the ODLI and the relations with the OSI reference model. The next section explains the relation between the terms used in the OSI world and the ODLI world. The following section describes the ODLI components.

2.3.2 Relation ODLI and OSI

Figure 2.5 shows an overview of the ODLI architecture on a Netware 386 file server. It shows the NetWare operating system services, protocol stacks such as IPX and AFP, a software layer called the Link Support Layer (LSL) which acts as a packet switchboard, Multiple Link Interface Drivers (MLIDs) which are LAN drivers written to the ODLI specification, and finally one or more LAN boards per MLID.

An 'ODLI protocolstack' consists of layer 3 and all layers above layer 3 of the protocol stack of a communication network in the OSI reference model. The relation between the MLIDs and the LAN boards and the OSI reference model is that these two components represent the lower 2 layers, often also referred to as the MAC and LLC layers, though these are not compatible with the OSI reference model layers.

The Open Data-Link Interface can be placed between layer 2 and 3 of the OSI reference model. This is above the *Link* Layer, therefore it is an Open *Link* Interface, with the *Link Support Layer* as most important component to offer the flexibility. It offers complete flexibility on both sides of the Interface. It is possible to configure different MAC options at the lower side of the LSL as described in the ISO 8802 series. Mostly a LLC layer is placed on top of a MAC layer, adapted to that specific MAC layer. The service of that LLC layer forms the common factor at the top of the

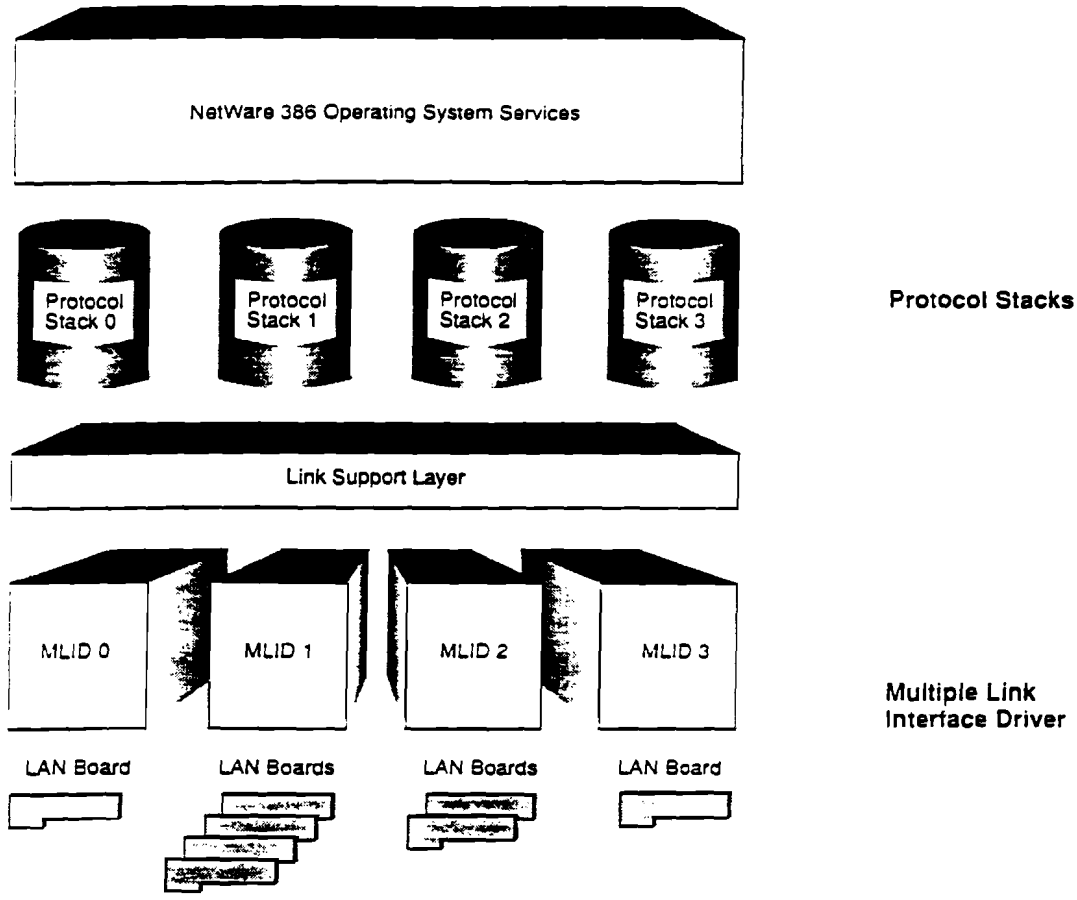


Figure 2.5: *Overview of NetWare 386 ODLI implementation*

lower layer stacks.

At the upper side different 'protocol stacks' (layer 3 and above) can be configured. This can be TCP/IP or SPX/IPX for example. This way, for example NETBIOS, an interface between the session and presentation layer (5 and 6) can be implemented, using SPX/IPX on layer 3 and 4 (see figure 2.2).

2.3.3 ODLI Components

This subsection shortly describes each component of the Open Data-Link interface. The overview of figure 2.5 shows an overview all ODLI components. Besides ODLI Protocol Stacks, the Link Support Layer and MLIDs, the ODLI specification requires the implementation of Media IDs and Protocol IDs.

All mentioned components will be discussed shortly. Furthermore, to increase the insight in the flexibility, the relation between ODLI Protocol Stacks and MLIDs is clarified.

ODLI Protocol Stacks

Network stations that implement ODLI must include one or more ODLI protocol stacks in computer memory. An ODLI protocol stack is a communication protocol (like IPX) written to the ODLI specification. In figure 2.3 both network stations include an IPX protocol stack and an Apple protocol stack (although the stacks are not shown) because both are sending and receiving IPX and Apple packets. The IPX protocol stack on each station sends IPX packets out onto the wire (via LAN drivers and LAN boards) and the Apple protocols send Apple protocols onto the wire (via the same LAN driver and LAN boards). In figure 2.4, network station 0 contains two protocol stacks and the other two stations contain one stack each.

A protocol stack on a NetWare 386 file server also contains a router and a service protocol. The communication protocol packages and unpackages packets, the router routes packets from one network to another, and the service protocol accesses NetWare services.

Multiple Link Interface Driver

Networks stations that implement ODLI must also include LAN drivers written to the ODLI specification. These drivers are called Multiple Link Interface Drivers (MLIDs) because each driver allows a network station to send and receive packets with the same link-level envelope but with different communication protocol headers. In figure 2.3 both packets contain a link-level envelope, but the "AFP packet" contains an AFP communication protocol header and the "IPX packet" contains an IPX communication protocol header.

MLIDs are compatible with non-ODLI compatible LAN drivers. An MLID in one station on a network can communicate with an earlier LAN driver in another station on the same network. The MLID will be able to send and receive multiple protocol packets, but the earlier LAN driver will be limited to sending and receiving only one type of packet. It will receive the packets of course but interpreting them is not possible and this is in this context also meant by receiving.

Link Support Layer

Network stations that implement ODLI must include a software layer called the Link

Support Layer (LSL). Both protocol stacks and MLIDs must be written to interface with the Link Support Layer residing with them in computer memory.

The NetWare 386 file server LSL collects information from and dispenses information to protocol stacks and MLIDs. As a result, protocol stacks and MLIDs do not need to know about each other initially.

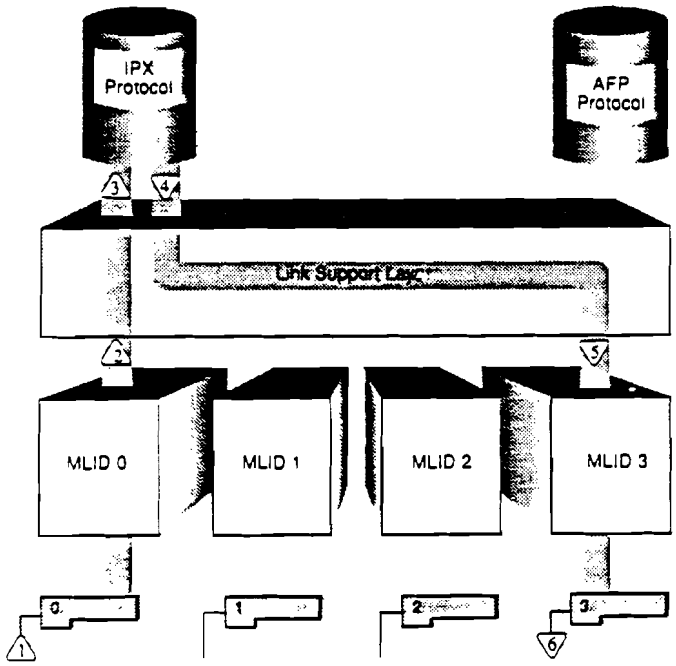


Figure 2.6: *The LSL directing traffic*

The LSL also routes incoming packets from the receiving MLID to the target protocol stack. The LSL routes outgoing packets from the originating protocol stack to the target MLID (and consequently to the target LAN board and network). Figure 2.6 shows the NetWare 386 file server LSL directing traffic. The steps shown in the figure are explained below:

1) Board 0 receives an IPX packet.

2) MLID 0 hands the IPX packet to the LSL.

3) The LSL directs the incoming IPX packet to the IPX protocol stack.

4) IPX prepares the packet for routing onto another network and hands the packet back to the LSL.

5) The LSL directs the packet to MLID 3.

6) Board 3 puts the packet onto the destination network.

The next packet received by board 0 could, for example, be an AFP packet that needs routing to board 2. The LSL would direct the AFP packet to the AFP stack (which would route the packet to board 2) in the same way that the LSL directed the IPX packet to the IPX stack. Both incoming packets would share the same kind of link-level envelope.

ODLI Protocol Stacks and MLIDs

In order to clarify the flexibility the ODLI offers the relation between ODLI Protocol Stacks and MLIDs is described.

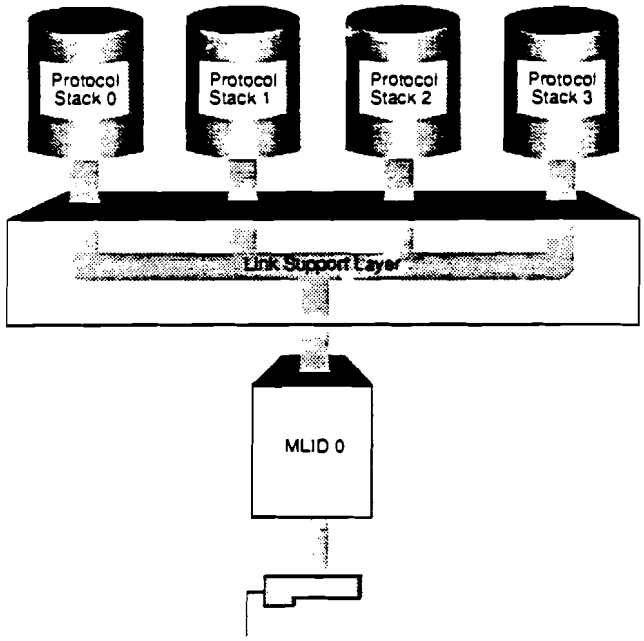


Figure 2.7: *One MLID servicing multiple protocols*

One LAN board (driven by an MLID configured to deal with link-level envelope type X can send and receive packets for more than one protocol stack, as shown in fig 2.7.

The packets that MLID 0 sends and receives on the wire differ only in that they have different communication protocol headers and data. Figure 2.8 illustrates this point.

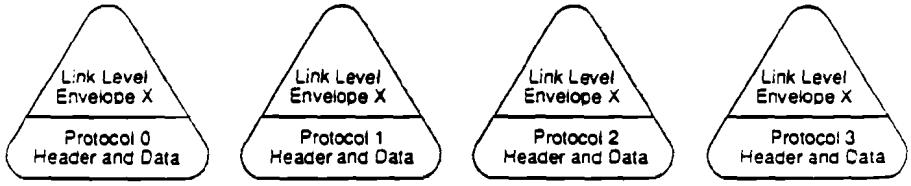


Figure 2.8: *MLID 0 handles all these packets*

One protocol stack can send packets to and receive packets from more than one LAN board. These boards may or may not be media-identical. Figure 2.9 illustrates how several different MLIDs can service one protocol stack.

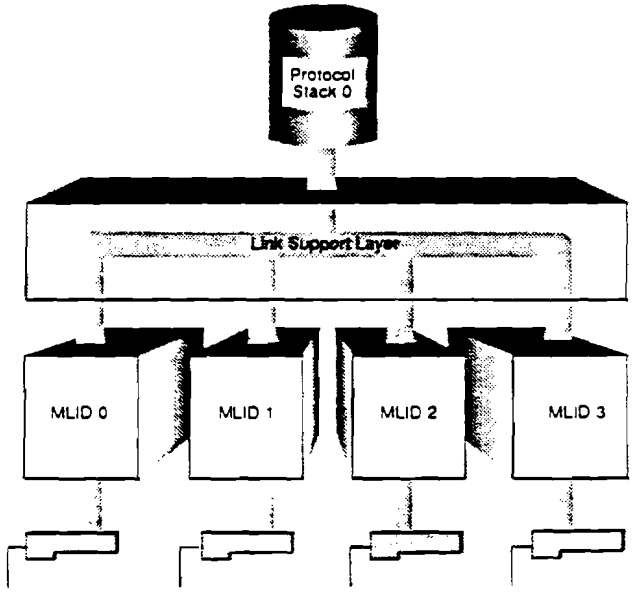


Figure 2.9: *One protocol stack serviced by many MLIDs*

As figure 2.10 illustrates, the packets originated by protocol stack 0 will all have different link-level envelopes.

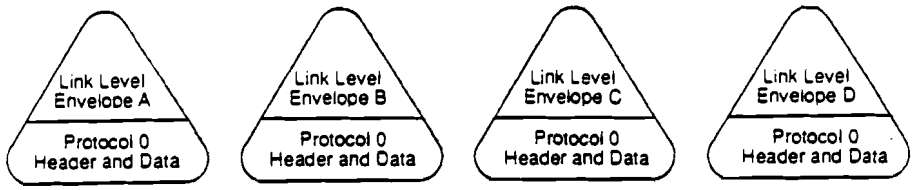


Figure 2.10: *Protocolstack 0 originates all these packets*

MLIDs can handle multiple-protocol packets and protocol stacks can handle multiple-media packets. This way communication is allowed between previous incompatible networks using different 'protocol configurations' provided the protocols on layer 3 and above are the same or compatible on both sides.

Media IDs

A Media ID refers to the type of link-level envelope that an MLID is configured to use. The ODLI specification categorizes all MLIDs into the following classes, according to the media type.

Media TYPE	Description
Virtual_LAN	For use when no Media ID/MAC envelope is necessary
LOCALTALK	The Apple LocalTalk media
ETHERNET_II	Ethernet using a DEC Ethernet II envelope
ETHERNET_802.2	Ethernet (802.3) using an 802.2 envelope.
IBM_TOKEN-RING	Token Ring (802.5) and 802.5 with an 802.2 envelope
ETHERNET_802.3	802.3 envelope as defined by IEEE (CSMA./CD)
802.4	Token-passing bus envelope (ARCNET)
NOVELL_PCN2	Novell's IBM PC Network 2 envelope
GNET	Gateways media envelope

Protocol IDs

Every packet on the wire contains a one- to six-byte value called the Protocol ID (PID). This value is embedded in the link level envelope of every packet on the wire.

Each media type (see section above) has a list of PID values that identify protocol stacks for that media type. Each PID value within a media class is unique. However PID values are not unique across all media classes.

When receiving a packet, the LSL first checks the media classification of the receiving MLID. Then it looks at a registered list of PIDs for that media classification to determine the target protocol stack.

2.4 Network Driver Interface Specification

2.4.1 Introduction

A group of vendors led by Microsoft and 3Com has proposed the Network Driver Interface Specification (NDIS) [25]. It also provides flexibility via an interface to allow communication between previous incompatible networks and the use of different network-independent protocols on one physical LAN. Because of the fact that the two specifications of ODLI and NDIS are fundamentally similar, not the entire NDIS is discussed. The NDIS structure also resembles the ODLI structure. It consists of low level drivers, drivers for provision of high level communication services (ODLI protocol stacks) and a managing entity. The next subsection deals with the global resemblances and differences of ODLI and NDIS. The next section discusses the components of the NDIS and their relation with the ODLI components.

2.4.2 Relation NDIS and ODLI

The two specifications are functionally similar even according to the competing vendors themselves [26]. The major difference between the two specifications is probably the Link Support Layer of ODLI versus the chained protocol stack of NDIS. Where the LSL routes the packet to the proper stack, NDIS routes the packet from one protocol stack to the next until one of the protocol stacks recognizes the packet's format. Figure 2.11 clarifies the comparison between both interfaces.

2.4.3 NDIS Components

This subsection shortly discusses each NDIS component. Its function is described and furthermore their relation with the ODLI components are discussed. The components

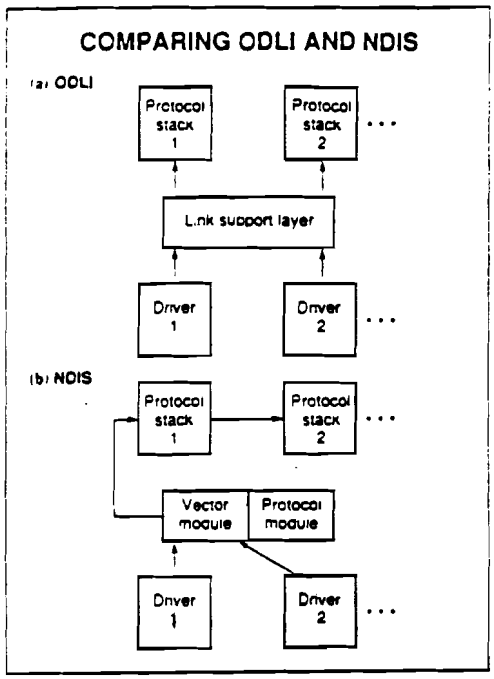


Figure 2.11 : *Comparing OLI and NDIS*

are Media Access Control Drivers, MAC layer entities, protocol drivers and a protocol manager driver.

Media Access Control driver

A Media Access Control (MAC) driver provides low level access to network adapters. The main function of a MAC Driver is to support transmitting and receiving packets, plus some basic adapter management functions. It resembles the Multiple Link Interface Driver (MLID) of the ODLI.

MAC-layer entities

MAC-layer entities bind to real MAC-Drivers and expose a new MAC-like layer interface on top. Both the Mac layer entities and the Mac Drivers can be seen as an Open System Interconnections (OSI) model sublayer that links communications between the physical layer and the software that runs on the network. A MAC-layer entity relates to the upper part of an MLID or the lower part of the LSL, also called the multiple link interface, of the ODLI.

Protocol Drivers

Protocol drivers provide higher-level communication services from data-link to session (depending on the driver). An example is a NetBIOS driver that provides a NetBIOS interface at the top and talks to a MAC driver at the bottom. It can be compared with the 'protocol stacks' of the ODLI. The NETBIOS example relates to an ODLI protocol stack with IPX, SPX and NETBIOS on layers 3,4 and 5 for example.

Protocol Manager Driver

The Protocol Manager Driver provides in a standardized way for multiple MAC and Protocol Drivers to get configuration information and bind together into the desired protocol hierarchy. This driver can be compared with the Link Support Layer of the ODLI. Only the LSL sends the packet directly to the desired stack and here every stack is 'polled' if it is the destination stack.

2.5 Summary

Standardization of protocols used in Local Area Networks is quite stable. Lower layers (1 and 2) often are implemented according to the ISO 8802 series which means a MAC layer and an LLC layer. This division is not completely according to the OSI references model but still occurs often in practice. Other combinations on the higher layers (3 and above) are often mutual incompatible.

The need for communication between these 'incompatible' Local Area Network gave rise to the development of flexible interfaces. The flexibility allows communication between Local Area Networks using different 'protocol configurations', provided the same protocols on layer 3 and above exist on both sides of the communication entities.

Two of those interfaces, Novell's Open Data-Link Interface and Microsoft/3Com's Network Driver Interface Specification, are discussed in this chapter. Both interfaces provide the same functionality and show a lot of resemblances. The major difference is found in the mechanism of the 'managing entity'.

Both interfaces try to earn the status of 'de facto' standard. It is not clear which one is going to get that status. Maybe even both get the status and compatibility is created in a way that end-user software is applicable on both interfaces.

Chapter 3

Flexible Programmer Interfaces in ISDN Environment

3.1 Introduction

The next public network will be an Integrated Services Digital Network (ISDN) [3]. It shall be a network in which a lot of services are integrated. The provision of this variety of services causes a need to standardize their individual access. The same philosophy arises which exists for some time in local area environment:'Create interfaces which shield the user for network dependent aspects and provide standardized access to network facilities and resources'.

The provision of datacommunication services by an ISDN changes the need for services. The existing public networks will be if possible integrated in the ISDN. The existence of Local Area Networks, often owned by companies causes a need for a service which allows mutual communication. Another need that arises is the desire for people to work at home on their own PC, making use of the facilities and applications of the LAN of their office.

Therefore, interworking between an ISDN and LANs has to be created. Next to that facilities have to be created which allow a terminal connected to an ISDN to gain access to a specific LAN and make use of its facilities and resources. This calls for a special user-network interface for an ISDN.

A proposition for such a basic rate interface is made and is contributed to standardization committees. It is called an ISDN Programmable Communication

Interface (PCI). It is derived from an earlier initiative of three german companies then called Common-ISDN-Application Programmer Interface (COMMON-ISDN-API) [2,31].

This chapter describes this interface but first standardization of the lower layer protocols inside the ISDN reference model will be explained.

3.2 Standardization

The ISDN reference model consists functionally of two protocol stacks. One stack is used for signalling, the other one is used for user-to-user communication. The figure shows the general protocol stack for ISDN.

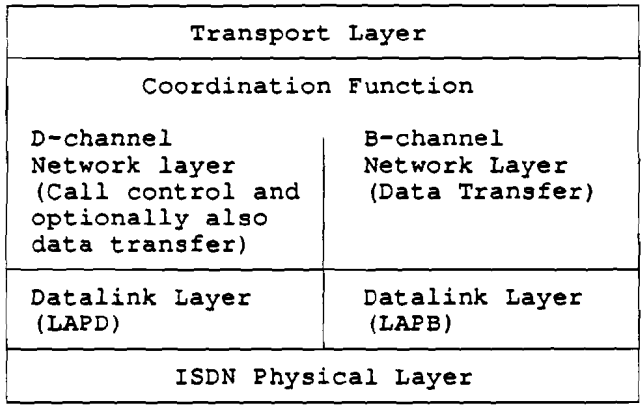


Figure 3.1: *General protocol stack for ISDN terminals*

Standardization of the physical layer and the D-channel layers is quite stable. The standards for the physical layer are CCITT I.430 [6] and I.431 [7] for the Basic Rate Access and the Primary Rate Access respectively. The corresponding european standards are formed by the European Telecommunication Standard Institute (ETSI) and are called ETS T/L 03-7 and T/L 03-14.

For layer 2 and 3 of the D-channel the CCITT Recommendations Q.921 [8] and Q.931 [9] are available. Again the corresponding european standards are ETS T/S 46-20 and ETS T/S 46-30. In most cases the european standards (ETS= European Technical Standard) are generally the same as the CCITT Recommendations however with several chosen options. The CCITT Recommendations offer an enormous amount of choices. The ETSs describe some of the european choices of these options.

Standardization of the B-channel is not really stable. There are several alternatives depending on the services of the ISDN. The differences in standardization protocols and services are mainly caused by a different point of view. There are two main streams in this case. One focuses on datacommunication and dataprocessing systems, often referred to as 'computer-communication', the other one on telecommunication and telematic services. These two streams require different protocols.

Some alternatives are available for the circuit mode bearer services [section 5.2]. In T.90 [11], CCITT X.75 SLP is used for layer 2 of the B-channel and also ISO 7776 [14] is an alternative. CCITT X.75 is a network interconnection protocol. CCITT X.75 SLP stands for Single Link Procedure referring to the used part. Though many modifications must be made, the choice can be understood for a historical reason, namely CCITT T.70 describes (among other things) the protocols to be used on the lower four layers on Circuit Switched Public Data Networks (CSPDNs), for example for FAX- or Videotex-like applications. The datalink layer for CSPDNs conforming T.70 uses X.75 SLP. The T.90 description is based on this description [23].

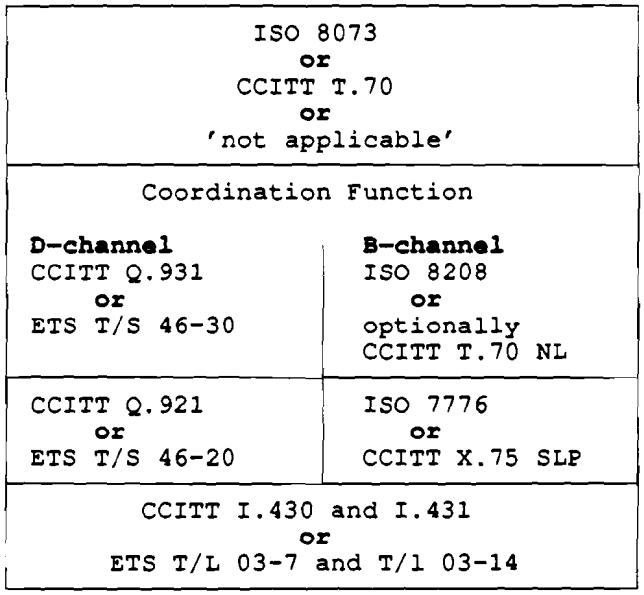


Figure 3.2: *ISDN Protocol Stack for Circuit Mode*

ISO 7776 is developed by ISO as a LAPB protocol for terminal-terminal communication. So modifications have to be made here also. The exact difference between these protocols is evaluated by ETSI PT15 and the conclusion is that the two implementations will be able to interwork, but not very efficient.

For the packet mode bearer services [section 5.2] T.90 requires X.25 LAPB on layer 2. Another alternative is ISO 7776. As the description in ISO 7776 for the DTE-DCE case is technically equivalent to X.25 LAPB, this does imply compatibility.

For the circuit mode bearer services layer 3 of the B-channel there are some other alternatives. The first alternative is ISO 8208 [17], which is in most cases default. ISO 8208 is the ISO version of CCITT X.25, which includes a description of the DTE-DTE case besides the DTE-DCE case.

In T.90 two incompatible options are described. One possibility is the implementation of ISO 8208. The other (additional) option in T.90 is to leave layer 3 functionally empty by using the so-called 'telematica header' in the layer 3 PDUs. This procedure is described in CCITT T.70 [10] and is in use in CSPDNs. T.90 refers to this option as T.70NL.

For the packet mode bearer services T.90 requires the use of X.25 PLP. Again another alternative is ISO 8208. As the description in ISO 8208 for the DTE-DCE case is technically equivalent to X.25 PLP, this does not imply incompatibility

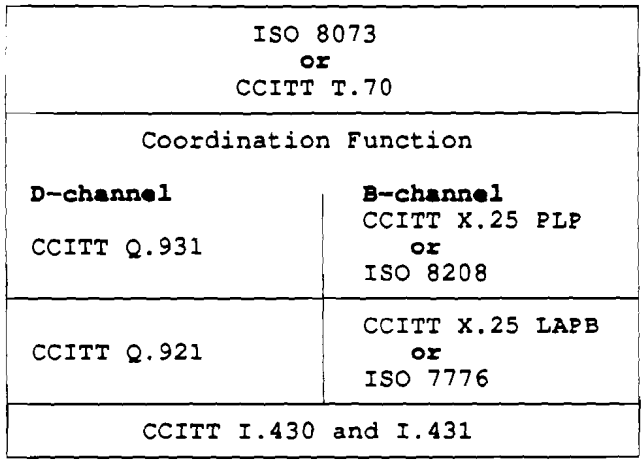


Figure 3.3: *Protocol Stack for Packet Mode (B-ch)*

The coordination function on top of layer 3 takes care of the interaction between D-channel and B-channel. A description of such a function is described in ISO 9574 PDAD1 [22], which describes the provision of OSI-CO-NS by a packet mode terminal connected to an ISDN. It contains a mapping of Q.931 primitives on X.213 primitives

CCITT T.90 does not require explicitly the use of a specific layer 4 protocol for both circuit as packet mode bearer services. However some references are made to certain layer 7 protocols which in turn use CCITT T.70 on layer 4, or leave layer 4 empty (Videotex). T.70 defines the transport protocol for telematic services. T.70 is more or less equivalent to class 0 of X.224. ISO 8073 [16] is also equivalent to CCITT X.224.

3.3 Common ISDN API (PCI)

3.3.1 Introduction

Common ISDN API stands for Common ISDN Application Programmer Interface [2]. It describes a standard interface between application programs and ISDN adapters. The initiative came at first from three german companies AVM, Stollman and Systec. This initiative is taken over by the Deutsche Bundespost (DBP) Telekom and is also supported by France Telekom. Both countries have contributed the specification of the Common-ISDN API to the standardization committees (CCITT and ETSI). It is no longer called Common ISDN API but ISDN PCI, which stands for ISDN Programmable Communication Interface.

This section describes the modelling of the ISDN PCI and explains its mechanism. Appendix A contains some illustrations of this mechanism.

3.3.2 Overview

From its logical viewpoint, the ISDN PCI provides a connection between any number of application programs (applications) and any number of ISDN drivers and ISDN controllers via a standard interface. The ISDN PCI enables both one application to be used for several drivers and controllers and several applications to be serviced by a single driver.

The applications can use different protocols at different protocol levels, the ISDN PCI providing the selection mechanism needed for this. The specification of the ISDN PCI standardizes the protocol between an application and the ISDN PCI. The protocol between the ISDN PCI and the supported drivers is manufacturer-specific.

The ISDN PCI is intended to facilitate the connection of ISDN terminals which are designed for communications over an ISDN. It provides access to those ISDN bearer services defined in the ISDN CCITT Recommendation I.230 [4]. It provides a standardized access to services needed to set up an ISDN connection.

3.3.3 Architecture

The ISDN PCI is divided into three functional units, which are assigned to the ISDN Reference Model:

control plane

Uses the services as defined in Recommendation CCITT Q.931 and Q.932 and is therefore located at the upper boundary of the D-channel protocol.

user plane

Provides default the X.213 services at the Network Service Access Point (NSAP). Options that are available for providing other services at other Service Access Points are;

* X.75/T.90 services at the Data Link Service Access Point (DL-SAP).

* I.430/I.431 services at the Physical Service Access Point (PH-SAP).

administration plane

Placed outside of the OSI Reference Model, but for the time being restricted to the access of the so-called lower layers (Layer 1-3)

The location of these planes and their relation with the OSI Reference Model is clarified in figure 3.4. The figure also clarifies the different locations of the user plane caused by the possible options of the lower layer B-channel protocols.

3.3.4 Functional Model

The model of the ISDN PCI only describes the basic concept. It does not imply a certain realisation. The model mainly consists of three parts, each applicable one at a time. The following subsections discuss each part separately.

3.3.4.1 Initialization of an ISDN application

Before the communication entity of an application can access the services of an ISDN

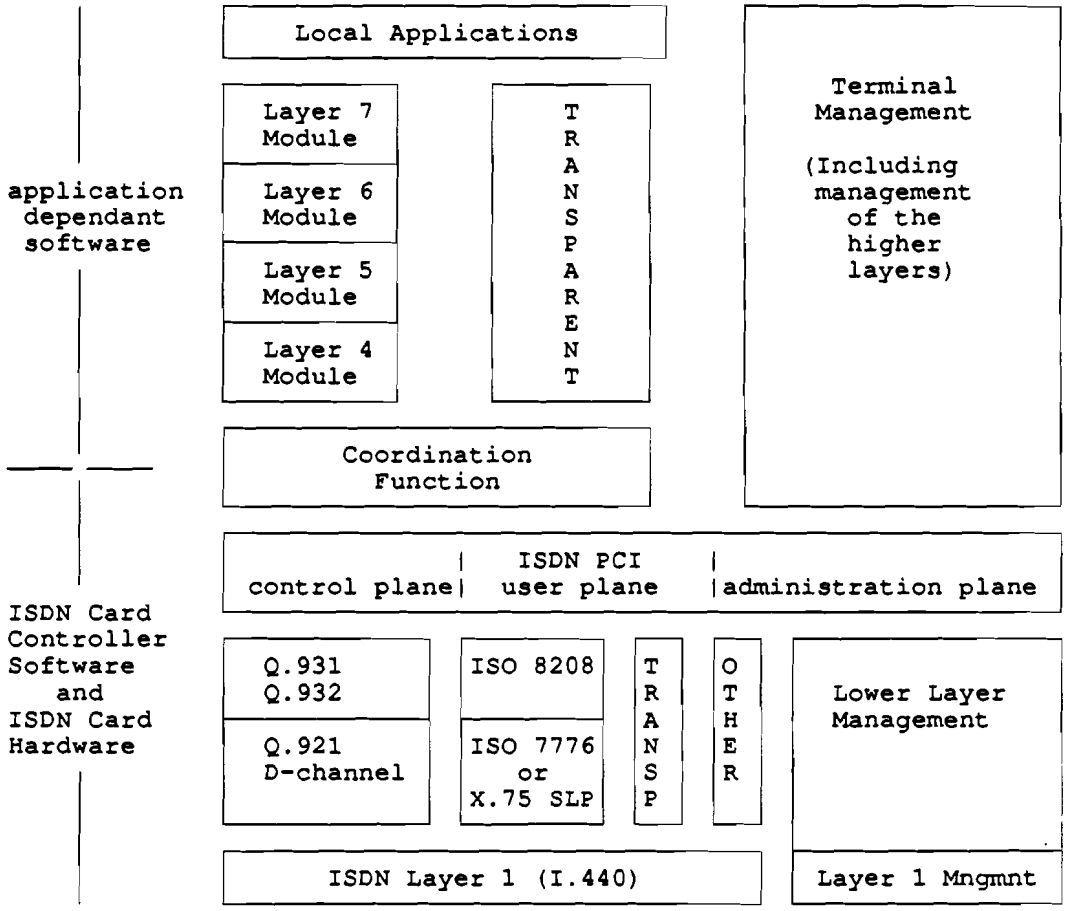


Figure 3.4: *Proposed location of ISDN PCI in OSI Reference Model*

network access facility, it must be initialized and bound at the PCI. The aim of such initialization is to;

- allocate the necessary resources
- identify the application without any ambiguity, by assigning an unique application identifier

An application is registered to the ISDN PCI with the PCI_REGISTER function. The ISDN PCI uses this function to assign an system-wide unique application identification (APPL-ID) to the application. In this context system-wide doesn't mean unique within a network but 'S/T-reference point unique'. The message queue for the application is set up at the same time. The application makes the storage space needed for this available to the ISDN PCI.

3.3.4.2 The message transfer mechanism

The communication between an ISDN application and the ISDN PCI is provided by means of operations on messages. Figure 3.5 shows the environment.

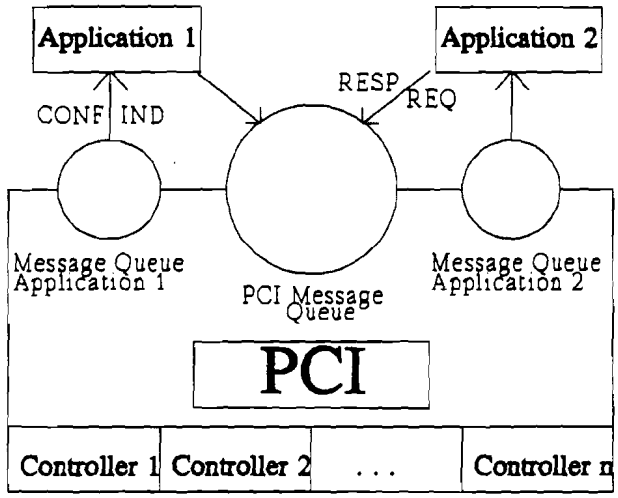


Figure 3.5: *Environment of message queues in the ISDN PCI*

Each application has its own message queue destined for messages to that application and all messages from the applications are gathered in the PCI message queue.

A message consists of a fixed length header and a data area of variable length as shown in figure 3.6. This format is applicable for both directions

Message Header	Message Data	Message Data			Message Data

Figure 3.6: *Message structure*

The structure of the message header is shown in figure 3.7

Total Length	APPL ID	Command	Sub-Command	Message Number

Figure 3.7: *Structure of the message header*

The abbreviations used in figure 3.7 have the following meaning;

Total Length	Word, total length of the message including header in bytes.
APPL-ID	Word, identification of the application. The APPL-ID assigned to the application by the ISDN PCI in the PCI_REGISTER operation is used.
Command	Byte, Command
Subcommand	Byte, command subdivision
Messagenumber	Word, unique number assigned to each message

The structure of the message data consists of two distinct types , namely the fixed length type such as Byte, Word and DWord and the variable length type such as Struct. Figure 3.8 shows the structure of both types.

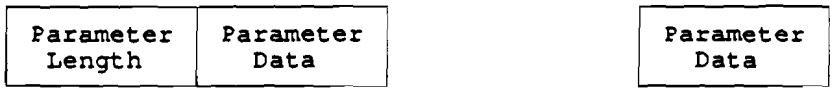


Figure 3.8: Structure of variable and fixed length message data

The number of parameters transferred in a message and their data types are defined implicitly by the particular message. The maximum length of a message is 180 bytes.

The ISDN network access facility is responsible for providing all the utilities for message exchange over the PCI, so the management entity of the ISDN application needs no special utilities, but it does need to provide the resources (memory) for handling the messages.

The exchange of messages between applications and network access facilities over the ISDN PCI takes place according to the following protocol. Each message is given a messagenumber. Two different (disjunct) sets of numbers are used, depending on the initiator: between 0x0000 and 0x7FFF when the application is initiator and between 0x8000 and 0xFFF when the ISDN network access facility is the initiator.

When the application sends a request (_REQ) to the ISDN network access facility, a message number from the application's set of numbers is used. The ISDN PCI confirm's this request with a corresponding confirmation (_CONF), using the same message number as in the request.

When an asynchronous event is indicated to the application by an indication (_IND), a message number from the ISDN PCI's set of number is used. The application replies

to the indication with the corresponding response (_RESP), using the same message number as in the indication.

The structure, the contents and the effects of the different messages will not be described. This is a too detailed matter for this report. Messages sent from the application to the ISDN PCI use the function PCI_SEND_MESSAGE. An error code is returned if the message could not be accepted. Messages from the ISDN PCI to the application use the function PCI_RECEIVE_MESSAGE. Again an error code is returned if this wasn't successful.

3.3.4.3 Releasing of an ISDN Application

After an application has registered, it must always release itself from the ISDN PCI. This can be done at any time with the PCI_RELEASE operation. Note that when this is done the application must clear down all existing connections. Releasing the application frees the previously used message queue.

3.4 Summary

At the writing of this report, the Common ISDN API or ISDN PCI is an agenda item on the meetings of CCITT SG8 and ETSI TE7. The originators of the concept of the ISDN PCI try to get their contribution standardized.

The problem however is that the modelling is not complete. The provision of 'primitives' (messages) and procedures working with those primitives is a step in the right direction, but only access is provided to some services. The supplementary services and their access for example are completely omitted in the represented model.

The next chapter will dig deeper into this subject, also describing the other actions and developments in the area of the API or PCI.

Chapter 4

Modelling of a Flexible Programmer Interface for ISDN

4.1 Introduction

The area of interfacing is very actual nowadays. There are a lot of different opinions from also different viewpoints. Not only the standardization organizations (ETSI, CCITT, NIU) are working on a conceptual model, but many large manufacturers, for example Hayes, AT&T and IBM develop API's which they would like to be seen standardized [29,30].

The question is very simple:'What is the model of the interface, inside a terminal, connected to the ISDN, demanding access to and use of the services an ISDN offers?'

The answer to this question however is not so simple. The required integration of typical 'computer communication' and 'telecommunication' oriented services into one interface increases its complexity. The interface has to provide access to services which require different protocols. This calls for a kind of programmability. The user-network interface discussed in the previous chapter is therefore called ISDN Programmable Communication Interface. The first section of this chapter attempts to define a concrete definition of an ISDN PCI which forms a basis for its proposed conceptual model. In addition, the relation between an API and a PCI is explained.

The next section discusses the location of the user-network interface referring to the layers of the ISDN and OSI reference models. Then, an attempt is made to construct a model of an ISDN PCI. Therefore, first two ways of gathering the requirements are described in two sections, the so-called 'Application View' and 'Network View'. The

last two sections contain a proposition for a conceptual model and a summary with some evaluations about the modelling of the ISDN PCI in general added with some aspects about the actual practical situation concerning the ISDN PCI's or APIs

4.2 Definition of an ISDN API or ISDN PCI

As already mentioned API stands for 'Application Programmer Interface' and PCI stands for 'Programmable Communication Interface'. Sometimes the API works confusing because of the term 'Application'. It is sometimes related to the application layer in the OSI Reference Model and sometimes an application is seen as a software package of a user which interfaces with, for example, the network layer in OSI terms.

The term 'Programmable' in PCI promises a kind of flexibility. The interface can be programmed with options which means that it offers different services for different applications. This does not mean that an API can't offer different functions. This is possible but it is also possible that an API offers just one interface and is not 'programmable' at all. Hence an API and a PCI can be exactly the same but not all API's are PCI's while on the other hand all PCI's can be called API's. Furthermore it is also possible that a PCI consists of more than one API's which can be programmed active or non-active.

A PCI or API is in most cases nothing more than a software implementation of a service specification. The service specification specifies a set of primitives and a set procedures using those primitives. The API consists of a library with the software implementation of the specified procedures which work on a datastructure representation of the specified primitives. Additionally procedures are added for, mostly local, maintenance functions.

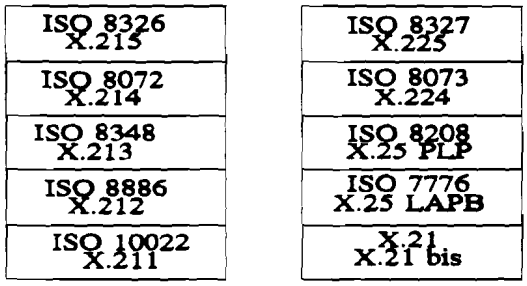


Figure 4.1: *Protocol stacks for services and protocols*

This service specifications are standardized in OSI environment. It is possible to configure a protocol stack filled with service specifications next to the stack filled with protocol specifications which are suitable to provide the specified service as figure 4.1 shows. For example X.25PLP is a specification of a protocol that can provide the service specified in X.213. In this case a Connection Oriented Packet Switched Network API or PCI is a software implementation of X.213. It consists of a library of functions which hides the networkaspects to the user of the service. Additional functions have to be implemented which allocate and maintain necessary resources.

If the OSI reference model was applicable for an ISDN problems would have been solved. An ISDN however, has its own reference model which is not completely compatible with the OSI reference model. Still it is possible to formulate a definition of an ISDN PCI:

> **An ISDN PCI is a software implementation of a service definition of an ISDN, with the term service definition used as defined in OSI, with additional functions for allocating and managing resources'.**

The additional functions mentioned in the definition are 'environment-specific' and therefore not interesting for modelling aspects. In the next sections some attempt are made to model an ISDN PCI more exactly than this given definition. First of all the location of an API or PCI is discussed.

4.3 Location of an ISDN PCI

Within the seven-layer OSI Reference Model it is possible to, in accordance with the definition, place an API on top of each layer, because an API is a language binding of an OSI service definition. The 'A' of Application has in each case a different meaning.

Different viewpoints exist on the location of an API. IBM for example places an API on top of layer 7. That is why it is called an API, they say. It is an application programmer interface, so on top of the application layer. This API forms then an interface which is specific for some applications and only functions if the OSI protocol stack beneath that API is the same or at least compatible with the IBM stack.

Another viewpoint is to locate a flexible interface on top of the network layer. The boundary between layer 3 and layer 4 has always been a special one. Above layer 3 network-independency can be created. The network is given and the functions that can be executed are contained in the interface. A second argument is the concept of the

well-known 'OSI wineglass'. This wineglass picture of standards has its thinnest point at layer 3, so the least alternatives can be found there. If an interface can be created that implements those alternatives, the whole 'upper stack' of standards can be implemented on that interface.

The boundary between layer 3 and layer 4 is also a very important boundary for the networkoperator. The original question at the start of this chapter was about the model of an interface inside a terminal connected to an ISDN. This interface is on top of layer 3 because there the network ends.

The location of an ISDN API or ISDN PCI which is discussed in this report is decided on top of layer 3.

4.4 Application View

One possibility to model an interface is to gather the requirements of all the desired applications and/or services. A model then can be constructed by formalizing the requirements into standardized messages and procedures and providing a mechanism to use them. This is meant by the 'Application View' in this section.

The main problems of modelling a PCI are caused by the availability of already existing services. Not the availability forms the problem, but the variety in required resources is. The purpose of an ISDN (Integrated Services Digital Networks) is to integrate the existing services into one single network.

The major problem is the integration of 'computer-communication services' and 'telecommunication services'. A Local Area Network is an example of a computer-communication system and FAX or Videotex are examples of telecommunication systems. Both systems require typical communication services.

This means for example that parallel to the Telephony-, FAX- and Videotex-systems, the existing widespread variety of Local Area Networks has to be connected to the ISDN. This requires special designed relays. With those relays communication between LANs, connected only to an ISDN, is possible. The provision of flexible interfaces for LAN environment on a relay even allows communication between previous incompatible LANs (see chapter 2). If this flexible LAN interface is provided on an ISDN user-network interface as well, access to the existing LANS can be gained from a single terminal.

Regretfully more problems arise if such attempts are made. Mostly LANs are connectionless networks, while an ISDN is connection oriented (see 2.2). This partly is caused by the structure of both network types. Presently efficient interworking between both types of networks is studied. At this moment the provision of connection-less services is for further study in the CCITT Recommendation X.213. The technically aligned ISO standard ISO 8348 [18] has an addendum dedicated to this subject. This is ISO 8348ad1 [19].

Despite the described problems it is still possible to divide the services, in OSI terms, all applications need, into three groups:

- Connection Oriented Network Services (CONS) X.213/ISO 8348
- Connection-Less Network Services (CLNS) ISO 8348ad1
- NON-OSI Services

This division is easily made, but in practice the NON-OSI services are very hard to standardize. It is very difficult to gather the requirements of the widespread variety of existing applications into one general purpose service definition. Besides that, the specification of a service definition derived from the existing applications is in fact standardizing the applications. Therefore, only the 'Application View' is not very suitable in this case from a modelling viewpoint.

4.5 Network View

The opposite method of the Application View is of course the Network View. The ISDN Reference Model and the available standards form the basis of this method.

ISDN stands for Integrated Services Digital Network. All mentioned applications are applicable over an ISDN, provided application-specific adaptions, needed in addition to the ISDN service, are made . The only problem is to get access to the right services of an ISDN. The model of an ISDN PCI is, according to the definition in section 4.3, nothing more than a service definition in OSI terms of the top layer of the ISDN Reference Model. With this definition the required service could be accessed straightforward.

The services an ISDN offers are divided in 'Bearer Services' and 'Tele Services' which are described in Recommendations CCITT I.230 [4] and CCITT I.240 [4]. The problem however is although these are specifications of the services of an ISDN, they are not the same as the service specifications in OSI-terms.

The difference in supplied services and access to this services are partly caused by the difference between the ISDN and the OSI reference model. The major difference is the separate signalling plane of the ISDN. Figure 4.2 shows an outline of both reference models.

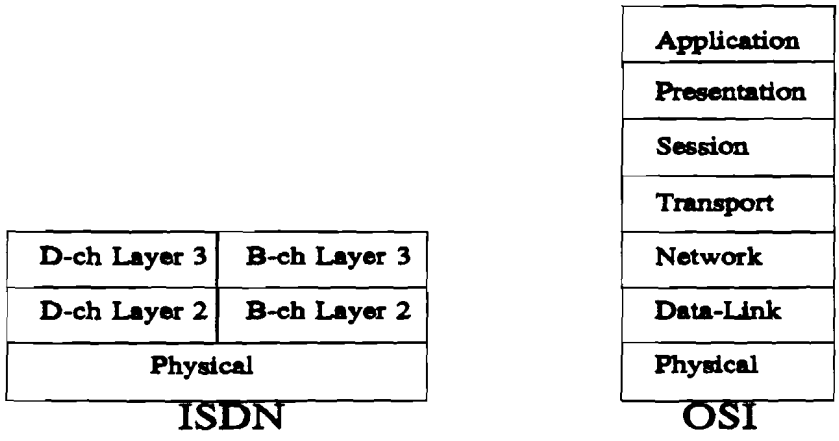


Figure 4.2: *Reference Models*

In order to make the existing OSI applications applicable via an ISDN, a mapping needs to be created to map layer 4 and up of the OSI reference Model on top of the ISDN Reference Model. To provide the OSI Connection Oriented Network Services some work is done on a coordination function [see Section 3.2] which describes the procedure to create an X.213 interface on top of the ISDN.

In fact this is the other way around. In this manner the service definition in OSI terms of an ISDN (which presently not exists) is converted into a less functional service definition. In this way only the OSI CONS service can be provided. The proper way to do this is to create a general service definition in OSI terms of an ISDN and then describe which subset of this definition should be used to provide the OSI CONS.

The adjustment of applications to an ISDN PCI will be inevitable. Even if a detailed service definition of an ISDN, and thus an ISDN PCI is created, still mapping functions will be necessary to adapt the communication requirements of an applications to the according procedure calls of the ISDN PCI.

New developments concerning the lower layer protocols of the ISDN Reference Model complicate matters even more. A new bearer service category will be added called: "Frame Mode Bearer Service Category". The protocol is called Q.922 and is an addition to Q.921. With Q.922 core it is possible to offer an LLC service on top of the

layer 2 B-channel and methods are developed which simulate the provision of a connection-less service. Chapter 2 describes the Open Data Link Interface with the Link Support Layer using the LLC service. Implementing this interface on top of this layer is very interesting, because it opens up the whole local area world is for the user with his single terminal connected to an ISDN. Although no studies have been made concerning the adjustments necessary to access and manage the appropriate signalling functions.

4.6 Proposed conceptual Model of an ISDN PCI

This section tries to define a conceptual model of an ISDN PCI combining the results of the described 'Application View' and 'Network View'. Although the results of the Application View showed that it is very difficult and maybe even impossible to abstract a service definition of all the requirements the different applications need, the division of the required services remains useful. The services are divided into three groups;

- Connection Oriented Network Services (CONS) X.213/ISO 8348
- Connection-Less Network Services (CLNS) ISO 8348ad1
- NON-OSI Services

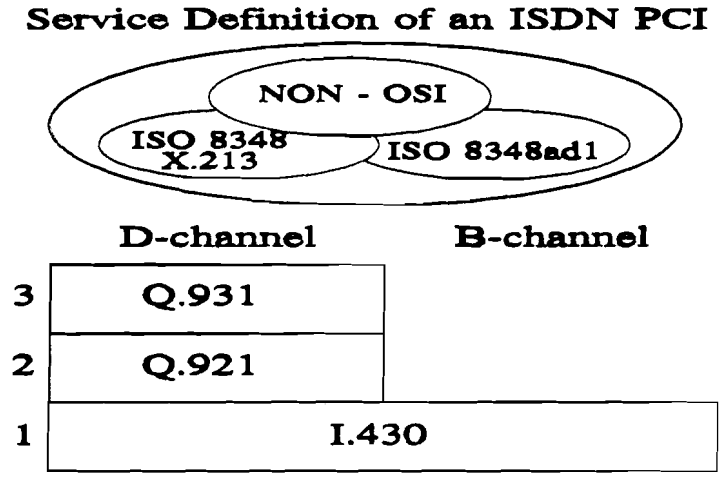


Figure 4.3: *Possible conceptual model of a general ISDN PCI*

The results of the Network View show that the model of an ISDN PCI will be a service definition in OSI-terms, based on the ISDN Reference Model. This service definition has to be a functional superset of all the existing service definitions for applications that are applicable on an ISDN and has to offer the functionality for Non-OSI applications.

Combining these results into one model a picture of a 'proposed conceptual model can of an ISDN PCI' can be drawn as shown in figure 4.3.

4.7 Summary

This chapter describes a proposed conceptual model of an ISDN user-network interface. Both the required services of applications and the provided services by the network are taken into account.

Integration of the 'computer-communication world' and the 'telecommunication world' introduces great complexity and solvation of this problem requires further study. The absence of a stable service definition of the Q.931 protocol [9] and the developments of the lower layer D-channel protocols complicates matters even more.

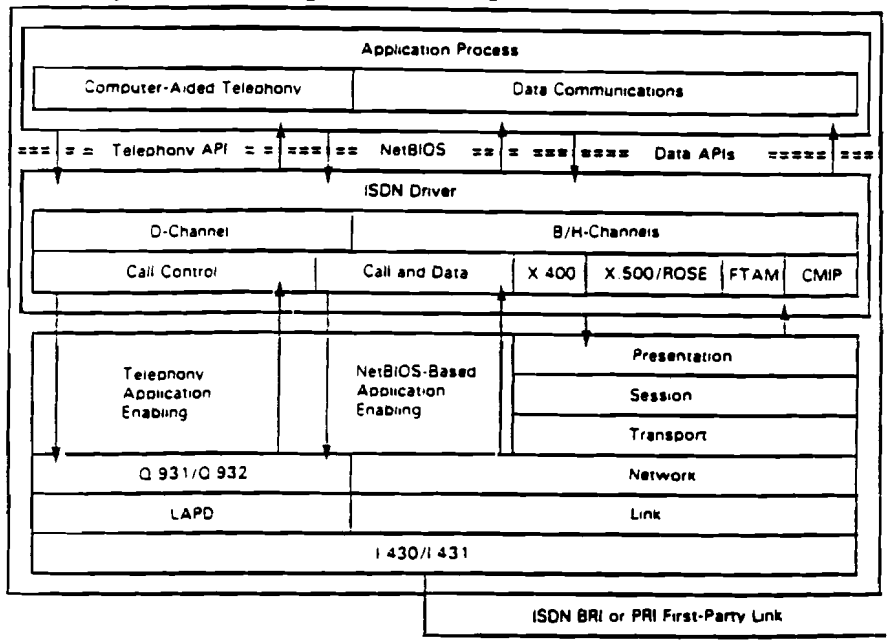


Figure 4.4: *IBM's vision on API's*

The actual situation is that a lot of manufacturers bring their own API or PCI onto the market and push them to becoming a 'de facto' standard [29,30]. These APIs or PCIs often provide only parts of the available services. IBM for example has the philosophy of separately developing "Telephony API's" and "Data-API's", which are placed on top of layer 7, the application layer. In the end when both API's are fully developed and their specifications are known, a common API can be developed offering the functionality of both the "Telephony API" and the "Data-API". Figure 4.4 shows IBM vision on APIs [29].

Presently work is done in standardization committees such as CCITT SG8 and ETSI TE7 to produce a recommendation or a standard conceming an ISDN PCI but this might be an perpetual task considering the effort needed to provide service to all present and future applications. Maybe IBM's view of developing separate API's and integration if both are evaluated might be the best solution.

Chapter 5

Some security aspects of an ISDN

5.1 Introduction

The ISDN concept leads to a structure characterized by digital communication systems, supporting multi access terminals and an increasing number of databanks. More and more the ISDN will be used to process information that is sensitive to an individual, a company or society. This calls for security measures. Services which are not mentioned in the previous chapter. Within ISO and CCITT work is done to standardize these security measures. Some of the measures that are worked on are [13];

- Authentication
- Access Control
- Data Confidentiality
- Data Integrity
- Non-repudiation

Each of these services offer a different kind of security. It is beyond the scope of this report to deal with a subject like security thoroughly. A technique that is very useful in providing these services is encipherment of data. Encipherment, for example, is one method to offer Data Confidentiality. This chapter will only deal with a method of encipherment and some of the services in an ISDN on which it is applicable

An interesting question is what kind of services an ISDN has to offer concerning security. Encipherment is, next to a kind of data confidentiality, also useful as a sub-service to provide more complex security services, such as non-repudiation. If this question is answered, the next question:'How to offer these services', occurs.

Encipherment can be applied in several layers of the OSI Reference Model. As already discussed in the previous chapter, the ISDN Reference Model differs substantially from the OSI Reference Model. Still it is possible to look upon the B-channel as an OSI-stack. The services an ISDN provides can be divided in Bearer Services and Tele Services [3,4]. This chapter concentrates on the Bearer Services, which will be described in the first section.

Within the Research Neher Laboratory a system is developed called TIRO. TIRO stands for Temporal Input Random Output. The system consists of modules which provide the possibility to encrypt and decrypt data and of a management system which provides the management of the keys and the modules. This system is globally discussed in the second section.

The last section describes the possibilities of integrating a system like TIRO in an ISDN.

5.2 Bearer Services in an ISDN

It is possible to divide the telecommunication services provided by an ISDN into Bearer Services and TeleServices

The bearer services provide the possibility for the exchange of information between two points. Teleservices offer the service requested by the user. This section only deals with the bearer services only. The bearer services can also be divided into two different types;

- Packet mode services
- Circuit mode services

Within those types there are different categories of services.

Packet mode service categories;

1 Virtual call and permanent virtual circuit
2 Connectionless
3 User signalling

Circuit mode service categories;

1 64 $^{kbit}/_{s}$ unrestricted, 8 kHz structured
2 64 $^{kbit}/_{s}$, 8 kHz structured, usable for speech information transfer
3 64 $^{kbit}/_{s}$, 8 kHz structured, usable for 3.1 kHz audio information transfer
4 Alternate speech / 64 $^{kbit}/_{s}$ unrestricted, 8 kHz structured
5 2 X 64 $^{kbit}/_{s}$ unrestricted, 8 kHz structured
6 384 $^{kbit}/_{s}$ unrestricted, 8 kHz structured
7 1536 $^{kbit}/_{s}$ unrestricted, 8 kHz structured
8 1920 $^{kbit}/_{s}$, 8 kHz structured

The packet mode bearer service categories 2 and 3 are under study. At this moment it is clear that service 1 will be implemented. At this time it is possible to communicate with X.25 though the kind of bearer service used is a circuit mode 64 $^{kbit}/_{s}$ unrestricted, 8 kHz structured connection to an Access Unit (X.31 case A) or between two users. In future the network will offer a packet switched connection. In that case communication is possible according to X.31 case B.

The focus on the user interface of an ISDN in this report causes only interest in the basic rate bearer services. Furthermore only services 1, 2 and 3 are marked as ''essential bearer services'', all other services are marked as ''additional bearer services''. Hence, only the essential bearer services are described.

Circuit-mode, 64 $^{kbit}/_{s}$ unrestricted structured bearer service category

This bearer service category provides unrestricted information transfer between S/T reference points. It may, therefore, be used to support various user applications. Examples include;

- speech
- 3.1 kHz audio
- multiple subrate information streams multiplexed into 64 $^{kbit}/_{s}$ by the user
- transparent access to an X.25 public network (X.31)

User information is transferred over a B-channel, signalling is provided over a D-channel.

Circuit-mode 64 $^{kbit}/_{s}$, 8 kHz structured bearer service category usable for speech information transfer

This bearer service category is intended to support speech. The network may use

processing techniques appropriate for speech such as analogue transmission, echo cancellation and low bit rate voice encoding. Hence, bit integrity is not assured. This bearer service is not intended to support modem derived voice-band data.

Circuit mode 64 $^{kbit}/_{s}$, 8 kHz structured bearer service category usable for 3.1 kHz audio information transfer

This bearer service category corresponds to the service which is currently offered in the PSTN. This bearer service category provides for the transfer of speech and of 3.1 kHz bandwidth audio information such as voice-band data via modems and facsimile group 1,2 and 3 information.

Connections provided for these services should offer the transfer capability for the information indicated above. (This means that the network may include speech processing techniques provided they are appropriately modified or functionally removed prior to non-speech information transfer.) The control of echo control devices, speech processing devices, etc is only made by the use of disabling tones (See recommendation V.25). Bit integrity is not assured. The network may use analogue transmission.

5.3 The TIRO System

The TIRO system is developed by the PTT Netherlands and designed to secure end-to-end communication with use of the ROAL (Random Output Algorithm) encryption algorithm. For this purpose two entities are introduced;

- TIRO module, which provides the securing
- Tiro Management System (TMS), which provides the management of the keys and the TIRO modules

Each workstation that wishes to use the end-to-end securing has to be connected to the network by a, through the TMS initialized, TIRO module.

An ASIC is used in the implementation of the system. Hence the encryption and decryption is performed by the hardware with a maximum of 2 Mbps. Another advantage is the storage of a key inside the ASIC. The hardware is designed in a way that reading the key-information in impossible. The initializing of the key is done by the TMS in a save environment. This strengthens the security of the system.

The TIRO Management System (TMS) provides furthermore in the management of the keys. This keys and their use form the strength of the system. Keys are used for; Authentication, Data Encipherment and communication between the TMS and a Module.

The keys are contained in a very strong hierarchy. The transmission and storage of keys always happens with encryption with a key that is one step above in the hierarchy. Since there is no key in the hierarchy that is above the highest key, this key cannot be treated in the same manner. Therefore this key is physically distributed and physically protected inside the TIRO modules.

A key is used in two ways. First they are used to transmit other keys or data. Secondly, the keys are used for implicit authentication on the basis of synchronisation.

It is possible to create closed usergroups within the TIRO system. The members each have a special key. Within those groups key exchange is possible. For example the lowest keys in the hierarchy, used to encrypt the data, are only valid for a period of time. After this time they are replaced by another key that is generated and sent to the communication partner encrypted with the key one step higher in the hierarchy.

5.4 Use of TIRO in an ISDN

As both the bearer services of an ISDN and the mechanism of the TIRO system are described, now it becomes interesting to examine if the TIRO encipherment can be used within an ISDN especially with regards to the described bearer services.

First must be stated that application of the complete TIRO system, together with the TIRO Management System (TMS) in an ISDN is not examined. Only the encipherment part plays a role in this discussion. Hence when mentioned TIRO, not the TMS is meant.

At this moment a TIRO-X.25 system is developed by the Research Neher Laboratory. This system places the encipherment on top of the network layer. In that case the enciphered data is transmitted transparently through the network. This system is also applicable in an ISDN environment because the B-channel can be looked upon as an OSI stack. Note that the use is possible in both X.31 case A and case B scenario's.

The use of TIRO requires bit integrity. Otherwise the encrypt and decrypt algorithms lose their synchronization and data is lost. Therefore TIRO can only be applied in the first essential bearer service, because this is the only one assuring bit integrity.

In this case the encipherment entity is placed on top of layer 1, but on the users premises' side of the S-reference point. Hence end to end data-confidentiality can be provided.

5.5 Summary

This chapter deals with the integration of the TIRO system in an ISDN. The TIRO system is used in the ISDN bearer services. Therefore first the essential bearer services are described and the TIRO system is discussed.

It is possible to provide 'TIRO end-to-end security' for the packet mode beare service category 'virtual call and permanent virtual call' and for the circuit-mode, 64 $^{kbit}/_{s}$ unrestricted structured bearer service category. The first requires the use of the TIRO-X.25 system and adapted to an ISDN and the latter requires a special module, called the *'TIRO_B-Channel_Encryption (TBCE) Module'*.

The rest of this report deals with the design of this module which allows the use of the TIRO encipherment within a Siemens ISDN Basic Rate Access Configuration. The next chapter describes the hardware environment in which the TBCE Module has to be applicable. Then, the following chapter deals with the actual design of the TBCE Module.

Chapter 6

The Hardware Environment

6.1 Introduction

This chapter describes the hardware environment which forms the basis for the development of the TIRO_B-Channel_Encryption (TBCE) Module. The main component is a Siemens ISDN Evaluation Kit. This kit consists of a main board with 6 slots, 3 inside and 3 outside the PC, on which modules can be plugged. These modules performed the different functions, for example layer 1 and layer 2 modules.

Siemens has developed a concept called the ISDN Oriented Modular (IOM) Interface. The interface consist of two generations, IOM1 was only a Siemens initiative but IOM2 is supported by other companies as well. Communication between the modules and the main board and their mutual communication takes place over this interface. Its specification is described in the first part of this chapter.

The add-on modules are designed around a certain Siemens ICs. This IC forms the main functional component of the module. For example, the layer 1 module perfoms typical layer 1 functions and is designed around the S/T Bus interface circuit (SBC PEB 2080). The layer 2 module is designed around the ISDN Communications Controller (ICC PEB 2070). Also a S-Access module exists with the ISDN Subscriber Access Controller (ISAC-S PEB 2085). The second part of this chapter discusses the previous mentioned Siemens IC's.

The integration of the TBCE Module is described in the last part of this chapter. Figures are used to clarify the location and hardware configuration of the TBCE Module.

6.2 ISDN Oriented Modular Interface

6.2.1 Introduction

This section describes the ISDN Oriented Modular (IOM) Interface. This interface consists of two generations, the IOM1 interface and the IOM2 interface. The IOM2 interface is an expanded version of the current IOM1 interface. Therefore functional compatibility between IOM1 and IOM2 devices is retained.

The IOM interface is a standard 4-wire local interface for the interconnection of ISDN devices within the ISDN basic access. It consists of a receive and a transmit dat line, an 8 kHz frame signal and data clock signal for frequencies ranging from 512 kHz to 8192 MHz. The interface is designed to fit the needs of the ISDN basic access as defined by CCITT.

The IOM1 concept is an initiative of Siemens. IOM2 however is an initiative of the 'group of four'. This 'group of four' stands for four companies namely Alcatel, Itatel, Siemens System House and Plessey. The IOM2 interface is then also called General Circuit Interface (GCI). The name IOM2 is derived from the fact that the IOM1 concept fitted exactly in the concept.

This section describes only the global specification of the IOM2 interface, because the IOM1 interface can be seen as a part of the IOM2 interface and it is not necessary for the comprehension of the reader to understand the details of the interface. The interested reader can find the detailed specification in the Siemens document: "ISDN Oriented Modular Interface Specification Rev.2.2 - IOM™ 2nd Generation - 9/88"

6.2.2 Scope of the IOM interface

The IOM interface is a standard 4-wire local interface for the connection of ISDN devices within the basic access. The IOM2 interface is an expansion of the IOM1 interface. In addition to the IOM1 interface the IOM2 interface has new characteristics designed to assure greater flexibility such as:

- Use of C/I codes for maintenance functions
- More flexible multiplexed mode
- Monitor channel transmission protocol

These characteristics, which will be clarified later, make the application of the IOM2 interface possible on different fields in the ISDN Reference Model as figure 5.1 shows.

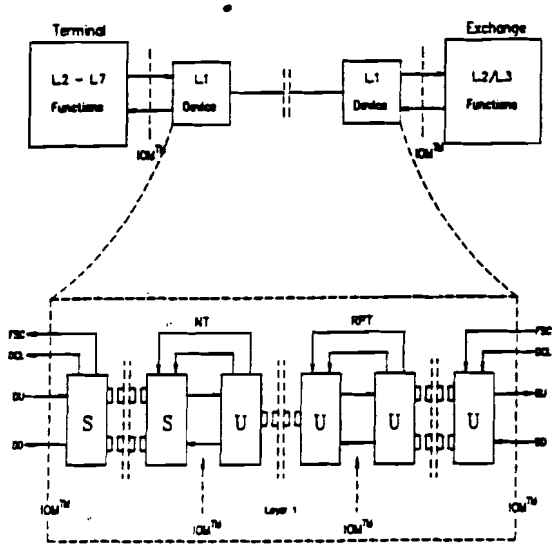


Figure 6.1: *Application of the IOM2 interface in the Basic Access*

The applications for the IOM2 interface as illustrated in figure 6.1 are:

- In the terminal and the exchange, where the IOM2 interface interlinks the layer 1 and layer 2 devices. In addition to the point-to-point configuration, the IOM2 interface can support a point-to-multipoint configuration for connecting different IOM2 (sub)channel link/source-devices to one pair of data lines by wired-OR technique. This is primarily intended for terminal applications.

- In the network termination (NT1), where two different transmission devices are used to convert layer 1 between the 4-wire subscriber interface (S/T) and the 2-wire public telephone lines (U).

- In a repeater, where two identical transmission devices are combined, back-to-back, to achieve an increase in range.

Furthermore the IOM2 interface can also be used on analog line cards, where one IOM channel can serve two lines.

6.2.3 IOM2 Interface Signals

The IOM2 interface consists of four physical connections. Two lines allow for the serial transmission of data in both transmit and receive directions. The other two lines are for the frame and data clock signals (figure 6.2)

- DU *Data Upstream.* Transmission of data from the subscriber side to the exchange side.
- DD *Data Downstream.* Transmission of data from the exchange side to the subscriber side.
- DCL *Data Clock.* Twice the transmission frequency on DU and DD.
- FSC *Frame Synchronization Clock.* Resets the bitcounter at the start of each frame.

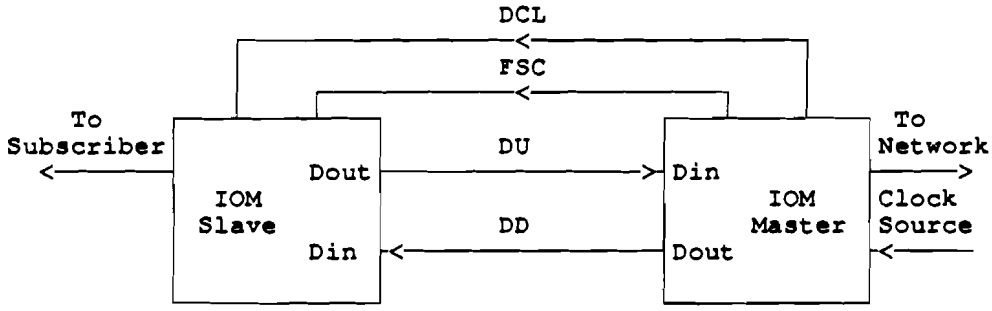


Figure 6.2: *Interface Signals for IOM2 Interface*

The downstream and upstream directions are always with respect to the exchange. Downstream refers to the information flow from the exchange side of the interface to the subscriber side. Upstream refers to transmission in the opposite direction. For the device point of view, these connections can also be named Din and Dout, referring to the reception or transmission of data, respectively.

The IOM2 interface is synchronized by frame and data clock signals supplied by a master clock source. A device used in the master mode can deliver the clock signals required for synchronization either directly or via a PLL. FSC is defined as the system clock used as reference for all PLL functions. Internal device clocks should be independent of the data clock.

6.2.4 IOM2 Interface Frame Structure

In all applications, the ISDN basic access rate of 144 kbit/s is transferred transparently across the IOM2 interface. In addition, there is a need for the interchange of control information for activating/deactivating layer 1 and for switching test loops. Most applications also require further capacity for the transfer of maintenance information over this interface. All of this information is transferred in a time-division multiplexed mode based on an 8 kHz frame structure (figure 6.3) and assigned to the following four octets per frame and direction:

- 64 kbit/s B channel (1) in the first octet
- 64 kbit/s B channel (2) in the second octet
- The third octet (monitor channel) is used for transferring maintenance information between the layer 1 functional blocks (SBC,IBC,IEC) and the layer 2 controller.
- The fourth octet (control channel) contains,

  in case of ISDN lines:

  * two bits for the 16 kbit/s D channel
  * four command/indication bits C/I (4:1), for controlling activation/deactivation and for additional control functions.
  * two bits MR and MX for supporting the handling of the monitor channel.

  in case of analog lines:

  * six command/indication bits C/I (6:1) for controlling the CODEC/SLIC functions.
  * two bits MR and MX for supporting the handling of the monitor channel.

These four octets per frame call for a bit rate of 256 kbit/s. The interface can also be used in a multiplexed mode for up to eight subscribers. In this case, the data transmission rate over the interface is dependant on the data clock, DCL. Typically, the clock rate for the multiplexed mode would be 4096 MHz, giving the standard bit rate of 2048 Mbit/s. For special applications, bit rates up to 4096 Mbit/s, half the maximum clock rate are possible.

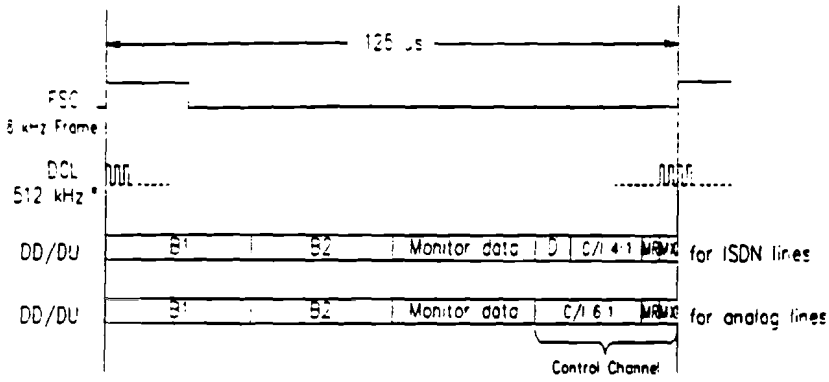


Figure 6.3: *Frame Structure of the IOM2 Interface*

data rate	:	256 kbit/s
bit rate	:	256 kbit/s - 4096 Mbit/s
clock frequency	:	512 kHz - 8192 MHz

For the multiplexed mode, each subscriber connection consists of one block of four octets and is called an IOM2 channel with a corresponding number (CH0 through CH7) assigned to it.

6.2.5 IOM2 Interface Protocol

6.2.5.1 B and D Channels

In the activated state, the two B and the D channel are normally transmitted transparently across the IOM2 interface.

In S/T interface terminals, the well defined D channel access procedure (cf. CCITT recommendation I.430) requires some signalling between layer 1 and layer 2 devices. In case of collision at the S/T interface or when waiting for the idle state, the layer 1 device stops transmission of D-bits.

6.2.5.2 Monitor Channel

The monitor channel is used for the transfer of maintenance information between two ISDN basic access functional blocks. If a management block is available for managing the different IOM devices, the monitor channel may be used for writing/reading internal registers of layer 1 devices.

The monitor channel is operated in a full duplex manner. By use of two monitor control bits (MR, MX) per direction, the data are transferred in a complete handshake procedure. The MR and MX bits in the fourth octet (control channel) of the IOM2 frame are used for the handling of the monitor channel.

6.2.5.3 Command/Indication Channel

The ISDN basic access is activated on a "call-by-call" basis, indicating that the transmission line as well as the terminal equipment are in a "power down" state as long as no transmission is done. When a transmission between terminal equipment (TE) and line termination (LT) is initiated, the transmission line- consisting of one or more sections- between TE and LT must be activated.

The necessary activation/deactivation procedures have been specified in the CCITT recommendation I.430. The procedure is always initiated at one of the end points, whereby control information must be sent along the subscriber lines and the IOM2 interface. For the IOM2 interface, this information is contained in the command/indication (C/I) channel.

The exchange of information in the C/I channel takes place according the C/I channel protocol. If a layer 1 device receives a new code it will interpret the code and send control information on the transmission line or active a control function (e.g. switch a test loop). A layer 2 device, on the other hand, will respond to a change in the C/I code by an interrupt request to the processor.

6.3 The Siemens ISDN Components

As already mentioned in the introduction of this chapter, the hardware is developed around a Siemens ISDN Evaluation Kit. This kit consists of a main board with slots on which modules can be plugged. These modules provide different functions. Each module has a main component. For example the S/T Bus Interface Circuit (SBC) is the main component of the Layer 1 Module. Due to the fact that the rest of the hardware on the modules add no functionality, only the IC's are discussed. The timing characteristics of the IC's are contained in appendix B.

6.3.1 S/T-Bus Interface Circuit

The S-Bus Interface Circuit (SBC PEB 2080) implements the four wire S/T interface

used to link voice/data terminals to an ISDN. The device provides all electrical and logical functions according to CCITT Recommendation I.430. These include:

- Mode-dependent receive timing recovery
- D-channel access and priority control
- Automatic handling of activation/deactivation procedures

Through selection of operating mode, the device may be employed in all types of applications involving an S-interface (TE,NT,LT-S,LT-T). Two or more SBC's can be used to build a point-to-point, passive bus, extended passive bus or star configuration.

The SBC does not require direct microprocessor control. This is due to the fact that the IOM interface provides all the necessary functions for layer 1 - layer 2 communication. The SBC provides conversion between S/T frames and IOM frames.

This frame conversion only converts S/T frames to IOM-1 frames (DCL= 512 kHz). The successor of the S/T-Bus Interface Circuit, the S-Bus Interface Circuit Extended (SBCX PEB 2081) does provide frame conversion between S/T frames and IOM-2 frames. By provision of some additional features at the IOM2 interface the use of the SBCX can be combined with other IOM-2 devices in various configurations. Timing characteristics can be found in appendix B.1

6.3.2 ISDN Communication Controller

The ISDN Communication Controller (ICC PEB 2070) supports the LAPD protocol and acts as the D-channel-link-access protocol controller. The ICC consists of serial interface logic for the IOM and the SLD and SSI interfaces, with B-channel switching capabilities and Logic necessary to handle the D-channel messages (layer 2).

The IOM interface logic allows interaction between layer 1 and layer 2 functions. It implements D-channel collision resolution for connecting other layer 2 devices to the IOM interface, and the C/I and monitor channel protocols to control peripheral devices.

The ISDN Communication Controller can be used in different modes of operation;

- IOM1 Mode
- IOM2 Mode
- HDLC Controller Mode

In IOM1 Mode two additional serial interfaces are available besides the IOM1 interface. The Synchronous Serial Interface (SSI) and Subscriber Line Datalink (SLD) interface. These interfaces are used to connect B-channel devices. The IOM1 interface is used to connect the ICC to a layer 1 component.

In IOM2 Mode the IOM2 interface is used for connection of layer 1 devices and as a general purpose backplane in bus terminal equipment. The auxiliary serial SSI and SLD interfaces are not available in this case. Appendix B.2 contains the timing characteristics.

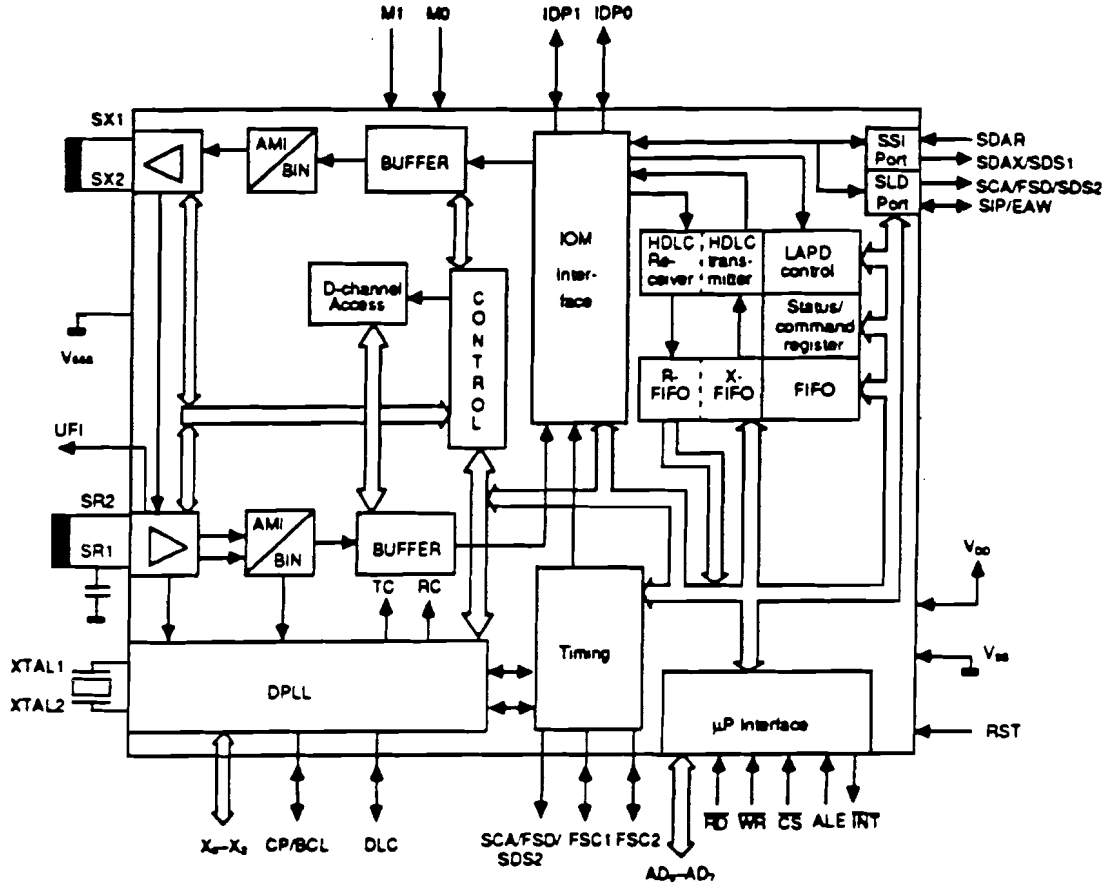


Figure 6.4: *Architecture of the ISAC-S*

6.3.3 ISDN Subscriber Access Controller

The ISDN Subscriber Access Controller (ISAC-S PEB 2085) combines the functions

of the S/T Bus Interface Circuit and the ISDN Communications Controller on one chip.

Both the S/T Bus Interface Circuit and the ISDN Communications Controller are previously discussed. Figure 6.4 shows the functional block diagram of the ISAC-S.

The functional block diagrams of the SBC and the ICC are easy to recognize in the figure. The left-hand side of the diagram contains the layer-1 functions according to CCITT I.441, and thus resembles the functional block diagram of the SBC. The right hand side consists of the serial and D-channel logic mentioned in the previous section which discussed the ICC. Timing characteristics of the ISAC-S can be found in appendix B.3.

6.4 Integration of the TBCE Module

This section describes the location of the TIRO_B-Channel_Encryption (TBCE) Module within the previously described hardware environment.

For the realisation of the hardware, a XILINX Field Programmable Gate Array (FPGA) will be used. Within this FPGA all the necessary logic can be implemented. The actual encryption and decryption of the data will be executed by the TIRO-ASIC.

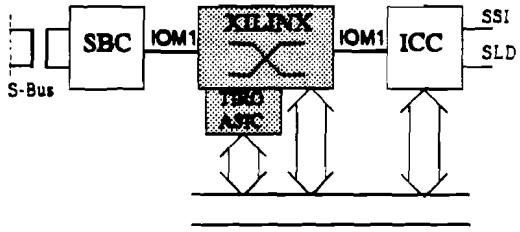


Figure 6.5: *Integration TBCE Module in IOM1 Interface*

The module has to be applicable in both the IOM1 and IOM2 interface though the situation is different. Figure 6.5 and 6.6 show the new IOM1 and IOM2 hardware configurations of the Siemens ISDN components

With this equipment a 'hardware model' of the TBCE Module can be constructed which is shown in figure 6.7 and is applicable in both situations.

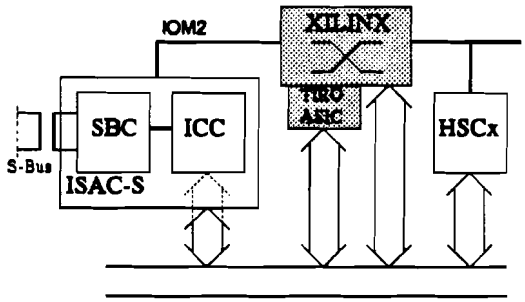


Figure 6.6: *Integration TBCE Module in IOM2 Interface*

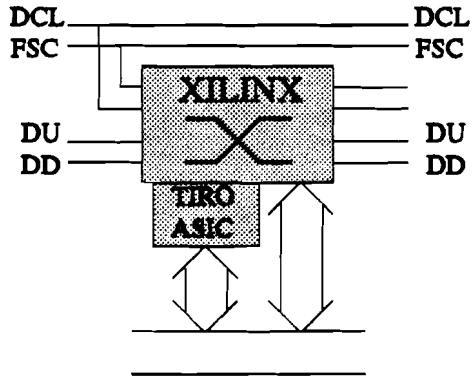


Figure 6.7: *'Hardware model' of the TBCE Module*

6.5 Summary

The goal of this chapter is to describe the hardware environment of the TIRO_B-Channel_Encryption Module. A global specification of the ISDN Oriented Modular (IOM) interface, an interface for the interconnection of ISDN devices is given in the first section. The Siemens ISDN Evaluation Kit containing the Siemens ISDN chipset uses this IOM interface. The S/T-Bus Interface Circuit, ISDN Communications Controller and the ISDN Subscriber Access Controller, are also described.

A 'hardware model' of the TIRO_B-Channel_Encryption Module is constructed in the last section. This model is applicable in both IOM1 and IOM2 situations. The next chapter will deal with the design of this module.

Chapter 7

The Design of the TBCE Module

7.1 Introduction

This chapter describes the design of the TIRO_B-Channel_Encryption Module. The environment in which it is applicable is described in the previous chapter. Because the difference between the IOM1 and IOM2 mode is caused by variation of the DCL and the FSC, it is possible to design one module for both IOM1 and IOM2 devices.

The Siemens ISDN components, discussed in the previous chapter, form the basis of the ISDN system. The TIRO-ASIC is, of course, also a necessary component. Furthermore, a XILINX Field Programmable Gate Array (FPGA) will be used to design the controlling logic.

The next section formulates the design requirements, followed by a section which deals with the design in functional blocks.

7.2 Design Requirements

The TBCE Module is placed in an ISDN basic access configuration constructed with the Siemens' ISDN chipset. Its function is to encrypt or decrypt a given slot in an IOM frame. This means that the Module has to take the data out of the IOM frame, lead it through the ASIC and put it back in an IOM frame again. This relates to both IOM1 and IOM2 frames.

Furthermore the user has to be able to decide which slot has to be enciphered. This means that there has to be an user access facility and also a facility to store the data or parameters that the user provided. The storage will take place in some registers and they are accessed via a microprocessor interface.

To keep the connection confidential, key changes within certain periods of time are required. The TBCE Module is placed in a synchronous interface, which complicates the key changing. This is caused by two reasons. First, data can't be buffered and second, the specifications of the IOM interface cannot and may not be violated. The latter reason has two possible solutions. One is to take out only the user data and delay this slot one frame every time. In this delay time it is the possible to encipher the buffered user data. This is possible because the delay is only one frame every time, so the buffer can't be overloaded. The second solution is to take out the user data, encipher it very quick and put the enciphered data back into the same frame.

In appendix C the relation between the timing of the IOM interface and the TIRO ASIC is described. The TIRO-ASIC needs 500 ns to encrypt a bit. The conclusion of this appendix is that in IOM1 mode 1700 ns are available to encrypt one bit and in IOM2 TE-mode this is 975 ns, so it has to be possible to encrypt and decrypt the data within one frame and the IOM interface will not be influenced

It is impossible to change keys in the system without buffering a lot of data. Furthermore synchronization is necessary, otherwise at least 96 bits will be lost. A solution in this case is to apply two ASIC's. In this way it is possible to use the active ASIC to encipher the data and the other one can be parallel installed with new keys. A way has to be found now to synchronize both communication partners in such a way that they can switch at the same time from active ASIC, otherwise still user data is lost.

With this knowledge it is possible to formulate a set of design requirements:

- The TBCE Module is placed inside a Siemens ISDN System
- The TBCE Module has to be both IOM1 and IOM2 compatible
- The TIRO-ASIC is used for the enciphering at 2 Mbit/s
- It has to be possible to encipher an arbitrary slot in an IOM frame
- The TBCE module has to have a microprocessor interface.

7.3 Design in Functional Blocks

With the formulated design requirements it is now possible to create a design in functional blocks. The TBCE Module has to contain multiplex units and a microprocessor interface and also some registers in which the user data can be stored.

The TIRO-ASIC operates at a clockfrequency of 16 Mhz. The signals DCL and FSC are tapped of the IOM interface signals. The Data Downstream (DD) and Data Upstream (DU) lines are physically interrupted. The DD is data from the network to the subscriber and has to be decrypted, so this line is connected to the decrypt part of the TIRO-ASIC. DU is data from subscriber to network and has to be encrypted so this line is connected to the encrypt part of the TIRO-ASIC. The design in functional blocks is pictured in figure 7.1

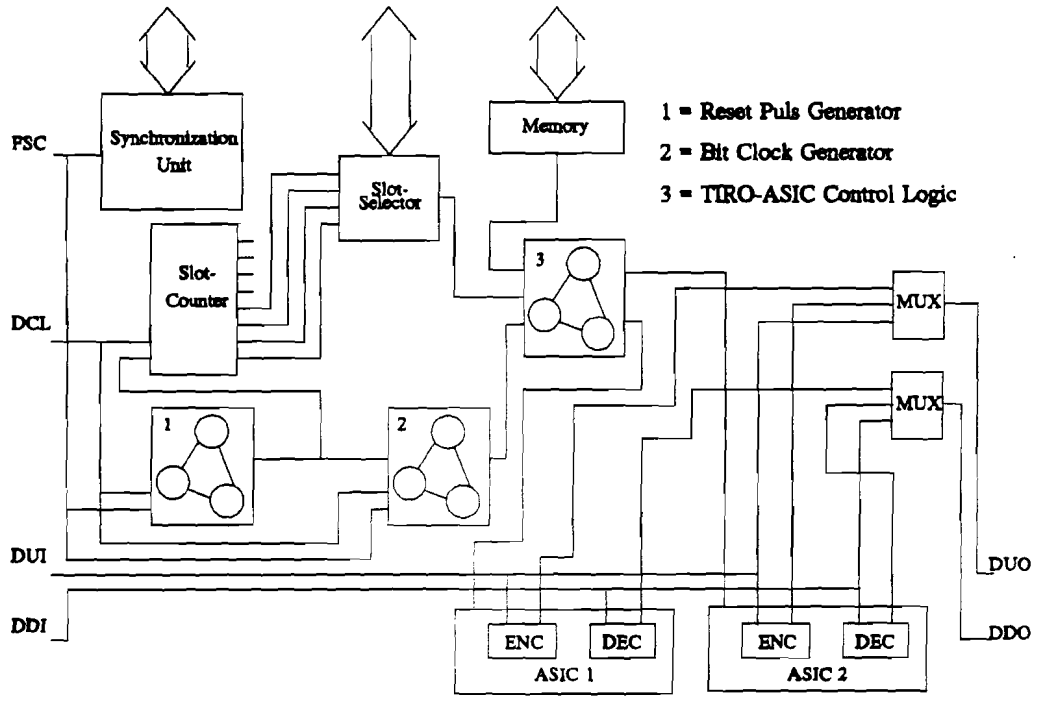


Figure 7.1: *Functional Block Design of TBCE Module*

7.3.1 Slot counter

In the IOM specification is given that DCL has a frequency twice the bitrate. This means that a counter which is clocked by DCL counts 2 times for every bit. A slot consists of 8 bits, so 16 DCL periods. The IOM2 TE-mode frame consists of 12 slots. The counter clocked by DCL has to count at least until 191. This can be realised with a 8 bit counter and use only of the 4 most significant bits. It then counts every 16 DCL edges and thus every slot. When the last slot has passed, the counter has to be reset. In IOM1 mode this happens after 4 slots and in IOM2 mode after 12 slots.

7.3.2 Slot selector

The slot selector gets a slot number as input. The outputsignal of the slot selector decides whether or not the actual slot (bit) has to be enciphered. Furthermore it has to be possible for the user to program which slots have to be enciphered.

The slot selector contains 2 registers of 8 and 4 bits. Via the microprocessor interface it is possible to write a pattern of 12 bits into the registers. These registers are bound to two multiplexers, one 8-to-1 and an 4-to-1 multiplexer. The output of those multiplexers is again input for a 2-to-1 multiplexer. The multiplexers are enabled by the slot counter. The 2-to-1 multiplexer is driven by the most significant bit of the counter. The previous multiplexers are driven by the 3 and 2 least significant bits of the slot counter.

The outputsignal of the slot selector is active low and is called 'encrypt'. A picture of the slot selector is contained in appendix D

7.3.3 Reset Puls Generator

The Reset Puls Generator synchronizes the Slot Counter and the Bit Clock Generator. The slot counter has to be reset at the start of every new frame. Input of the Reset Puls Generator are FSC and DCL. Because their mutual position can differ from time to time, the realisation of the Reset Puls Generator has to be done with a state machine.

This state machine keeps track of the rising edge of FSC. If this has occurred and also a rising edge of DCL occurs, a reset puls is given. In this case the rising edge of FSC occurred first. If the DCL level is high and a rising edge of FSC occurs, the also a

reset puls is given, because in that case the corresponding DCL edge occurred sooner than the rising edge of FSC. In appendix E the state diagram of the state machine is given.

7.3.4 Bit Clock Generator

Because the frequency of DCL is twice the bitrate, it is not clear at which rising edge the data on the datalines alters. The Bit Clock Generator divides the DCL by two and has a rising edge corresponding with every rising edge of DCL at which the data alters.

The synchronization of both rising edges is done with the use of the reset puls signal, which always occurs after a rising edge on which the data alters. So synchronization takes place with every new frame. The state diagram of the Bit Clock Generator is given in appendix F.

7.3.5 TIRO- ASIC Control Logic

The function of the TIRO-Asic Control Logic is to provide the ASICs of a clocksignal at the right time. It is realised in a state machine that starts its cycle on a rising edge of BCL. After this rising edge some wait states are implemented to wait until the data is valid and the switches are set. At that time the TIRO-ASIC Control Logic decides on the values of the encrypt signal and the asic_nr signal if this slot is to be enciphered and by which ASIC.

The output of the TIRO-ASIC Control Logic is a sequence of eight clock pulses on the clock pin of one of the two ASICs if and only if the actual slot had to be encrypted or decrypted. After that the state machine waits for BCL to go low and returns into the IDLE state. The state diagram is given in appendix G.

7.3.6 Multiplex Unit(s)

The TBCE Module contains two Multiplex Units, one for each direction. A Multiplex Unit is nothing more than a 3-to-1 multiplexer, which in practice is a 4-to-1 multiplexer with one input on two entries. This multiplexer is driven by the signals encrypt and asic_nr. If encrypt is low the outputsignal is defined by the asic_nr signal. Otherwise the data is led through the TBCE Module transparently.

The signal on the multiplexers are from input one to four: Outputdata from ASIC1, Outputdata from ASIC2 and two times the inputdata (DD or DU).

7.3.7 Synchronization Unit

For key changing it is necessary that both communicationpartners process their data synchronous. Because the slot counter only has a period of 125 microseconds, a frame counter is added. In this way a larger margin is created. It is a simple counter riven by FSC.

Additional an "Auto Sync Unit" could be added. This Unit resets the frame counter when a match occurs between a preloaded pattern in a register and the synchronous data stream. If this pattern is the same at both ends of the connection, both frame counters will be reset at the same point in the datastream and will count synchronous from that point. The frequency of this reset signal depends on the length of the preloaded pattern. The length of the preloaded pattern determines also the chance on a 'false reset' when a bit error on the line occurs and the patterns match only on one side.

7.3.8 Memory Unit

The Memory Unit consists of a 4-bit register which can be accessed via the microprocessor interface. One bit is used to store the number of the active ASIC. The other bits are free to use in future enhancements, for example to choose from different preloaded patterns in the synchronization unit.

7.4 Results and Evaluation

The design is implemented in an XILINX FPGA. The detailed implementation is given in appendix H. This implementation has been realised two times and is placed in the point-to-point connection between the TE and the NT of the Siemens Evaluation Kit.

Because of the available hardware only the IOM1 system is tested and worked. The test was performed with a Siemens demonstration software tool. It is possible to encipher the voice connection and data connection separately and also simultaneously.

The IOM2 hardware is just shortly available and therefore not tested. The hardware came available It is necessary to develop some testsoftware with which predefined data can be sent in the B-channels for both IOM1 and IOM2 situations. Only in that way it is possible to clarify the relation between the plain_data_sent, encrypted_data_sent, encrypted_data_received, plain_data_received. This relation is very important because it can provide proof there is no data lost.

The Synchronizing Unit has to be developed in detail. At this moment only a frame counter is implemented. The already mentioned 'Auto-Sync' principle can be added to this, but also the way the synchronization takes place has to be determined. This is also relation to the frequency at which key changes occur.

Chapter 8

Conclusions and Evaluation

This chapter consists of conclusions and evaluations of the subjects dealt with in this report. First the conclusions and evaluation about the flexible interfaces are dealt with and secondly the conclusions and evaluation about the hardware part are described.

8.1 Conclusions and Evaluation concerning Flexible Interfaces

The first part of this report describes the flexible interfaces in LAN environment and in ISDN environment.

Concerning the interfaces in LAN environment can be said that at this moment both products try to become a 'de facto' standard and it is to be expected that one of them will become that status because their functionality is similar. Another option is the creation of compatibility between both products.

The Integrated Services Digital Network Programmable Communications Interface, shortly ISDN PCI, discussed in chapter 3 is a first attempt to standardize the user-interface of an ISDN which provides access to multiple services. This standardization will cost much effort because no defined model is available for such a flexible user interface. An attempt is made to derive a model from the previous COMMON ISDN API which was more or less a product specification. This model however is at this moment point of discussion in ETSI and CCITT meetings and is not complete.

The requirements of the user interface in chapter 4 point out that it is very difficult to create a general model for an ISDN PCI. The fundamental differences between the particular 'datacommunication' and 'telecommunication' services cause a great problem. The found model of an ISDN PCI has to be a service description in OSI term

of the complete ISDN reference model. This specification has to contain a set of primitives and a description of procedures specifying which primitives to use in what sequence to get a certain service. It is obvious that such a service specification forms a functional superset of all existing service specifications.

Considering the results of in chapter 4 the initiative of the german ISDN PCI fits in the model. Though it consists of a set of "primitives" and "procedures" in an OSI-like style, it is only a part of the model. It does not contain the entire collection of provided services. Supplementary services for example, is an important missing item.

Because of the major differences between the 'datacommunication world' and the 'telecommunication world', which are elucidated in the almost incompatibility of both reference models, it is worthwhile considering if the standardization of an ISDN PCI for all services is a realistic idea. Many manufacturers develop of different API's or PCI's for those services. IBM for example develops "Data-API's" and "Telephony-API's". Their philosophy is to integrate these two API's if both are developed completely.

Another analogy which is worthwhile considering is the existence of MS-DOS as a 'de facto' standard. MS-DOS is an Operating System which provides a specified access for users and applications to different kinds of hardware. The ISDN PCI intends to provide a specified access to a network with different services for users and applications. Caused by the speed of developments in this area a 'de facto' standard without any standardized model might be expected.

8.2 Conclusions and Evaluation concerning the TBCE Module

The second part of this report describes the design and realisation of the TIRO_B-Channel_Encryption (TBCE) Module. This module is applicable in a Siemens ISDN Basic Rate hardware environment. It allows encipherment of data in an IOM frame just above layer 1 of the B-channel stack.

The module is designed IOM1 and IOM2 compatible but is only tested in an IOM1 environment because the IOM2 hardware was not available in time.

The test results of the module are positive. In IOM1 environment both the voice and data connection can be enciphered separately or together (with the same key).

For testing the IOM2 compatibility, a more sophisticated testing environment must be set up. Software has to be written which allows sending predefined patterns in the B-channels and check if they are encrypted and decrypted properly.

As the module is tested thoroughly it forms, together with the TIRO-X.25 system, adapted to an ISDN, a hardware basis for an ISDN (TE) PC-board which is capable of providing security services.

References

[1] A.S. Tanenbaum, Computer Networks, Prentice Hall, 1988

[2] AVM Berlin GmbH, COMMON-ISDN-API, Einheitliche Schnittstelle zwischen Applikationsprogrammen und ISDN Adaptern, Spezifikation Version 1.1, Profil A, 9/90

[3] J.M.M. de Zoete et al, Verslag 547 DNL/86 Integrated Services Digital Network, september 1986

[4] CCITT I.230, Bearer Services Supported by an ISDN - Definition of Bearer Service Categories

[5] CCITT I.240, Teleservices Supported by an ISDN - Definition of Teleservices

[6] CCITT I.430, ISDN Basic Rate User-network Interface - Layer 1 Specification

[7] CCITT I.431, ISDN Primary Rate User-network Interface - Layer 1 Specification

[8] CCITT Q.921, ISDN User-network Interface Data Link Layer Specification

[9] CCITT Q.931, ISDN User-network Interface Layer 3 Specification for Basic Call Control

[10] CCITT T.70, Network-independent Basic Transport Service for the Telematic Service

[11] CCITT T.90, Characteristics and Protocols for Terminals for Telematic Services in ISDN

[12] CCITT X.31, Support of Packet Mode Terminal Equipment by an ISDN

[13] CCITT X.509, The Directory - Authentication Framework

[14] ISO 7776, Information Processing Systems - Data Communications - HDLC Description of the X.25 LAPB compatible DTE Single Link Procedure

[15] ISO 8072, Information Processing Systems - Open Systems Interconnection - Transport Service Definition, 1986

[16] ISO 8073, Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification

[17] ISO 8208, Information Processing systems - Data Communications - X.25 Packet Level Protocol for Data Terminal Equipment, 1988

[18] ISO 8348, Information Processing systems - Data Communications - Network Service Definitions, 1984

[19] ISO 8348/AD1, Information Processing systems - Data Communications - Network Service Definitions - Addendum 1: Connectionless Mode Transmission, 1987

[20] ISO 8473, Information Processing systems - Data Communications - Protocol for providing the connectionless-mode network service, 1988

[21] ISO 8802-2.2/DIS, Local Area Networks - Part 2: Logical Link Control, 1987

[22] ISO 9574, Information Technology - Telecommunications and information exchange between systems - Provision of the OSI Connection-Mode Network Service by Packet Mode Terminal Equipment connected to an Integrated Services Digital Network (ISDN)

[23] Standardization of lower layer protocols for ISDN terminals, verslag 395 RNL/90, R.J. Helwerda, juni 1990

[24] Open Data-Link Interface Developer's Guide for NetWare 386 Server Drivers, December 4,1989, Revision 1.1.

[25] Microsoft/3Com LAN Manager Network Driver Interface Specification, August 26,1989, Version 2.0.0

[26] Mix and match network adapters: two upcoming specifications, NDIS and ODLI, will make it easier for you to create a network.
Fisher, Sharon
Byte VOL: v15 ISSUE: n8 PAGINATION: p.277-279
Publication Date: August, 1990

[27] LAN APIs - getting in touch with network resources
Hindin, E.M.
Data Communications VOL: v19 ISSUE: n2 PAGINATION: p.67-71
Publication Date: February, 1990

[28] Kicking and screaming into the present (DECLANworks and PC networks)
van Name, M.L.; Catchings, B.
Byte VOL: v15 ISSUE: n13 PAGINATION: p.125-130
Publication Date: December, 1990

[29] ISDN computer aided telephony
Cvijan, Zarko; Brock, Anthony E.; Corr, Frank P.; Grieg, Joi D.; Kuras, John E.; Schick, Thomas
IEEE Network VOL: v5 ISSUE: n1 PAGINATION: p.46-53
Publication Date: January, 1991

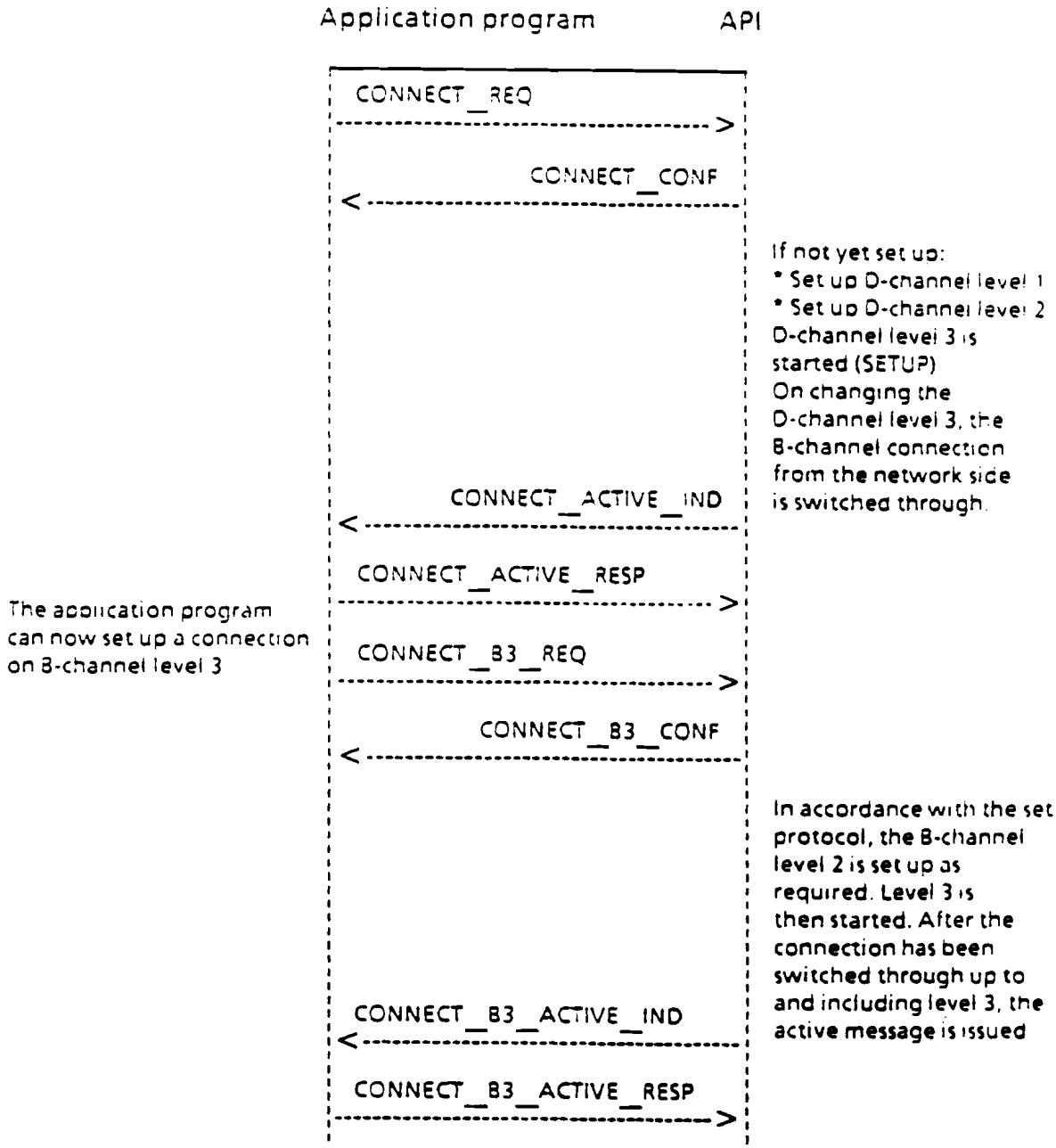
[30] Hayes unveils ISDN API, which could become standard; API may ease software development
Hindin, Eric M.
Data Communications VOL: v19 ISSUE: n2 PAGINATION: p.41-44
Publication Date: February, 1990

[31] ISDN terminals - PC as a convenience telephone and more
Test Report
Funkschau ISSUE: n26 PAGINATION: p.29, 32, 34
Publication Date: December, 1989

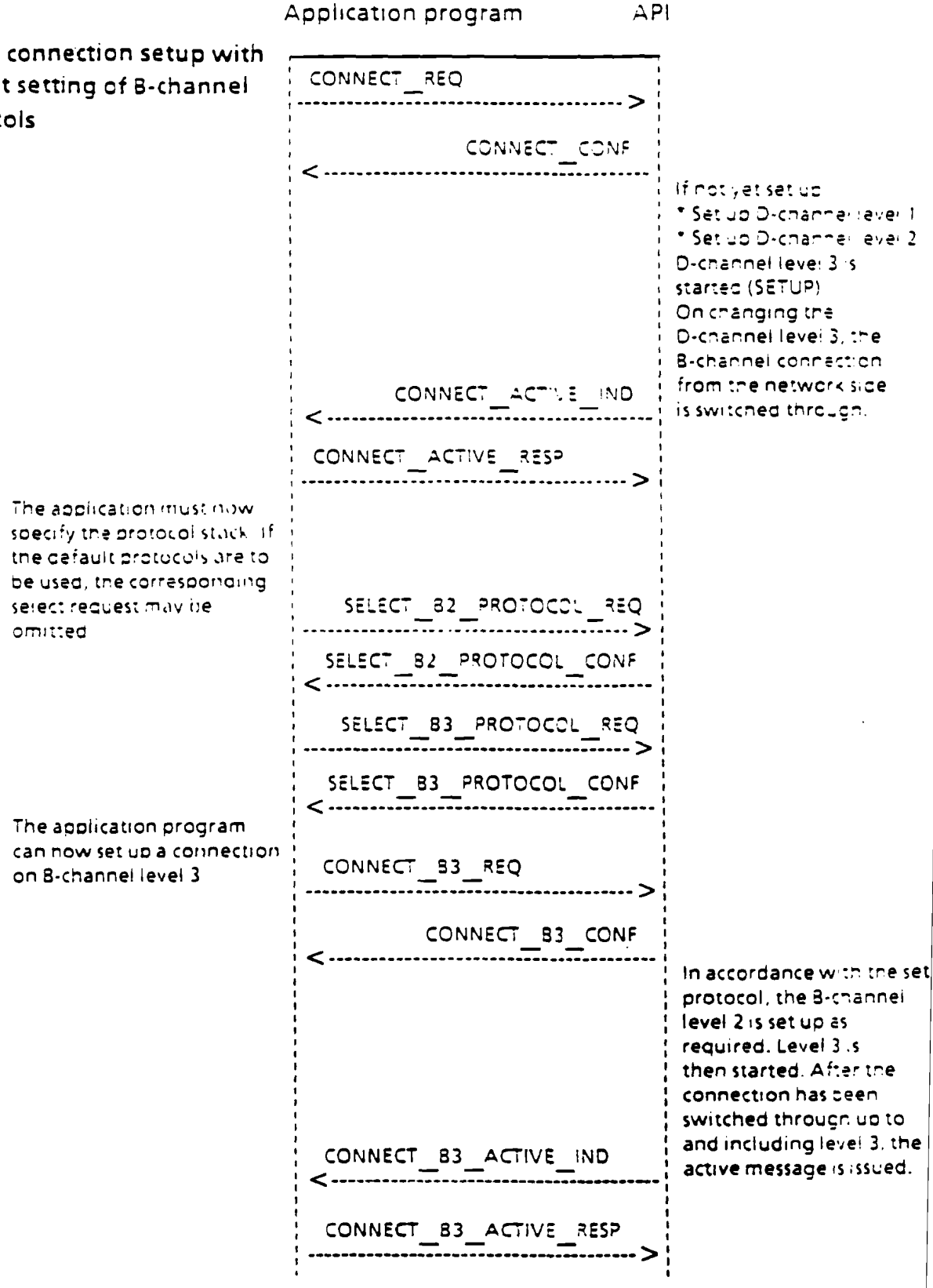
Appendices

A Flow Charts of the ISDN PCI

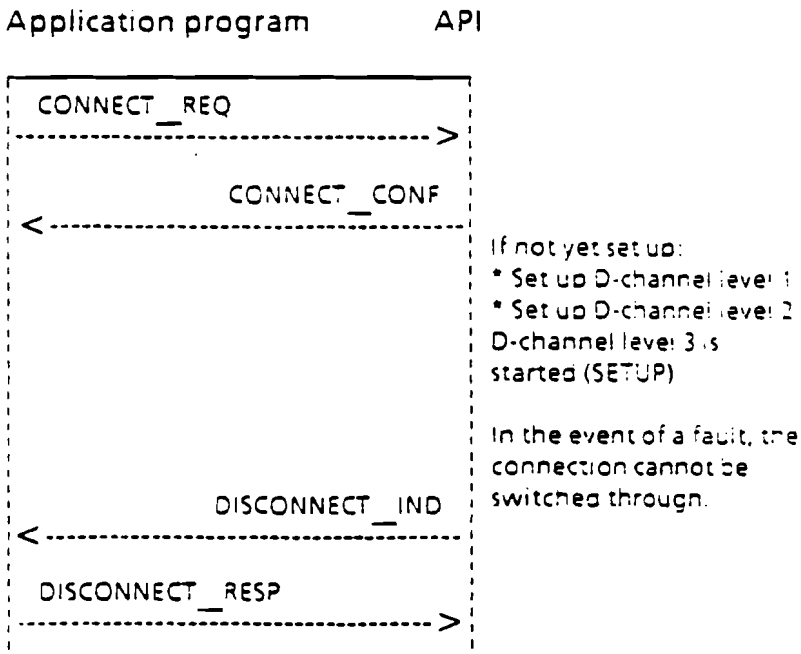
Active connection setup with default setting of B-channel protocols



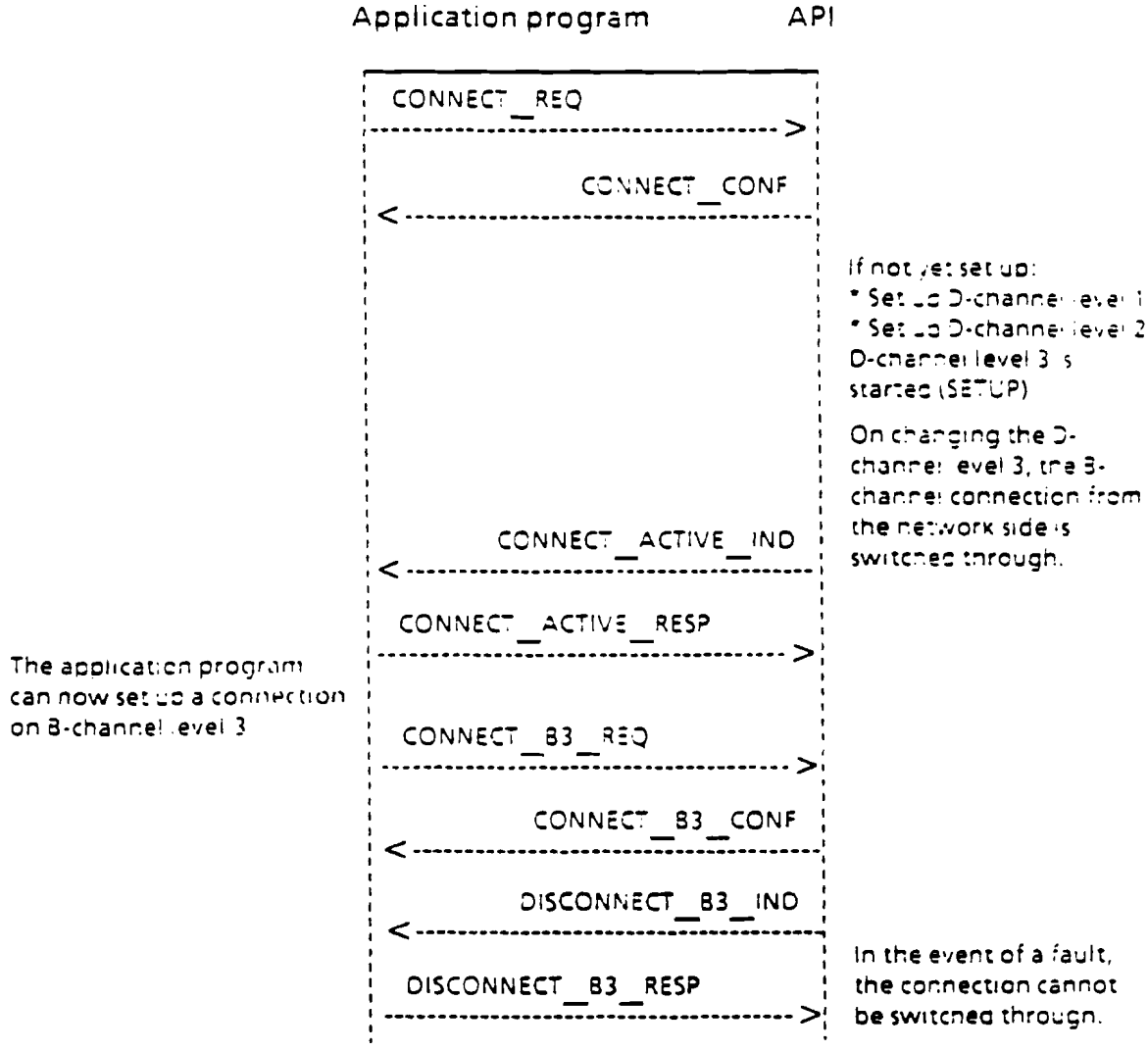
Active connection setup with default setting of B-channel protocols

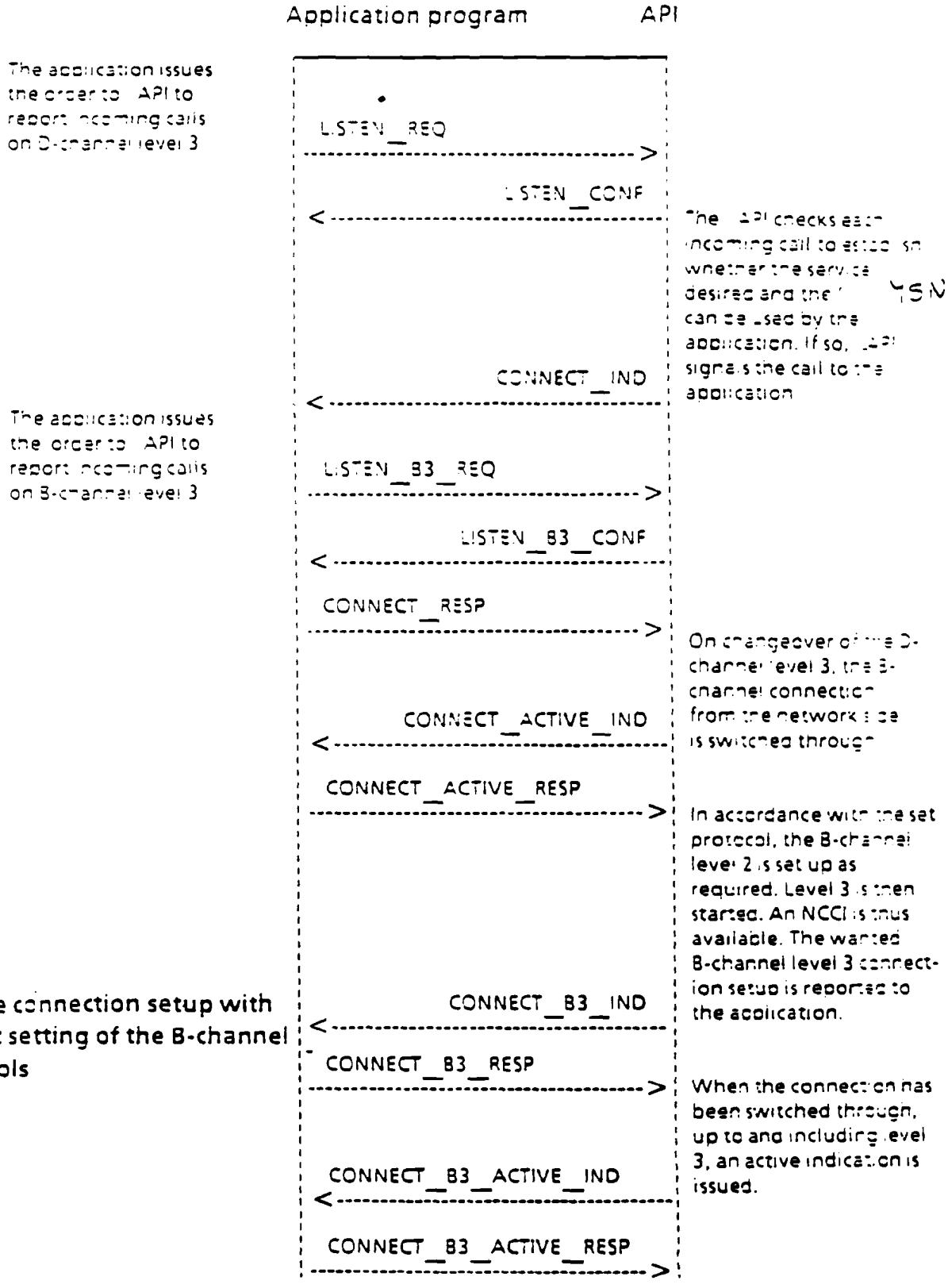


Active connection setup - Error situation CONNECT

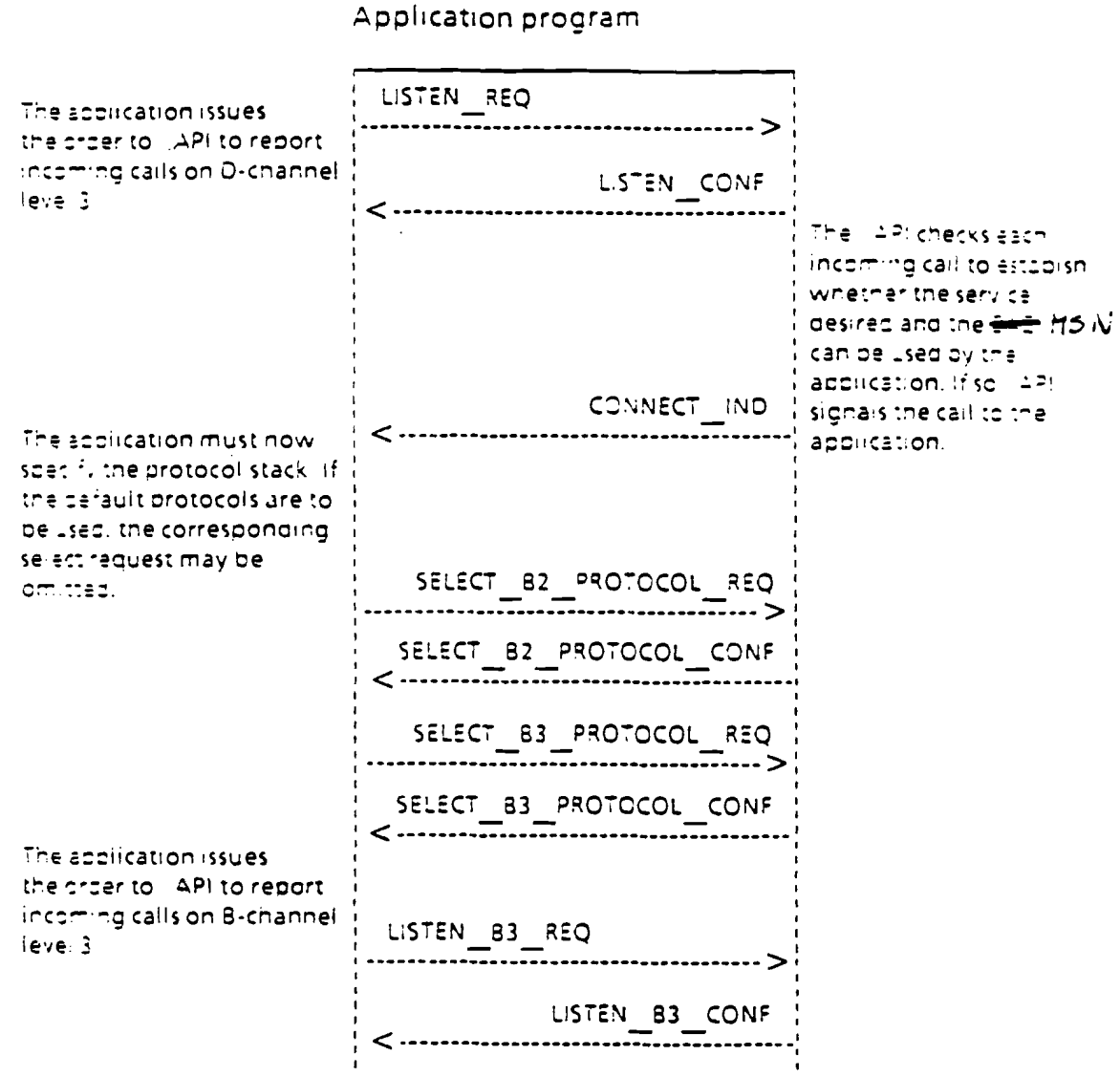


Active connection setup - Error situation CONNECT_B3



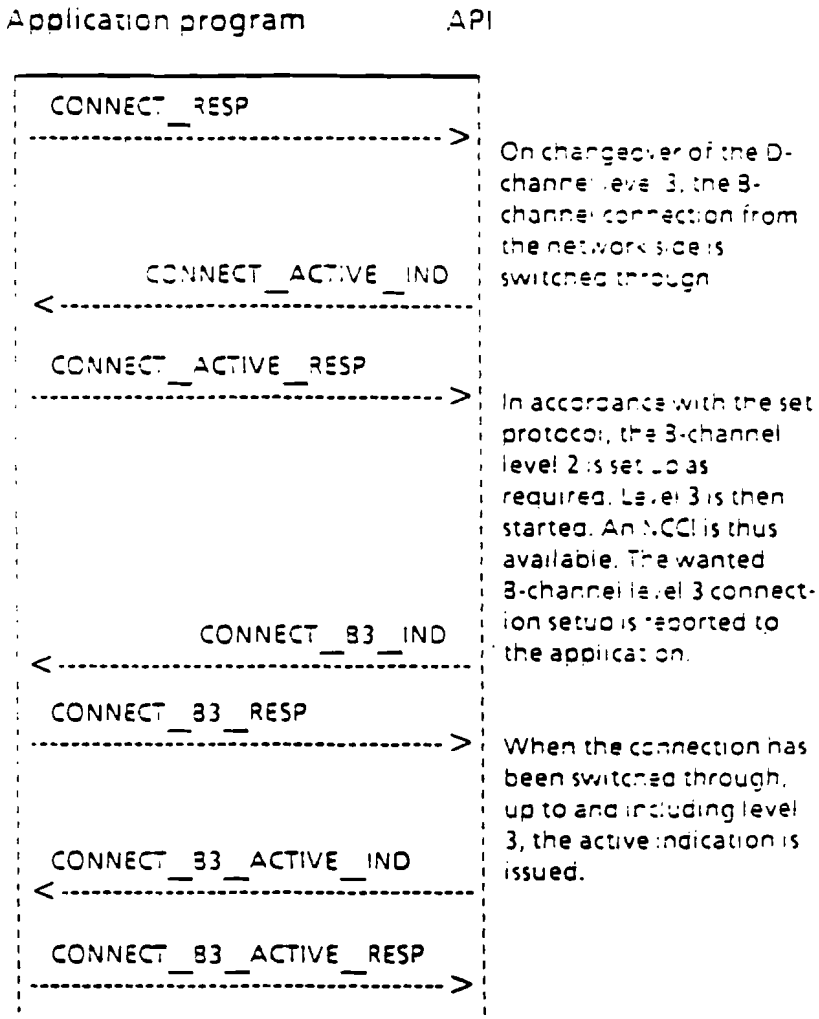

Application program
API
The application issues the order to API to report incoming calls on D-channel level 3
LISTEN_REQ
LISTEN_CONF
The API checks each incoming call to establish whether the service desired and the B-channel can be used by the application. If so, API signals the call to the application
CONNECT_IND
The application issues the order to API to report incoming calls on B-channel level 3
LISTEN_B3_REQ
LISTEN_B3_CONF
CONNECT_RESP
On changeover of the D-channel level 3, the B-channel connection from the network side is switched through
CONNECT_ACTIVE_IND
CONNECT_ACTIVE_RESP
In accordance with the set protocol, the B-channel level 2 is set up as required. Level 3 is then started. An NCCI is thus available. The wanted B-channel level 3 connection setup is reported to the application.
Passive connection setup with default setting of the B-channel protocols
CONNECT_B3_IND
CONNECT_B3_RESP
When the connection has been switched through, up to and including level 3, an active indication is issued.
CONNECT_B3_ACTIVE_IND
CONNECT_B3_ACTIVE_RESP

Passive connection setup with setting of the B-channel protocols



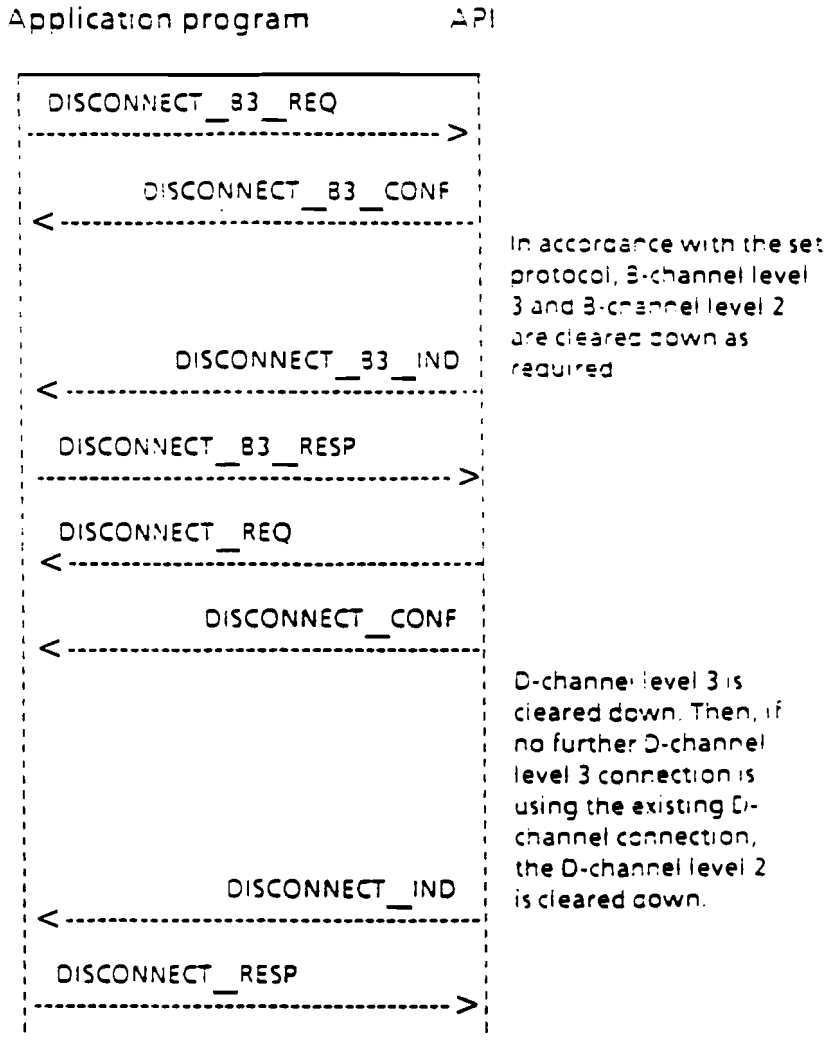
(Continued on next page)

Passive connection setup with setting of the B-channel protocols (continued)



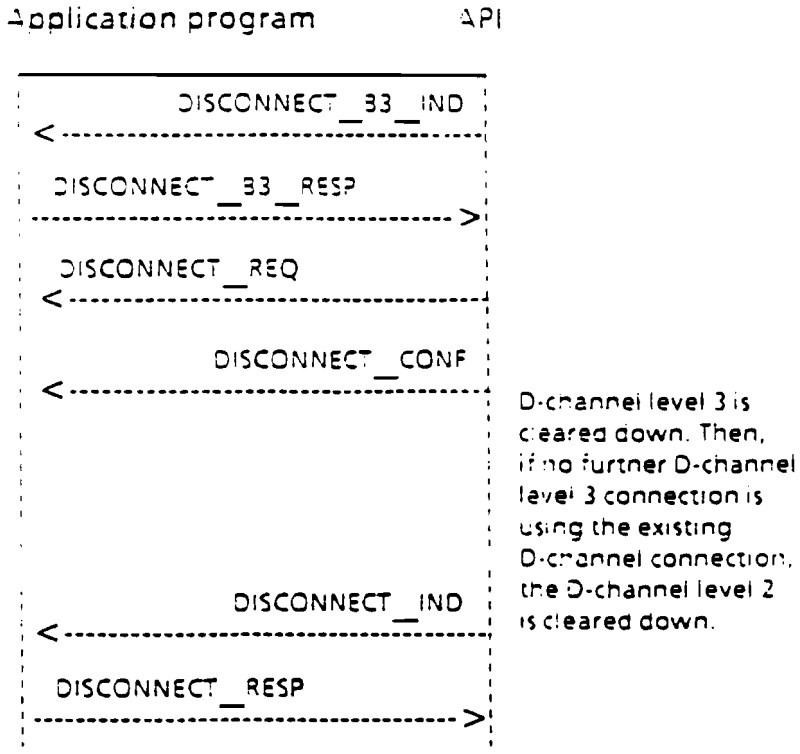
Active connection setup

Precondition: The application program is registered in CAPI with API_REGISTER.
An NCCI is made available by means of a CONNECT_B3_RESP or a CONNECT_B3_IND



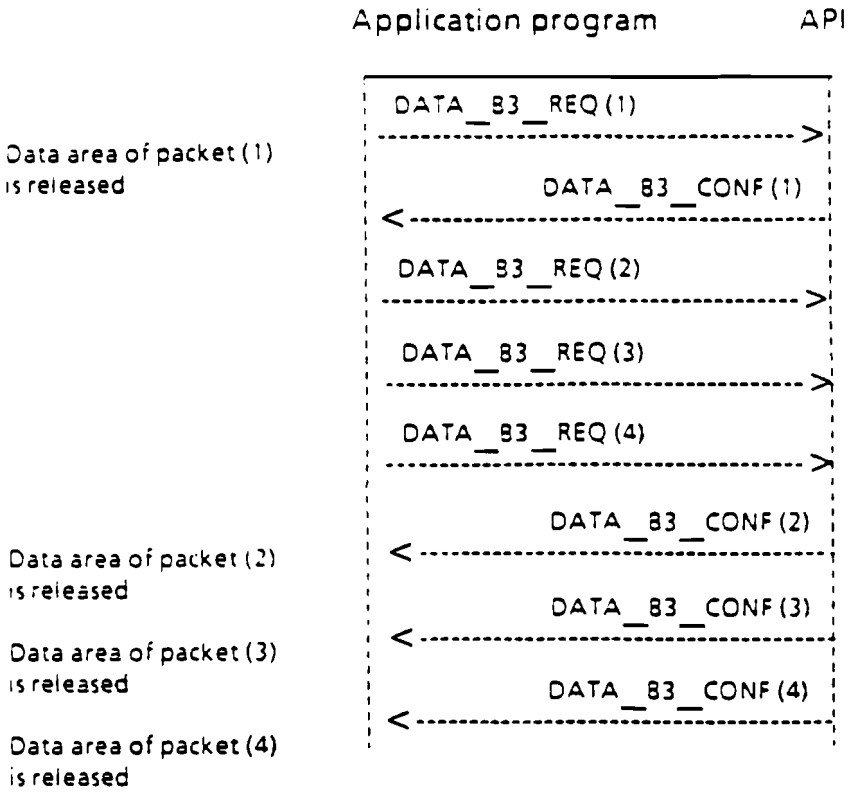
Passive connection setup

Precondition: The application program is registered in CAPI with API_REGISTER.
An NCCI is made available by means of a CONNECT_B3_RESP or a CONNECT_B3_IND.



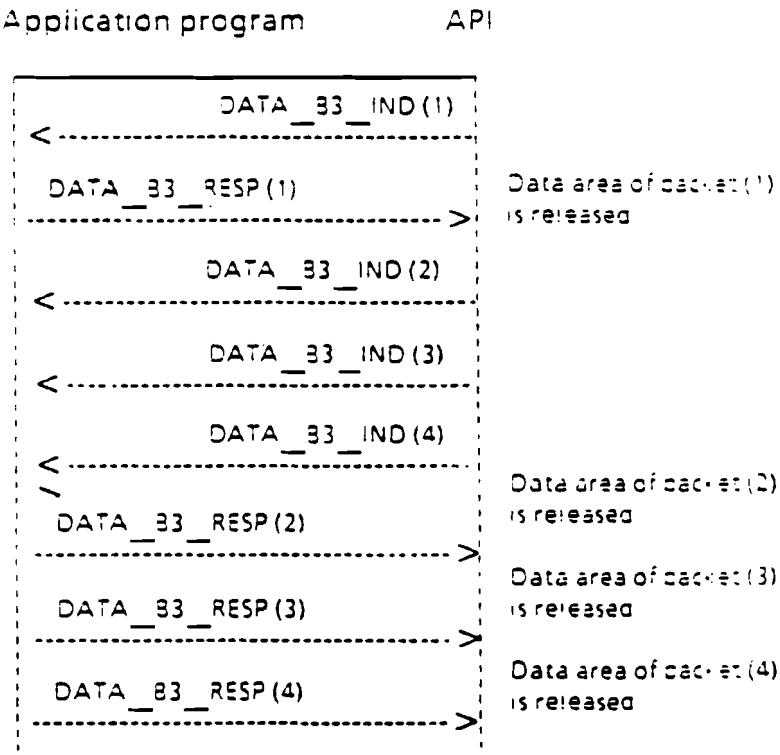
Transmission of data

Precondition: The application program is registered in CAPI with **API_REGISTER**.
An **NCCI** is made available by means of a **CONNECT_B3_RESP** or a **CONNECT_B3_IND**.



Reception of data

Precondition: The application program is registered in API with **API_REGISTER**.
An **NCCI** is made available by means of an **CONNECT_B3_RESP** or a **CONNECT_B3_IND**.



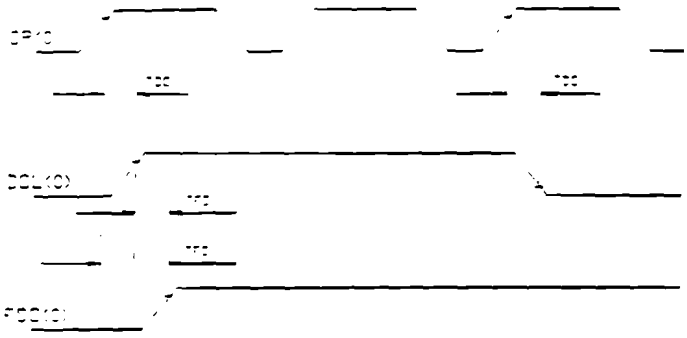
B Timing Characteristics

This appendix contains timing charachteristics of the Siemens ISDN components discussed in section 5.3. These components are:

- S/T-Bus Interface Circuit (SBC PEB 2080)
- ISDN Communications Controller (ICC PEB 2070)
- ISDN Subscriber Access Controller (ISAC-S PEB 2085)

B.1 Timing of the SBC PEB 2080

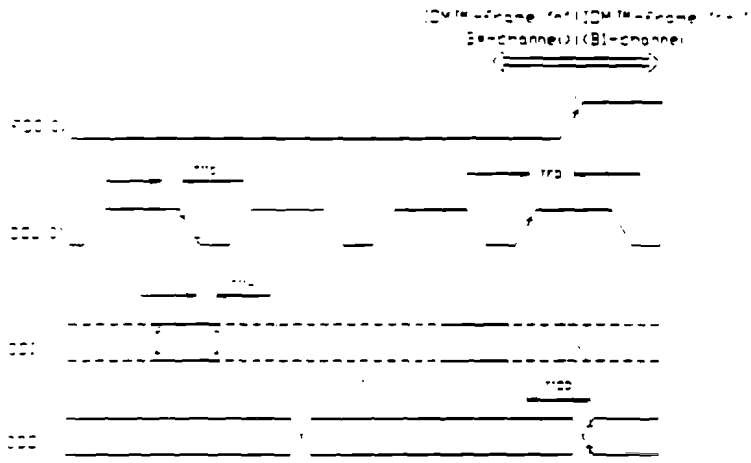
Figure 1 shows the CP,DCL and FSC relationships in IOM1 Master mode.



Symbol	Description	min	max	unit	Conditions
T_{DC}	clock delay CP - DCL	0	50	ns	C_L = 100 pF
T_{FC}	Clock delay CP -FSC	0	50	ns	C_L = 100 pF
T_{FD}	Delay DCL - FSC	-20	20	ns	C_L = 100 pF

Figure 1: CP, DCL and FSC relationships

The S/T-Bus Interface Circuit acts as a master device providing the FSC and DCL signals. Figure 2 shows the timing characteristics of the SBC in TE Master Mode.

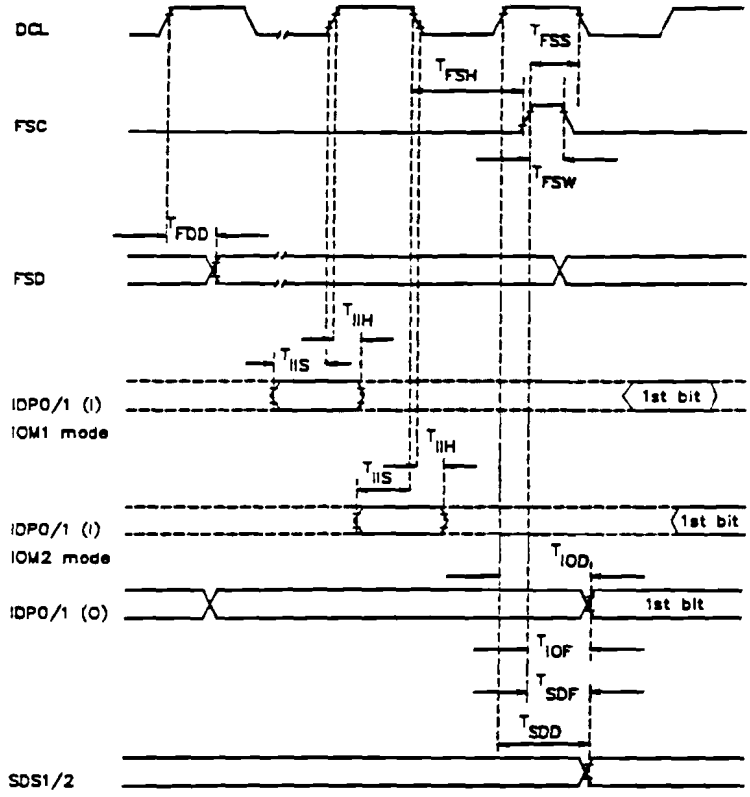


Symbol	Parameter	min		max	unit	Conditions
T_{FD}	Frame sync delay	-20		20	ns	C_L = 100 pF
T_{ICD}	IOM™ output data delay			200	ns	C_L = 100 pF
T_{IIS}	IOM™ input data setup	20			ns	
T_{IIH}	IOM™ input data hold	50			ns	

Figure 2: *SBC timing in Master Mode (TE)*

B.2 Timing of the ICC PEB 2070

The ISDN Communications Controller acts as a slave in the TE configuration. Figure 3 shows the timing of the ICC in IOM Mode.



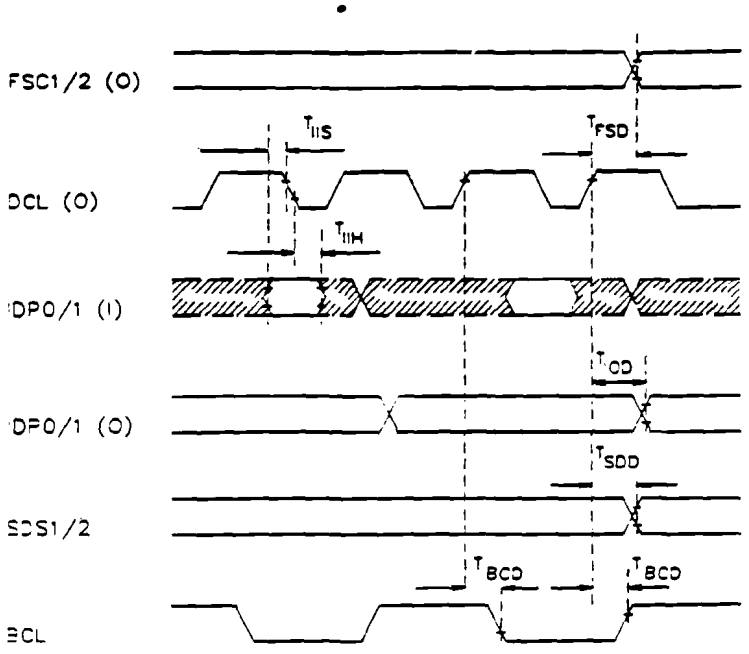
Symbol	Parameter	Limit values min	Limit values max	Unit	Conditions
T_{IOD}	IOM output data delay	20 20	140 100	ns ns	IOM1 IOM2
T_{IIS}	IOM input data setup	40 20		ns ns	IOM1 IOM2
T_{IIH}	IOM input data hold	20		ns	
T_{IOF}	IOM output from FSC		80	ns	See note
T_{SDD}	Strobe signal delay		120	ns	
T_{SDF}	Strobe delay from FSC		120	ns	See note
T_{FSS}	Frame sync setup	50		ns	
T_{FSH}	Frame sync hold	30		ns	
T_{FSW}	Frame sync width	40		ns	
T_{FDD}	FSD delay	20	140	ns	

Figure 3: *IOM Timing of the ICC*

B.3 Timing of the ISAC-S PEB 2085

The ISDN Subscriber Access Controller offers the functionality of the SBC and the

ICC on one chip. Due to this fact resembles the IOM interface of the ISAC-S very much the IOM interface of the ICC. Figure 4 shows the Timing of the ISAC-S.



Symbol	Parameter	Limit values min	max	Unit	Conditions
T_{IOD}	IOM output data delay	20 20	140 100	ns ns	IOM1 IOM2
T_{IIS}	IOM input data setup	40+TWH 20		ns ns	IOM1 IOM2
T_{IIH}	IOM input data hold	20		ns	
T_{FSD}	FSC1/2 strobe delay	-20	20	ns	
T_{SDD}	Strobe signal delay		120	ns	
T_{BCD}	Bit clock delay	-20	20	ns	
T_{FSS}	Frame sync setup	50		ns	
T_{FSH}	Frame sync hold	30		ns	
T_{FSW}	Frame sync width	40		ns	
T_{FDD}	FSD delay	20	140	ns	

Figure 4: *IOM Timing of the ISAC-S*

C Timing Calculations

This appendix describes the relation between the timing of the Siemens ISDN components and the TIRO-ASIC

Calculation of the Available Time

This part describes the calculation of the time that is available between the sending and receiving of one databit on the datalines in an IOM interface in order to find out if enciphering without delay can be implemented. this time is referred to as 'Available Time'.

Goal is to create a module which is IOM1 and IOM2 compatible. In order to achieve this compatibility the timing characteristics of both modes have to be taken into account.

Appendix B2, figure B.3, shows a major difference between the IOM1 and IOM2 timing characteristics. In IOM1 mode data is clocked in at an 'odd-numbered' rising edge of DCL, where this in IOM2 mode happens on the second falling edge after an 'even-numbered' rising edge of DCL.

This means that the total time available to manipulate the data lasts 1 IOM1-DCL period or $1^1/_2$ IOM2 DCL periods. The following formula tries to formulate the constraints ;

$$AT = MIN_{[DCL]}(IOM1,IOM2)-MAX_{[OD]}(SBC,ICC,ISAC\text{-}S)-MAX_{[IS]}(SBC,ICC,ISAC\text{-}S)$$

Where;

AT = Available Time OD = Output Delay IS = Input Setup

When substituting the values of appendix B it can be found that;

$MIN_{[DCL]}$ (IOM1,IOM2)	=	MIN($1*1950, 1^1/_2 * 650$)	= 975 ns
$MAX_{[OD]}$ (SBC,ICC,ISAC-S)	=	MAX(200,140,140)	= 200 ns
$MAX_{[IS]}$ (SBC,ICC,ISAC-S)	=	MAX(20,40,40+TWH)	= 40 ns
		AT	= 735 ns

The * refers to the '40+TWH'. This is only applicable in IOM1 mode. In that case the DCL period is 1950 ns. This mens that TWH is globally 975 ns. Hence the available time in IOM1 mode becomes also 975 ns. This is the time taken into account already, so the taken maximum of 40 ns won't violate the essence of the calculation.

Timing of the TIRO-ASIC

The TIRO-ASIC is developed for simultaneous encryption and decryption in several modes. The mode to be used in the TBCE Module is the Synchronous mode. Figure C.1 shows the timing diagram of the Synchronous Interface.

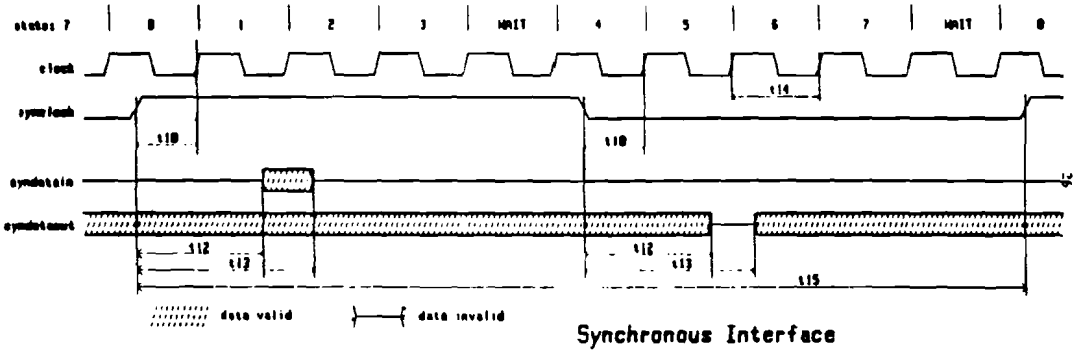


Figure C.1: *Timing of the TIRO-ASIC*

Table C.1 shows the values of the parameters mentioned in figure C.1.
The Table shows that the maximum synclock frequency is 2048 kHz. This gives a period of 488 ns. This means that it has to be possible to encipher one bit without delay with an available time of 735 ns. If figure C.1 is examined more closely it can be found that the outputbit is available at (t10+4*t14+t13) = 7*t14 = 437.5 ns after the rising edge of synclock. This is the case when the wait state is not implemented.

Conclusion

Results of the calculations give that the available time to encrypt or decrypt one bit is about 735 ns. The minimum time the TIRO-ASIC needs can be brought back to 437.5 ns, if it operates in the synchronous mode at the maximum clockfrequency. Therefore it can be concluded that, if the switch and multiplex hardware does not need more than hundreds of nanoseconds, it is possible to encipher the data without

REFERENCE	MIN [ns]	MAX [ns]	DESCRIPTION
t10	10	(t14-10)	synclock setup time to guarantee internal recognizing of the synclock signal on the next edge of the system clock.
t12		(t14-10)	data setup in relation to the active edge of the synclock signal
t13	(2*t14+10)		data hold in relation to the active edge of the synclock signal
t14	0	16.5 MHz	clock frequency symmetrical 50 %
t15		2048 kHz	synclock frequency (+/- 50 ppm) symmetrical 50 %

Table C.1: *Parameters of TIRO-ASIC synchronous mode timing*

introducing delay and without violating the IOM specification.

D Slot Selector

This appendix contains figures of the slot counter and the IOM frames. Figure D.1 shows the scheme of the slot counter.

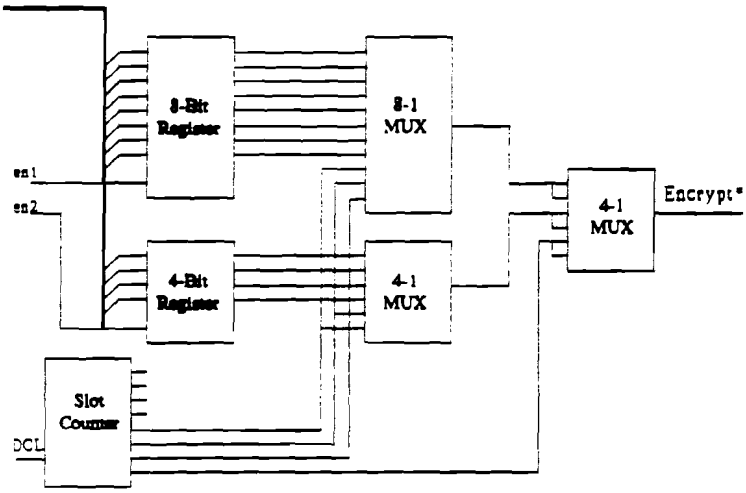


Figure .1: *Sceme of the Slot Counter*

Figure D.2 shows the IOM frames and their relation to the registers in the slot counter.

IOM2

B1	B2	MON	D	C/I	IC1	IC2	MON	C/I			C/I
D8_0	D8_1	D8_2	D8_3	D8_4	D8_5	D8_6	D8_7	D4_0	D4_1	D4_2	D4_3

B1	B2	MON	D	C/I
D8_0	D8_1	D8_2	D8_3	

DCL	Encrypt
0000xxxx	D8_0
0001xxxx	D8_1
0010xxxx	D8_2
0011xxxx	D8_3

DCL	Encrypt
0000xxxx	D8_0
0001xxxx	D8_1
0010xxxx	D8_2
0011xxxx	D8_3
0100xxxx	D8_4
0101xxxx	D8_5
0110xxxx	D8_6
0111xxxx	D8_7
1000xxxx	D4_0
1001xxxx	D4_1
1010xxxx	D4_2
1011xxxx	D4_3

Figure .2: *Relation IOM Frames , Registers and Encrypt signal*

E Reset Puls Generator

This appendix describes the state machine number 1 of the design in functional blocks, the "Reset Puls Generator".

Function

The function of the Reset Puls Generator is to generate a reset puls with a width of 1 clockperiod (62,5 ns) to reset the slot counter and synchronize the Bit Clock Generator. The puls has to be generated when the corresponding rising edges of DCL and FSC at the start of a new frame have occurred. FSC and DCL are input signals, the RPuls is an output signal.

Timing

The RPuls has to become active when the corresponding rising edges of DCL and FSC have occurred. This is the first time when both the DCL level and the FSC level are detected high. The timing specifications of the IOM2 interface allow a 'jitter' between both mentioned rising edges. The rising edge of FSC can occur almost immediately after the previous falling edge of DCL. It also can occur after the rising edge of DCL. Figure E.1 shows the timing of the Reset Puls Generator.

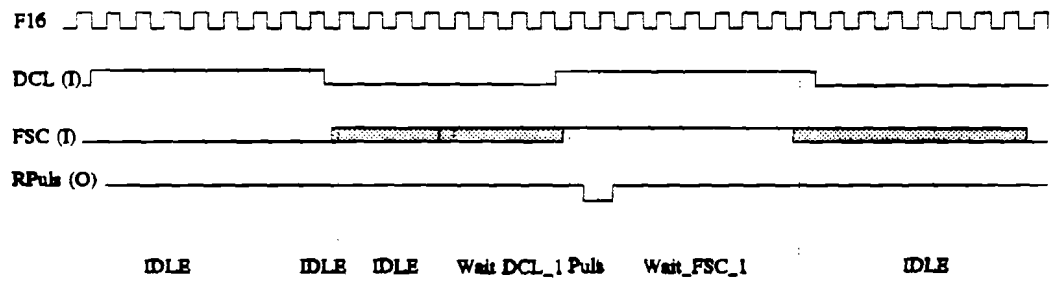
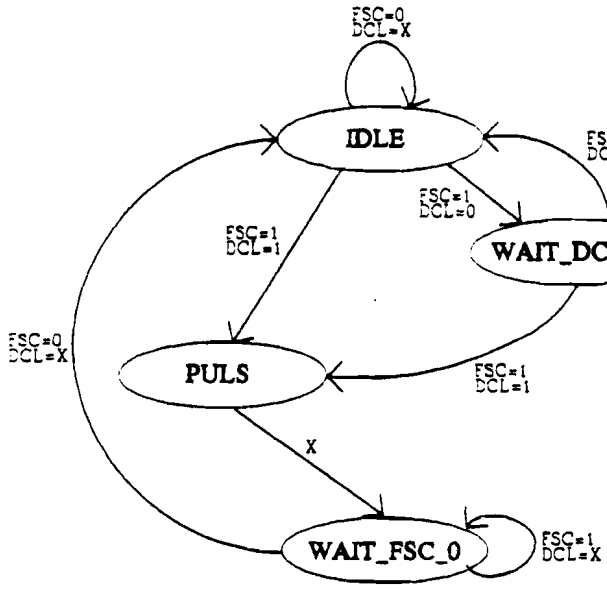


Figure 1: *Timing of the Reset Puls Generator*

State Diagram and State Encoding

The state machine has to know if the corresponding edge of DCL has already occurred at the time the rising edge of FSC occurs. The state machine detects the rising edge of FSC. If no rising edge of DCL has occurred, it waits in a state until that edge occurs. If it had already occurred, a reset puls is generated immediately. Figure E.2

shows the state machine and the state encoding.



	CNT1	CNT0	RPuls
IDLE	1	1	1
WAIT_DCL_1	1	0	1
PULS	1	1	0
WAIT_FSC_0	0	1	1

Figure 2: State Diagram and State Encoding of the Reset Puls Generator

F Bit Clock Generator

This appendix describes the state machine number 2 of the design in functional blocks, the "Bit Clock Generator".

Function

The function of the Bit Clock Generator is to generate a clocksignal derived from DCL which has half the frequency of DCL. The reset puls is also an input signal and is used to synchronize the rising edge of the BCL (Bit CLock) to an 'even numbered' rising edge of DCL according to the IOM2 Specification. The data on the datalines changes after the even numbered rising edges of DCL, and also on a rising edge of BCL. DCL and the RPuls from the Reset Puls Generator are input signals, BCL is an output signal.

Timing

As already mentioned, BCL has half the frequency of DCL. Furthermore a synchronization has to take place in order to get a rising edge of BCL synchronized to an even numbered rising edge of DCL.

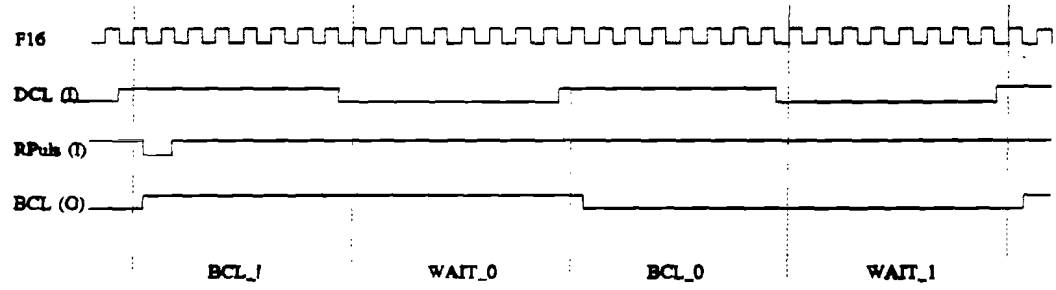


Figure F.1: *Timing of the Bit Clock Generator*

This is done with the RPuls from the Reset Puls Generator. The RPuls becomes active when the corresponding rising edges of DCL and FSC have occurred. This corresponding rising edge of DCL is the first of a frame (number 0) and indicates a change on the datalines. So this is an even numbered rising edge. Figure F.1 shows the timing of the Bit Clock Generator.

The RPuls signal is used to synchronize both clocks at the start of the system (e.g. Power Up) and, if synchronization is lost, at the start of each new frame. Figure F.2

shows the way which the synchronization takes place.

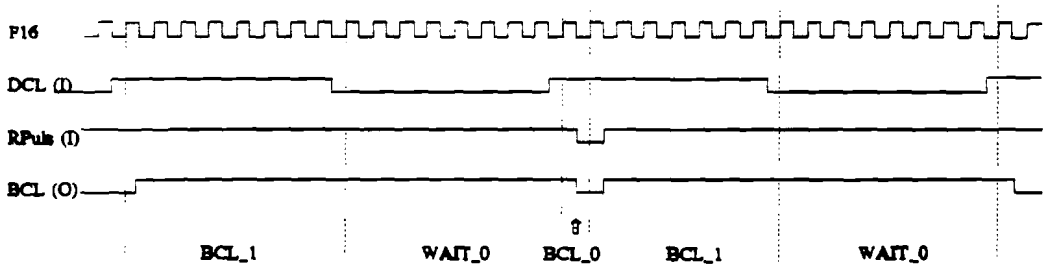


Figure F.2: *Synchronization of BCL and DCL with the Reset Puls*

State Diagram and State Encoding

The state diagram consists of two parts. The first part is only used at system start up. Once this has passed, the state machine remains in the second part. A period of four states corresponding to the two cycles of DCL. The Bit Clock output changes only once per DCL cycle and thus becomes half the frequency. In every state the level of the reset puls signal is checked, in order to recover synchronization at the start of a frame in case this was lost. Figure F.3 shows the state diagram and the state encoding of the Bit Clock Generator.

	CNT1	CNT0	BCL
AFTER_RESET	1	1	1
IDLE	1	1	0
BCL_1	1	0	1
WAIT_0	0	0	1
BCL_0	0	0	0
WAIT_1	1	0	0

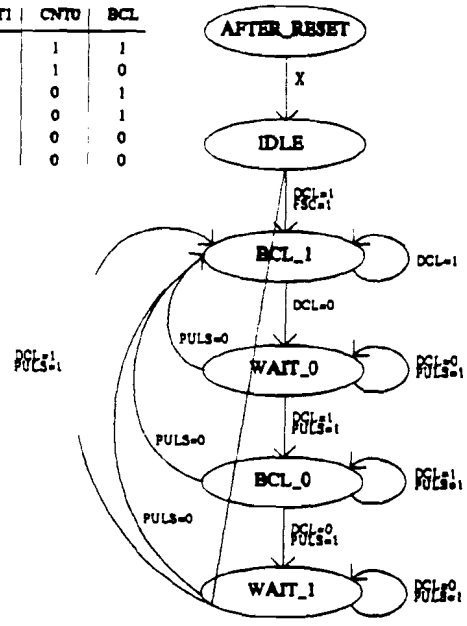


Figure F.3: *State Diagram and State Encoding*

G TIRO-ASIC Control Logic

This appendix describes the state machine number 3 of the design in functional blocks, the "TIRO-ASIC Control Logic".

Function

The function of the TIRO-ASIC Control Logic is control the two ASIC's by providing them of a clock signal at the right time. This 'right time' is determined by the slot selector by activating the Encrypt* signal. The TIRO-ASIC Control Logic checks this Encrypt Signal at every rising edge of BCL. It then also checks the Asic_nr signal to determine which ASIC is active at that time. In this way the TIRO-ASIC Control Logic drives the encrypt and decrypt units of the active ASIC at the time a slot passes which is programmed to be enciphered

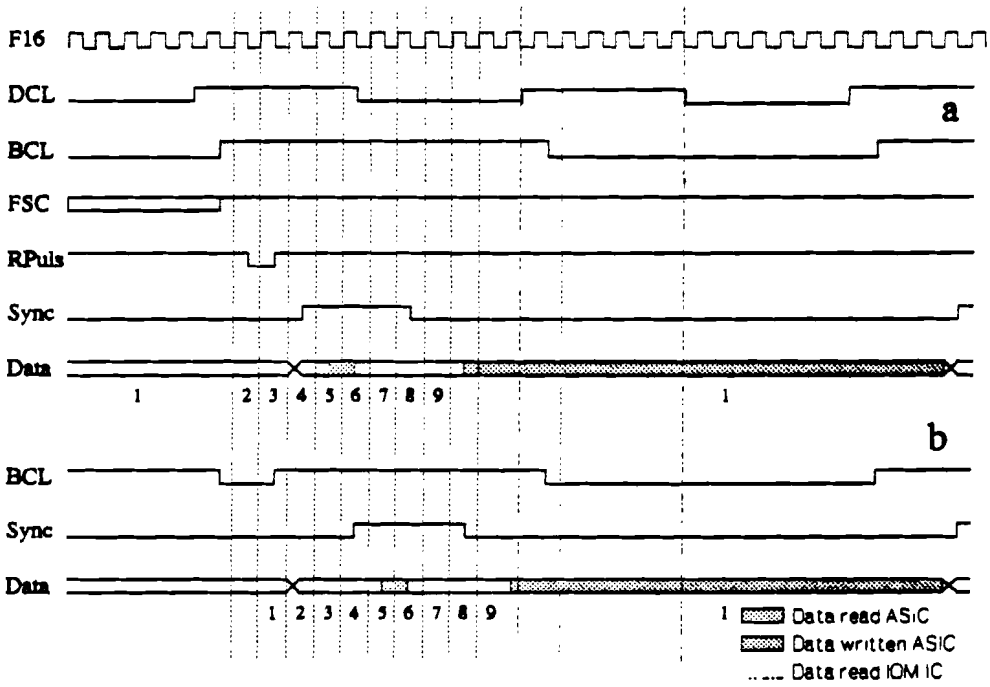


Figure 1: *Timing diagram with and without BCL/DCL synchronization*

Timing

BCL, Encrypt* and Asic_nr are input signals of the TIRO-ASIC Control Logic. The outputsignals are called Sync1 and Sync2 which represent the clock signal for ASIC1

and ASIC2. Figure G.1 shows the timing of the TIRO-ASIC Control Logic. Figure G.1a shows the scenario if BCL and DCL are synchronized. In order to clarify the relation between the signal in the total system, the figure contains all the signals of the system. Important to notice is that the TIRO-ASIC Control Logic provides only the timing relation between a rising edge of BCL and the rising edge of a Sync signal. Another important fact is that data is read from the datalines on a rising edge of DCL in IOM1 and on a falling edge of DCL in IOM2 interfaces.

Figure G.1b shows the timing when synchronization of DCL and BCL is lost. The only influence this has on the system is that the encryption or decryption is delayed two states. Each state stands for 62.5 ns. This delay does not cause any trouble in the correct functioning of the system. When this situation occurs, the ASIC still has written the enciphered bit on the datalines in time before the IOM-devices use the data. Table G.1 shows the relation between the numbers in figure G.1 and the states in figure G.2.

	Number	CNT2	CNT1	CNT0	SYNC1_	SYNC2_
AFTER_RESET	0	1	1	1	1	1
IDLE	1	1	1	1	0	0
WAIT_0	2	1	1	0	0	0
WAIT_1	3	1	0	1	0	0
ACTIVATE_ASIC	4	1	0	0	0	0
NOT_ACTIVATE	5	0	1	1	0	0
STATE_1_0	6	1	1	1	1	0
STATE_1_1	7	1	1	0	1	0
STATE_1_2	8	1	0	1	1	0
STATE_1_3	9	1	0	0	1	0
STATE_1_4	10	0	0	1	0	0
STATE_2_0	11	1	1	1	0	1
STATE_2_1	12	1	1	0	0	1
STATE_2_2	13	1	0	1	0	1
STATE_2_3	14	1	0	0	0	1
STATE_2_4	15	0	1	0	0	0

Table G.1: *State Encoding and Numbering*

State Diagram and State Encoding

Figure G.2 shows the state diagram of the TIRO-ASIC Control Logic. The two wait states are implemented to wait for the data on the datalines and the Encrypt* signal to get valid. After those wait states the signals Encrypt* and Asic_Number are checked and according to their values the state machine either activates one of the ASIC's or waits until the BCL gets low again. This cycle is executed for every bit that passes on the datalines. The Slot Counter takes care of the fact that the Encrypt* signal stays

active during the programmed slots only, so the TIRO-ASIC Control Logic generates 'bursts' of eight clockpulses to encipher slots.

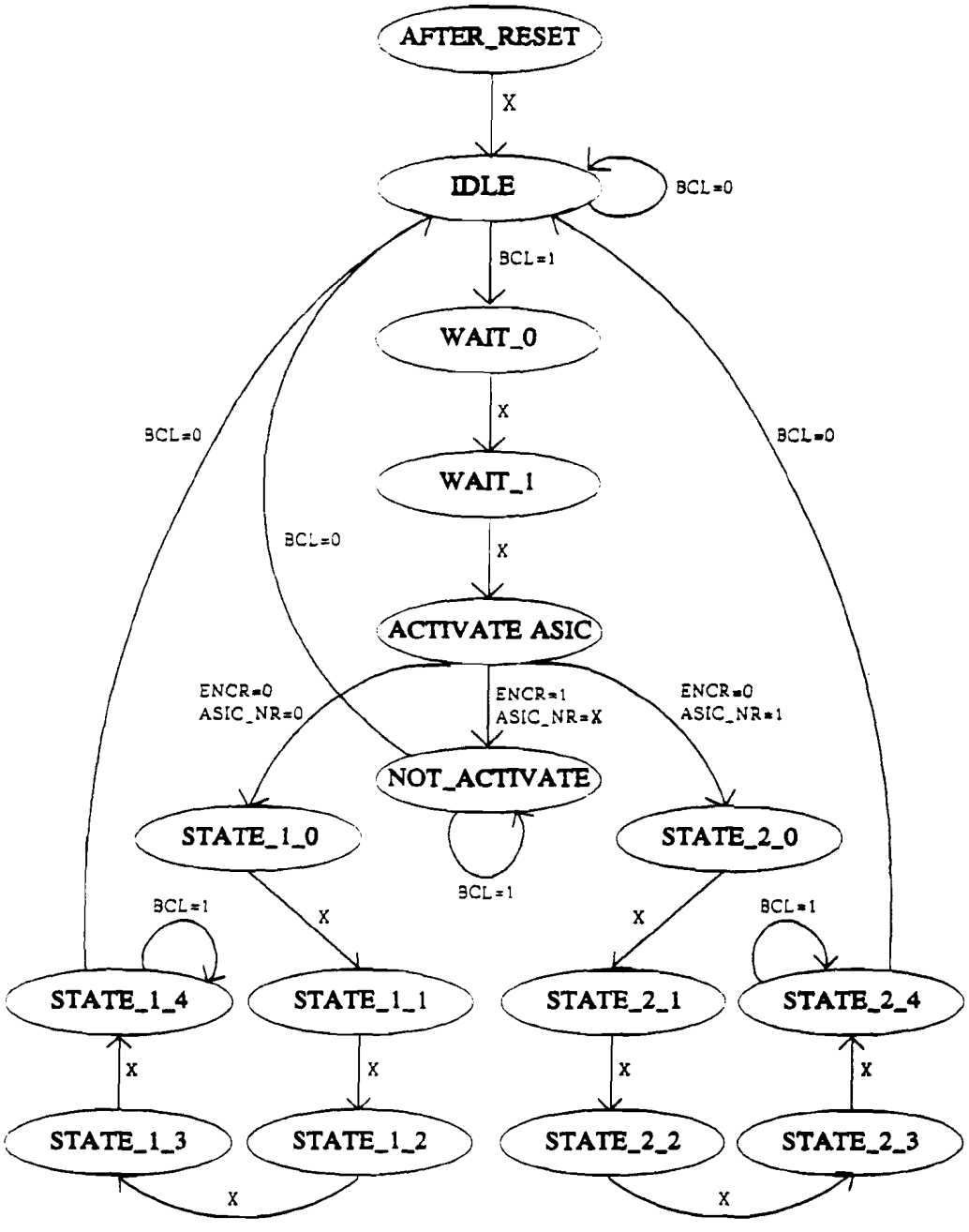


Figure G.2: *State Diagram of the TIRO-ASIC Control Logic*

H XILINX Layout

This appendix contains the figures of the actual layout that has been programmed into the XILINX FPGA. Figure H.1 shows the adres decoder, figure H.2 shows the databus, figure H.3 shows the 'switching logic' and finally figure H.4 shows a complete overview of the system.

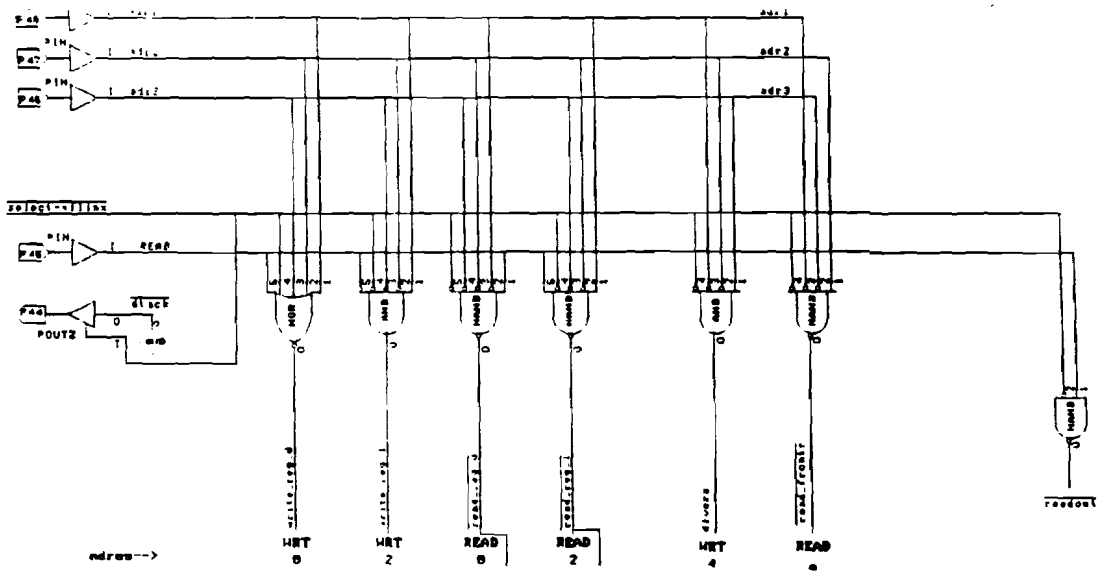


Figure H.1: *Adres Decoder*

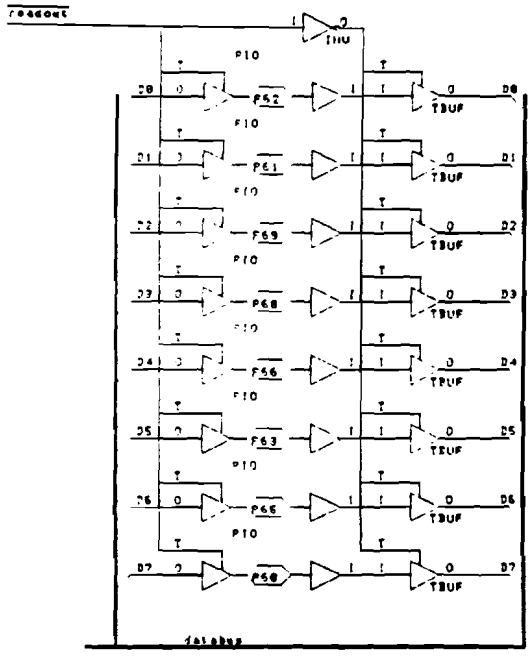


Figure H.2: *DataBus*

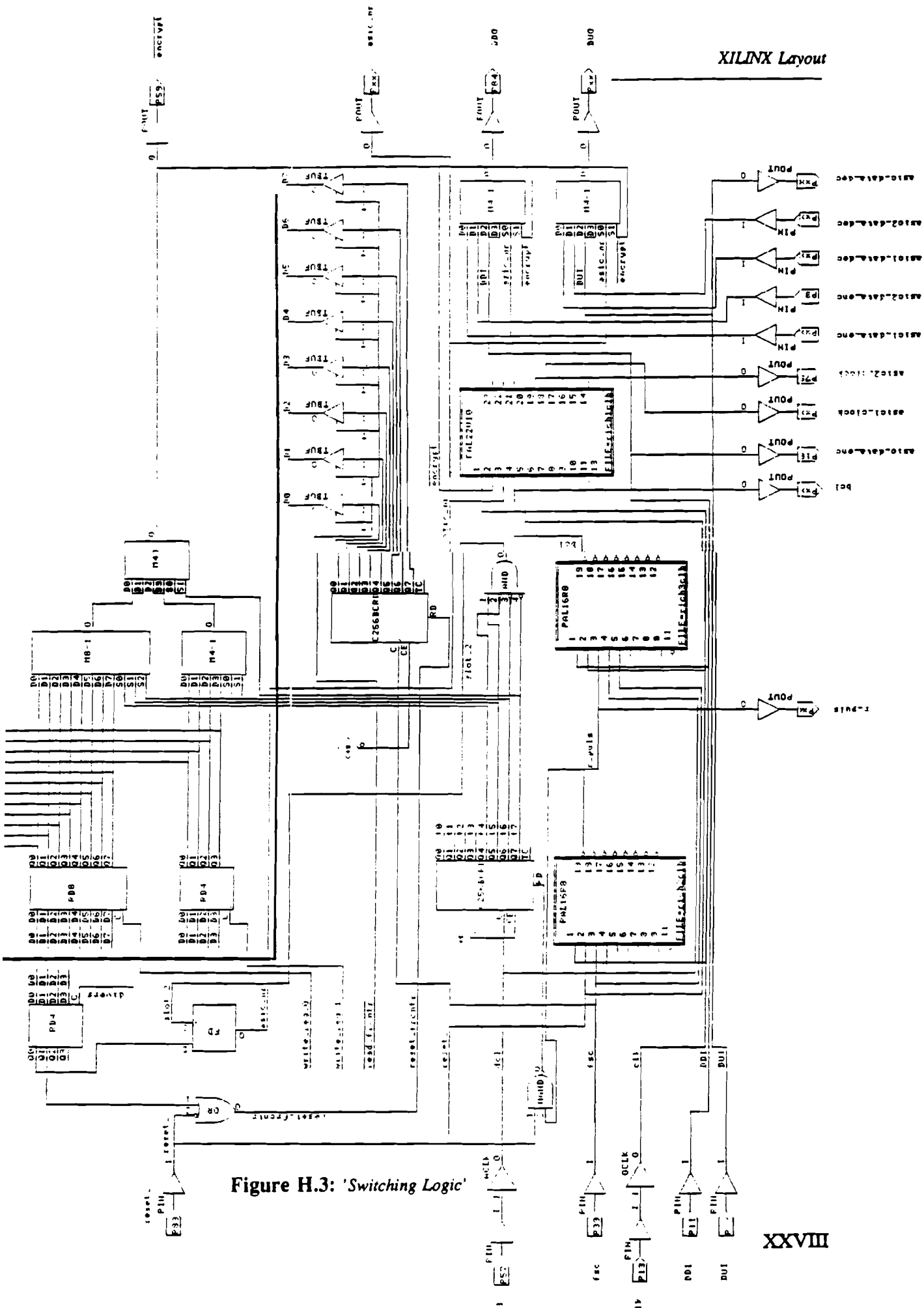


Figure H.3: *'Switching Logic'*

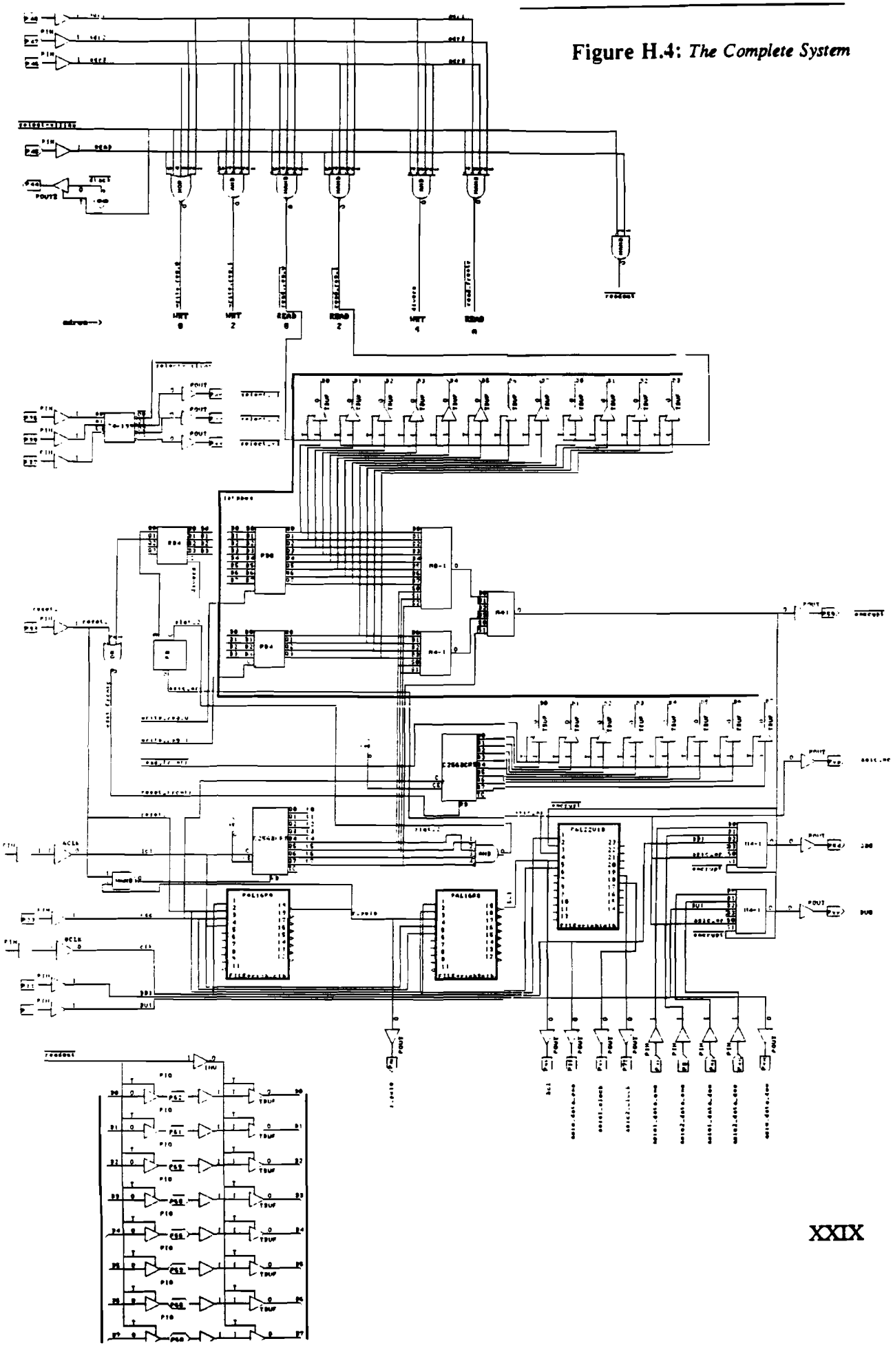


Figure H.4: *The Complete System*