FACULTEIT ELEKTROTECHNIEK

VAKGROEP DIGITALE SYSTEMEN

~~Stage~~ / Afstudeeropdracht

Onderwerp:	**Ontwikkeling van een private network terminal.**

Naam:	Willemsen, E.F.M. (Edgar/ITE-student)
Adres:	Kapelstraat 11A
Postcode/Woonplaats:	1404 HT Bussum 035 - 894005 (koc 54)
Tel.nr. prive/zaak:	02159 - 15401 / ~~02159 - 894001~~
Identiteitsnr.:	226398
Geb. datum:	6 januari 1967
Coach:	ir. M.J.M. van Weert
	des. ~~ir~~. D. van 't Zand (Philips APT)

Datum aanvang:	1 janauri 1990
Datum einde:	4 oktober '90
Verslag ingeleverd d.d.:	
Verslag beoordeeld d.d.:	11 oct '90
Paraaf coach:	[illegible]
Cijfer coach:	9
Cijfer Hoogleraar:	9 MSr
Bonnummer:	006590
Bondatum:	90/10/17
Aantal bladen verslag:	

VERGEET U NIET EEN KORTE INHOUD + TREFWOORDENLIJST BIJ TE VOEGEN ?
VERSLAG LOSBLADIG INLEVEREN OP A4 EN BLADZIJDEN DOORNUMMEREN.

VERSLAG INLEVEREN BIJ DE VAKGROEPSECRETARESSE;

EB 275

SPECIFICATION AND IMPLEMENTATION OF A PRIVATE NETWORK TERMINATION

MASTER'S THESIS
by
E.F.M. WILLEMSEN

Supervisor	: Prof. Ir. M.P.J. Stevens	(Technical University Eindhoven)
Advisors	: Ir. M.J.M. van Weert	(Technical University Eindhoven)
	Drs. D. van 't Zand	(PTDSN B.V. Hilversum)

September 1990

ABSTRACT

Is it possible to connect ISDN terminals to the existing digital line interface of SOPHO-S via a Private Network Termination?
This is one of the problems, that have to be solved, before SOPHO-S (i.e. the Philips digital PABX) can be regarded as the link between Office Automation and ISDN. Other matters, that have to be investigated before SOPHO-S can be considered as an IS-PBX include the following:

- Development of a digital line circuit, in order to be able to connect ISDN terminals directly to SOPHO-S;

- Definition of a primary rate ISDN trunk, for connection of SOPHO-S to the public ISDN network.

This report deals with the specification and implementation of a Private Network Termination (PNT), upon which ISDN terminals can be connected. The starting-point is that the development of the PNT must not impose modifications on either SOPHO-S or ISDN concepts!

The development of the PNT is focused on the specification of its layer 3 function. This PNT layer 3, that consists of a protocol interworking function, is specified using the CCITT's specification and description language SDL.
Chapter 5 treats the translation of the PNT layer 3 specification into a software implementation.
Finally, chapter 6 presents the conclusions and recommendations, including the most important one: the PNT layer 3 specification, as presented in this report, is indeed capable of supporting communication between SOPHO-S and ISDN terminals.

ACKNOWLEDGEMENTS

I would like to thank the following persons for their support during my work at Philips Telecommunication Systems in Hilversum.
Prof. Ir. M.P.J. Stevens, for allowing me to graduate under his responsibility. Ir. M.J.M. van Weert, for his support on behalf of the University of Technology in Eindhoven.
All the employees of BCS DCT DM for the nice conversations and the pleasant working environment they delivered.
Finally, special thanks to Drs. D. van 't Zand, my coach at Philips, for his support, which helped me to complete the project in a pleasant way.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Telecommunications in its advanced appearance has grown into ISDN, the Integrated Services Digital Network. ISDN is under construction in many countries of the developed world and will be the universal switching and transport medium for all kinds of information: voice, data, text and images.

In Office Automation it is the task of the Private Automatic Branch Exchange (PABX) to offer the business organization a set of adequate communication functions for in-house use. Historically it serves all voice communication, but with modern digital technology it can do far more than that: data, text and image communication. The PABX in particular can offer a general in-house communication infrastructure both for telephone and for office automation and data processing applications. Further, the PABX serves as the gateway to the external communications world. The world of today's telephone, telex and public data networks and of tomorrow's ISDN.

The PABX is in the position to bridge the gap between the two worlds of Telecommunications and Office Automation. However, before the digital PABX can be regarded as the link between Office Automation and ISDN, a lot of problems must be overcome. One of these problems is the subject of this report.

1.1 FORMULATION OF THE PROBLEM

The formulation of the problem, that has to be inverstigated, is depicted as follows:

"...The PABX developed by Philips, the SOPHO-S, offers a digital terminal interface based on a 2-wire u-interface with a proprietary layer 3 protocol. This layer 3 protocol is the Philips Terminal Message Protocol (TMP).
Philips has developed a Private Network Termination that converts the 2-wire u-interface into the 4-wire s-interface conform the CCITT

Recommendations (I.430, I.440). Philips digital terminals can be connected to this PNT which are physically identical to the public ISDN terminals, but on layer 3 still with the proprietary TMP protocol.
Although this kind of terminals (P-terminals) offers far more functionality than the public ISDN terminals, it is to be expected that the public ISDN terminals will become much cheaper and available in many types. <u>Therefore it is useful to examine the possibility of developing an extended version of the existing PNT, upon which public ISDN terminals can be connected.</u>

For this purpose the following must be investigated:

- What parts of the protocols can be converted?

- Describe the layer 3 conversion function using the CCITT specification language SDL;

- Implement the conversion function (partially) in order to demonstrate the feasibility of this conversion.

In first instance, ISDN terminals based on the German version of the ISDN Recommendations (1TR6) must be considered.
..."

An introduction into the concepts of ISDN and SOPHO-S will be given in chapter 2, before a more detailed investigation into the problematic nature can be started in chapter 3.
After SDL terminology is explained, the SDL specification of the PNT layer 3 conversion function is presented in chapter 4. The mapping of this SDL specification on a software implementation is the subject of chapter 5.
Finally, chapter 6 presents the conclusions and recommendations.

CHAPTER 2

ISDN AND SOPHO-S

In this chapter a short introduction will be given into the ISDN concept. Also the structure of a SOPHO-S will be treated. The terminology that will be explained in this chapter will be used frequently throughout the rest of this report.

2.1 THE OSI MODEL

2.1.1 METHODOLOGY USED

In this section the ISO (International Standardization Organization) reference model for Open System Interconnection will be explained. This model is developed in order to provide a framework which can be employed in the development of protocols for communicating devices. A protocol defines the rules and conventions used for this communication [TANE 81]. The OSI reference model is used by many organizations, which deal with the development of protocols. The reason is that this model allows a structured approach for defining protocols. The concepts defined by this model will also be used throughout the rest of this report.

The OSI model consists of seven layers. Each layer performs a well defined function. The boundaries between these layers are chosen so that the information flow across the interfaces is as low as possible.

A layer consists of logical *entities*, which are the active elements in a layer. Two entities in the same layer, which are in separate systems, are called *peer-entities*. These peer-entities communicate by means of a peer-to-peer protocol. The (N)-layer offers services to the (N+1)-layer and uses the services offered by the (N-1)-layer. The point at which (N)-services are provided by an (N)-entity to an (N+1)-entity is called the *Service Access Point*(SAP).

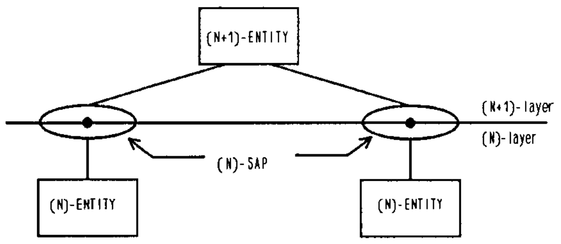


Figure 2.1 *"The layered concept."*

Because a layer can have several entities, it can also have several SAPs (see figure 2.1). Inside a Service Access Point there can be several *Connection Endpoints* (CEs), these are indicated in figure 2.1 with a • . By allowing several CEs in a SAP, multiplexing can be performed from layer (N+1) to layer (N) for one service. For information to be exchanged between two or more (N+1)-entities a data link connection must be established between the entities in the (N)-layer using an (N) *peer-to-peer protocol.*
Interaction between two adjacent layers is performed by means of *primitives*. A primitive may contain parameters. There are four different types of primitives (see figure 2.2).

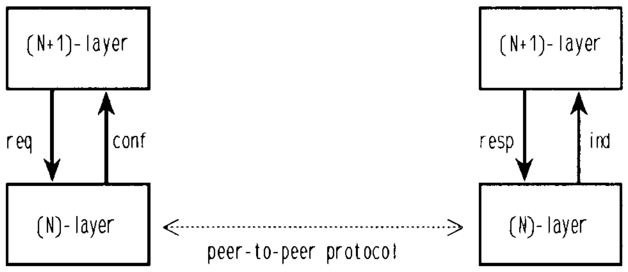


Figure 2.2 *"Communication primitives between layers."*

1. *Request* primitive type, used when an (N+1)-layer is requesting service from an (N)-layer;

2. *Indication* primitive type, used by an (N)-layer to notify the (N+1)-layer of activities related to the primitive type request.

3. *Response* primitive type, used by an (N+1)-layer to acknowledge receipt of the primitive type indication (which came from the (N)-layer).

4. *Confirm* primitive type, used by the (N)-layer providing the requested service, to confirm the (N+1)-layer that the activity has been completed.

2.1.2 THE DIFFERENT LAYERS

As described in the previous section, ISO uses a layered model to model the protocol stack of a communicating device. In this section a brief description will be given about the functions performed by the different layers.

<u>Layer 7:</u> The Application layer.
This is the highest layer in the model. It provides services to the application processes which are also part of this layer. It is up to the user to define the application present in this layer.

<u>Layer 6:</u> The Presentation layer.
The purpose of this layer is to deal with the syntax and semantics of the data to be transmitted, independent of the application layer. Functions like encryption and compression are services offered by this layer.

<u>Layer 5:</u> The Session layer.
This layer manages the dialogue in an orderly manner. It provides synchronization markers to indicate different communication phases. It also determines which service is allowed to send.

<u>Layer 4:</u> The Transport layer.
This layer provides an end-to-end service to the upper layers. Functions like automatic recovery from loss of network connections, multiplexing from different transport connections on a network connection to reduce costs and additional error detection are performed in this layer.

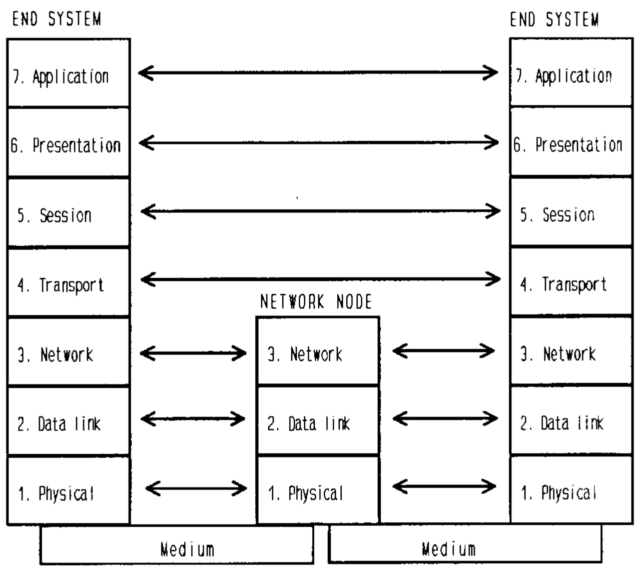


Figure 2.3 *"The OSI reference model."*

Layer 3: The Network layer.

Layer 3 functions provide the means to establish, maintain and terminate network connections between communicating application entities. Functions performed by layer 3 include the following:

- processing of primitives for communicating with the data link layer;
- generation and interpretation of layer 3 messages for peer-level communication;
- administration of timers and logical entities (e.g., call references) used in the call control procedures;
- administration of access resources including B-channels and packet-layer logical channels;
- checking to ensure that services provided are consistent with user requirements (e.g., compatibility, addresses, service indicators).

Layer 2: The Data link layer.

The task of this layer is to transform the errorprone physical channel into a reliable link. Errors in frames are detected and possibly corrected. Also flow control and frame sequencing are functions performed by this layer. The terms "Layer 2" and "L2" are used as abbreviations for data link layer.

<u>Layer 1:</u> The Physical layer.
This layer is concerned with the transmission of bits. Timing, voltages etc. are dealt with in this layer.

Figure 2.3 shows the OSI reference model.

2.2 INTEGRATED SERVICES DIGITAL NETWORK

ISDN is defined by the CCITT (Comité Consultatif Internationale de Télégraphique et Téléphone) as follows:
"An ISDN is a network, in general evolving from a digital telephone network, that provides end-to-end digital connectivity to support a wide range of services, to which users have access by a limited set of standard multipurpose user-network interfaces."
The ISDN concept can be summarized in the following:

- The ISDN supports a range of voice and non-voice services on the basis of a limited set multifunctional user-interfaces;

- The ISDN supports both packetswitched and circuitswitched connections;

- The ISDN evolves from the digital telephone network.

2.2.1 REFERENCE CONFIGURATION

Reference configurations are conceptual configurations useful in identifying various physical and functional user access arrangements to an ISDN. Two concepts are used in defining reference configurations: reference points and functional groupings. Functional groups are sets of functions which may be needed in ISDN user access arrangement. Specific functions in a functional group may performed in one or more pieces of equipment.
Interfaces between functional groups are identified by reference points. In a specific access arrangement, a reference point may correspond to a physical interface between pieces of equipment, or there may not be any physical interface corresponding to the reference point. Figure 2.4 shows the reference configurations for the ISDN user-network interfaces.

Figure 2.4.a defines the reference configuration with the functional groups NT1, NT2 and TE1. Figure 2.4.b illustrates that TE1 may be

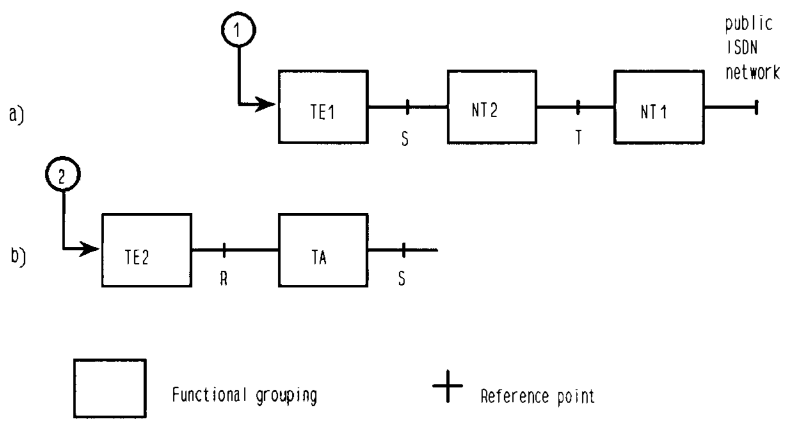


Figure 2.4 **"Reference configurations for the ISDN user-network interfaces."**

replaced by the combination of TE2 and TA. Lists of functions for each functional group are given below.

<u>NT1, Network Termination 1</u>, includes functions broadly equivalent to layer 1 of the OSI reference model. These functions are associated with the proper physical and electromagnetic termination of the network. NT1 functions are:

- line transmission termination;
- layer 1 line maintenance functions and performance monitoring;
- timing;
- power transfer;
- layer 1 multiplexing;

<u>NT2, Network Termination 2</u>, includes functions broadly equivalent to layer 1 and higher layers. PABXs and local area networks are examples of combinations of equipment that provide NT2 functions. NT2 functions include:

- layers 2 and 3 protocol handling;
- layers 2 and 3 multiplexing;
- switching;

<u>TE, Terminal Equipment</u>, includes functions broadly belonging to

layer 1 and higher layers. Digital telephones and data terminal equipment are examples that provide TE functions.

TE1, Terminal Equipment 1, includes functions belonging to the functional group TE, and with an interface that complies with the ISDN user-network Recommendations.

TE2, Terminal Equipment 2, includes functions belonging to the functional group TE but with an interface that complies with interface Recommendations other than the ISDN interface Recommendations.

TA, Terminal Adapter, includes functions that allow a TE2 terminal to be served by an ISDN user-network interface.

2.2.2 ISDN SERVICES

An user has, via the various reference points, access to the telecommunication services offered by the ISDN. These services can be divided in two main groups:

1. bearer services;
2. teleservices.

Bearer services are defined at the S- or T-reference point and use broadly the lowest 3 layers of the OSI model. An example of a bearer service is a 64 kbit/s transparent circuitswitched connection.
Teleservices are defined at points 1 and 2 in figure 2.4 and use broadly layers 1 up to 7 of the OSI model. At the lower layers these services use some bearer service. Telephony is an example of a teleservice.

2.2.3 ISDN USER-NETWORK INTERFACE CAPABILITIES

Up to now, CCITT has defined two user-network reference points S and T. At these reference points one or more channels are defined with the following structures:

- Basic access 2 B + D(16)
- Primary rate access 30 B + D(64)

The B channel (capacity 64 kb/s) is used to convey information and the D channel is used for signalling information.

In the CCITT Recommendations is defined that a maximum of 8 physical terminals may be connected to a s-interface. In case of a basic access s-interface, only 2 of this maximum of 8 physical terminals can use a B-channel simultaneously.

2.2.4 ISDN LAYER 2 D-CHANNEL PROTOCOL

The function of layer 2 is protection of layer 3 information against transmission errors. A well known layer 2 protocol is the High Level Data Link Control, HDLC. The ISDN layer 2 D-channel protocol, the Link Access Procedure on the D-channel (LAPD), is based on this HDLC protocol [Q.921]. The structure of a layer 2 frame is given in figure 3.5.

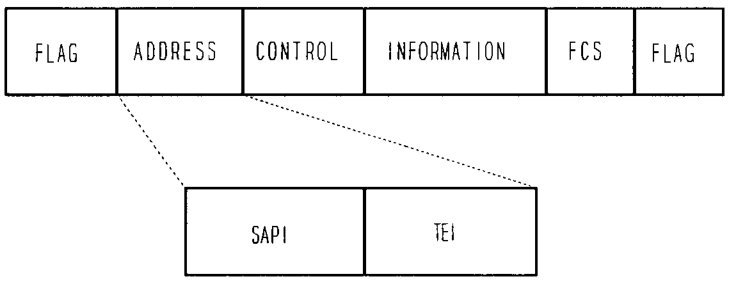


Figure 2.5 *"Structure of a layer 2 frame."*

The frame is marked at the begin and end with an unique bit-pattern. The errorprotection of a layer 2 frame is handled by a Frame Check Sequence (FCS). FCS takes care of faultdetection; faultcorrection takes place by retransmission of frames.
In order to handle multiple terminal installations at the network interface as well as multiple layer 3 entities such as signalling and packet communication, LAPD has the capacity of supporting multiple data link connections simultaneously on a single D-channel. In order to identify a specific data link connection, the layer 2 address field consists of two subfields, namely, the data link

Service Access Point Identifier (SAPI) subfield and the Terminal Endpoint Identifier (TEI) subfield. (See figure 2.5).
According to OSI layered model terminology, SAPI identifies a point at which data link layer services are provided to layer 3. For example, SAPI=0 identifies a layer 2 service intended for call control procedures.
The TEI identifies a specific endpoint within a SAP. Normally a TEI value identifies a specific logical terminal (Notice that a physical terminal can consists of one or more logical terminals). A group TEI is assigned to the broadcast (point-to-multipoint) data link. All other TEI values are used for point-to-point data links.
A TEI value can be assigned to a logical terminal in one of the following ways:

- Values 0 - 63 are reserved for TEIs that are static adjusted in a terminal;
- Values 64 - 126 are dynamically assigned by the network via the TEI assignment procedure;
- TEI value 127 is the global or broadcast TEI. This TEI value is used if a message has to be sent to all data link endpoints inside a SAP. For example, the layer 3 message SETUP (indicating that a call is present) will be sent by the network layer 2 with address TEI=127, SAPI=0.

2.2.5 ISDN LAYER 3 D-CHANNEL PROTOCOL

The ISDN layer 3 D-channel protocol is defined in Recommendations [Q.931]. These Recommendations describe the procedures for the establishing and clearing of network connections such as follows:

- Circuit-switched connections using the B-channel;
- User-to-user signalling connections using the D-channel;
- Packet-switched connections using either the D-channel or the B-channel.

Layer 3 defines the various signalling messages, such as the request to set up a call and to break off a connection. A layer 3 message consists of a header, followed by one or more information elements (Figure 2.6).

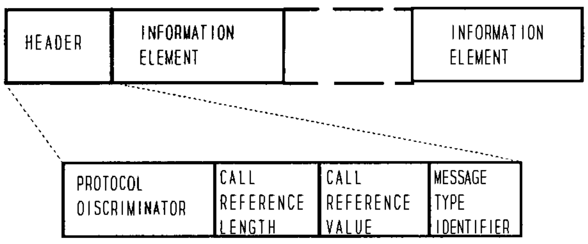


Figure 2.6 "*Structure of a layer 3 message.*"

The Protocol Discriminator indicates the type of protocol, according to which the messages are coded.
Each incoming and outgoing call is identified by a call reference. All messages related to the same call have the same call reference.
The message type identifier identifies the type of the signalling message: *SETUP*, *CONNECT*, etc..
A signalling message may contain one or more information elements, such as display-information, digit-information or channel identification.

2.3 SOPHO-S

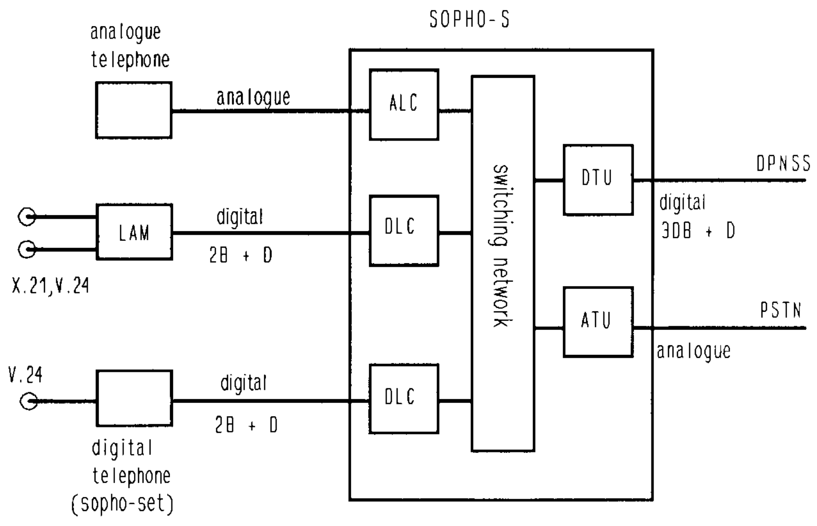
Synergetic Open PHilips Office Switch (SOPHO-S) is the name of the present line of digital PABXs developed by PTDSN Hilversum. The SOPHO-Switches belong to the office automation philosophy of Philips called SOPHOMATION (Synergetic Open PHilips Office autoMATION). SOPHO-S is based on ISDN principles:

- SOPHO-S offers a (2B+D) digital terminal interface based on the 2-wire u-interface with a proprietary layer 3 protocol;

- SOPHO-S offers various primary rate (30B+D) Digital Trunk Units. For example DPNSS, for signalling over leased lines.

Figure 2.7 shows a functional decomposition of SOPHO-S.

2.3. SOPHO-S



ALC= Analogue Line Cicuit
ATU= Analogue Trunk Unit
DLC= Digital Line Circuit
DTU= Digital Trunk Unit-Primary Rate
LAM= Line Adapter Module
DPNSS= Digital Private Network Signalling System
PSTN= Public Switched Telephone Network

Figure 2.7 *"Functional decomposition of the SOPHO-Switch."*

The SOPHO-S digital terminal interface uses a proprietary layer 3 protocol, called Terminal Message Protocol (TMP). The following categories of functions are provided by this layer 3 signalling protocol:

- Extension-equipment initialization;
- Downloading of parameter settings and function key contents;
- Connection establishment and disestablishment;
- User-to-user signalling;
- Execution of operational maintenance commands.

In contrary with the ISDN s-interface, the SOPHO-S digital terminal interface supports only <u>one</u> connected terminal. This terminal has two *ports*, one for each B-channel. The ports are addressed by SOPHO-S by means of separate portnumbers. In the first message that is sent to a port, SOPHO-S assigns one of the two B-channels to the port. From now, the port is the sole owner of the indicated B-channel. Due to this fact, the TMP protocol, active on layer 3 of the port, expects a fulltime present B-channel!

Currently, there exist two, basic types, SOPHO-S terminals: the SOPHO-SET and the LAM (Line Adapter Module). The SOPHO-SET has one voice port and one data port; the LAM offers two data ports. Various terminal models, based on one of these basic types, are available.

NOTE:
In contrary with TMP, the ISDN layer 3 does not expect a fulltime present B-channel. A terminal can use a B-channel only if it is not in use by an other logical terminal that is also connected to the s-interface.

CHAPTER 3

SOPHO-S FROM PABX TO ISPBX

3.1 INTRODUCTION

When looking at ISDN introduction scenarios in the various countries one should be aware of the effort that still is required in the standardization process and also take into account that the construction of a digital infrastructure in itself is already an enormous task. Interesting in this respect is the commitment of the EEC countries to achieve an ISDN coverage of 5% by the end of 1992. Who are these new customers? Apparently not the residential users, the homes. For them even the initial ISDN functions exceed most of the wishes and almost certainly their budgets. No, the users most eager to start with ISDN are in the business community. They have a justification for the improved functionality that ISDN may offer over the traditional communication means and therefore the money. And more important, a company-wide scale is much easier to manage than a nation-wide or even larger scale. Historically, however, the business community is served by PABXs. This leads to an interesting confrontation. On the one hand the public ISDN providers (the PTTs) try to get hold of this market by offering services via the public ISDN network or approaching the very small business without an immediate need for PABX services. On the other hand the PABX suppliers try to secure their share of the market by portraying their systems as Integrated Services PABXs (=ISPBXs). Is there such a thing as an ISPBX and if so what are the characteristics?

In the next sections, the position of the PABX in the ISDN network, will be discussed. In the ISDN reference configuration, the PABX is positioned as the NT2 functional grouping; see figure 3.1.

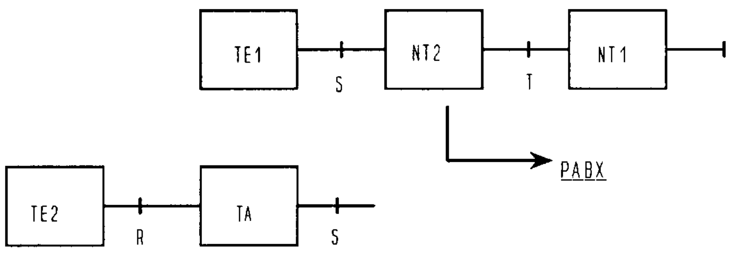


Figure 3.1 "*ISDN reference configuration model.*"

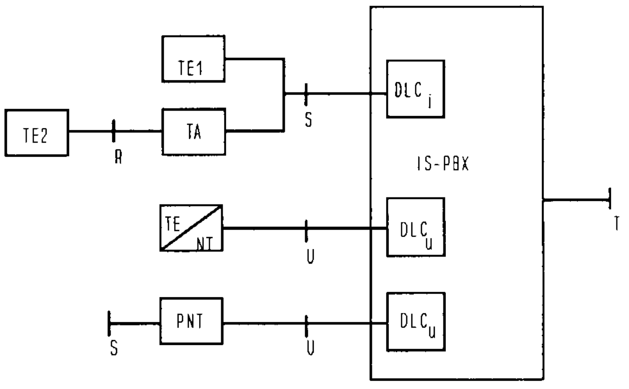
3.2 PABX CONNECTION TO PUBLIC ISDN

Up to now mature international signalling standards for PABX to public ISDN over primary rate interfaces, were not available. For that reason primary rate equipment with nationally defined channel associated signalling is being installed in various countries.
The CCITT basic access and primary rate interfaces can both be applied at reference point T (i.e. between PABX and public ISDN). The choice for one of them depends on the required number of 64 kbit/s trunk channels.
The SOPHO-S PABX supports a pre-ISDN scenario based on primary rate interfaces with channel associated signalling: DPNSS. Digital Private Network Signalling System is a private network signalling system for signalling over leased lines. Layer 1 and 2 specifications are nearly identical to the specification DASS II, the UK standard for PABX to public ISDN signalling. An interesting option of DPNSS is the usage of the public ISDN instead of leased lines!
Since it is expected that the standardization process takes quite some years, in the meantime DPNSS serves as a pre-ISDN standard.

3.3 PABX INTERFACES FOR ISDN EQUIPMENT

In order to transform the PABX to ISPBX, the PABX should have built-in s-interface circuits to which the terminals TE can connect directly, either in a single or multi-terminal arrangement. One of the disadvantages is that the s-interface is a 4-wire interface and most of the networks in place are still 2-wire. Another problem is that the s-interface may not cope with the terminal-to-PABX distance requirements. The distance limitations specified within the Recom-

mendations range from 1 km(single terminal) to 200 meters(multi-terminal). The average loop length of a medium sized PABX is approximately 600 meters.
Because of the wiring and distance requirements mentioned above, a cost-effective 2-wire private network is under discussion within the various standardization bodies in Europe. The Philips PABX SOPHO-S uses such a 2-wire echo canceller based private u-interface.
The outcome of the ISPBX terminal issue in Europe will probably that the S-terminals of the public ISDN can be connected to either 4-wire s-interface circuits integrated in the ISPBX(short loop) or NTs behind a 2-wire u-interface circuit(long loops). The NT can be integrated either in the terminals (TE/NT) or constructed as a PRIVATE NETWORK TERMINATION (PNT) to convert the "U" to standard "S". Figure 3.2 illustrates the different PABX interfaces for ISDN equipment.



DLCi = Digital Line Circuit s-interface
DLCu = Digital Line Circuit u-interface

Figure 3.2 *"ISPBX interfaces for ISDN terminals."*

3.4 THE PNT1

A more detailed description of the function of the Private Network Termination is given in figure 3.3. This figure illustrates the PNT OSI-model.

Development of the PNT includes both hardware and software conversion. The hardware conversion implies the conversion of the 4-wire ISDN s-interface into the 2-wire SOPHO-S u-interface.

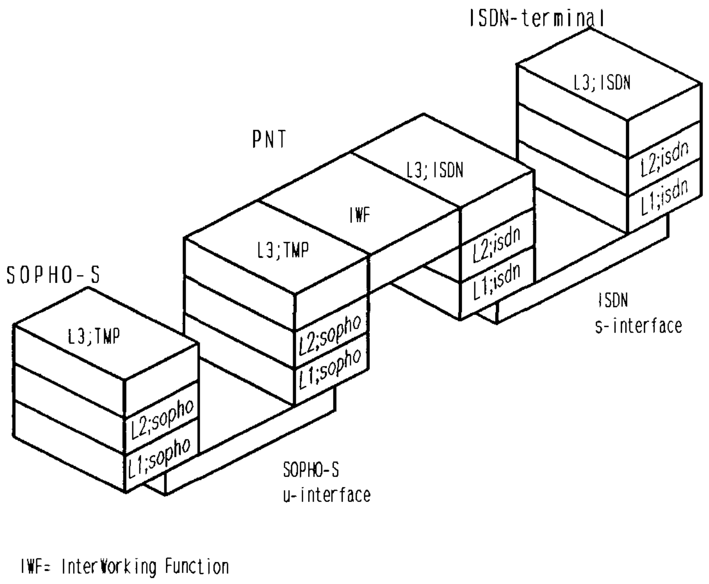


Figure 3.3 *"The PNT OSI-model."*

The software conversion implies the conversion of a layer 3 ISDN protocol into the layer 3 TMP protocol (i.e. SOPHO-S proprietary protocol).

As a first stage, Philips developed the PNT1 that takes care of the hardware conversion. The PNT1 is made in the first instance to bridge the difference in the transmission medium of the ISDN s-interface (4 wires) and that of the SOPHO-S echo-canceling based u-interface (2 wires). <u>Not the protocol conversion is implemented in the PNT1, but rather a message relay</u>. Consequently, the PNT1 supports only terminals with ISDN implementations on layers 1 and 2 and the TMP protocol on layer 3. This kind of terminals will be denoted as *P-terminals*. Figure 3.4 illustrates the PNT1 OSI-model.

Because the PNT1 is useless without a P-terminal, Philips also developed a P-terminal.

P-terminals have fixed TEIs (=Terminal Endpoint Identifiers):

P-terminal with TEI=0	:terminal uses B0-channel;
P-terminal with TEI=1	:terminal uses B1-channel;
P-terminal with TEIs=0 & 1	:terminal uses both B0 and B1.

3.4. THE PNT1

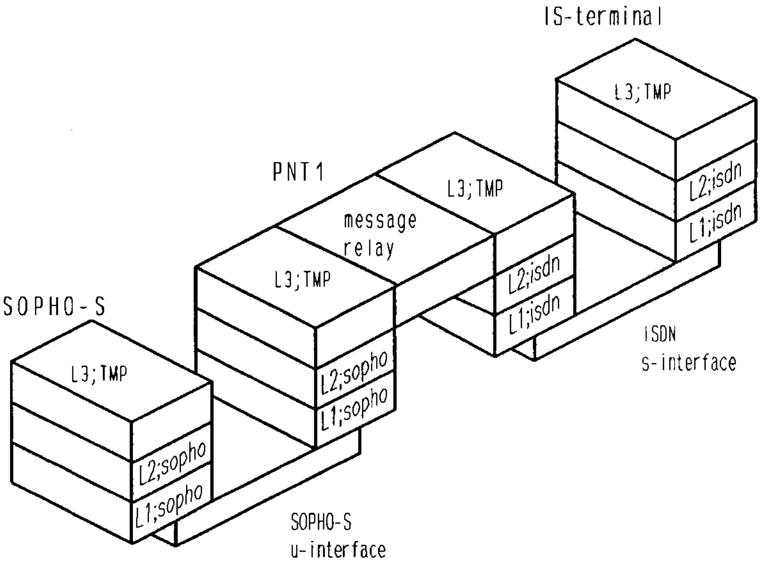


Figure 3.4 *"The PNT1 OSI-model."*

The TEI value of a terminal is used by the PNT1 layer-2-ISDN-side to address a specific terminal.

A P-terminal expects a fulltime present B-channel; this fact is inherent to the layer 3 protocol (=TMP) of this terminal. As a consequence a maximum of 2 P-terminals can be connected to the PNT1; one for each B-channel.

PNT1 Layer 3 function:
As stated before, the PNT1 performs at layer 3 not a real conversion but rather a message relay between the ISDN s-interface and the SOPHO-S u-interface. This means that at layer 3 level the PNT1 does not manipulate the layer 3 messages it receives but simply transmits them to the other side.

PNT1 Layer 2 function:
At layer 2 level the PNT1 maintains the links at both sides:
SOPHO-S <--> PNT1 and PNT1 <--> P-terminals.

At the SOPHO-S side the PNT1 supports the SOPHO layer 2 protocol and at the P-terminal side the PNT1 supports the ISDN layer 2 protocol.

PNT1 Layer 1 function:
At layer 1 level the PNT1 converts all ISDN physical signals into SOPHO-S physical signals.

3.5 THE PNT

In order to be able to connect public ISDN terminals to SOPHO-S via a Network Termination, the PNT1 (see figure 3.4) has to be transformed to the PNT (see figure 3.3). It shows, by comparing both figures, that the PNT will have the same hardware and layers 1 and 2 as the PNT1.
The difference between the PNT and the PNT1 is the layer 3 function: The PNT1 is transparent on layer 3 (i.e. message relay) and the PNT layer 3 must consist of an interworking function that performs a conversion between TMP and an ISDN protocol. As a consequence, we may only connect ISDN terminals to the PNT !

Because layer 3 function is implemented in software, it seems possible to combine both the PNT and the PNT1 layer 3 software in one and the same network termination. Both P-terminals and ISDN terminals could be connected to this new PNT. See figure 3.5.

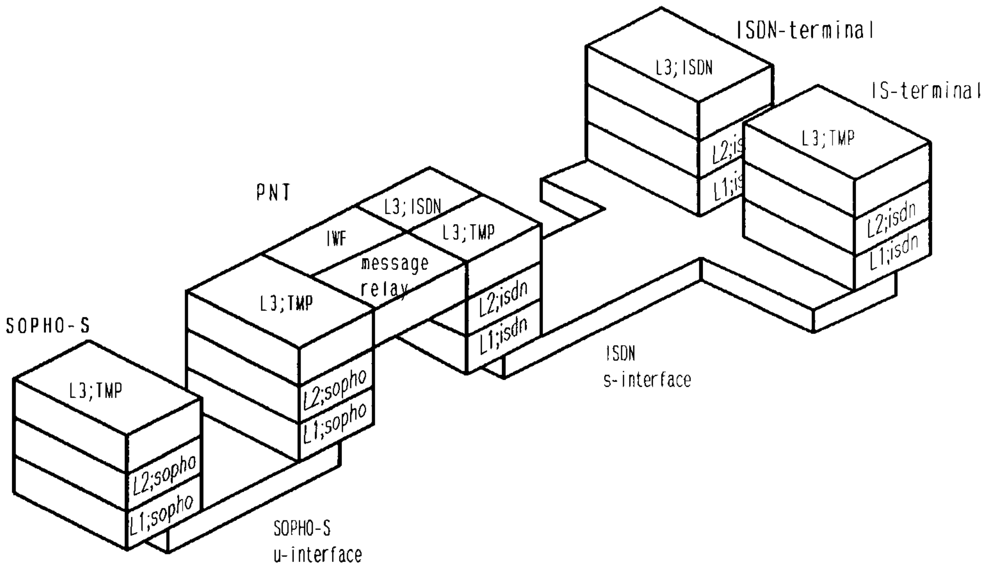


Figure 3.5 *"PNT OSI-model."*

<u>Remark on figure 3.5:</u>

The difference between P-terminals and ISDN terminals occurs on layer 3, and not on layers 1 and 2. <u>The PNT layer 2 terminal-side</u>

treats P-terminals and ISDN terminals in the same way. The PNT layer 2 terminal-side addresses a specific terminal by its TEI value, and is not aware of the type of the terminal (P- or ISDN terminal). Moreover, this is conform the OSI concepts, because the layer 2 task is to convey layer 3 messages on a peer-to-peer connection, regardless the kind of layer 3 function.

However, is it desirable to develop a PNT upon which both ISDN terminals and P-terminals can be connected? The answer to this question must be affirmative because of the following reasons:

- The combination-PNT replaces two PNT versions;
- The combination-PNT offers the client the option to connect a combination of P-terminals and ISDN terminals, without the necessity of changing the network termination:

The PNT of figure 3.5 offers the possibility to connect, for example, one sophisticated telephone (i.e. a P-terminal) and one public ISDN FAX-G4-terminal, to one and the same network termination.

Note:
From this point PNT refers to the PNT as described in figure 3.5.

As a direct consequence of connecting both P-terminals and ISDN terminals to the PNT, the following 2 restrictions occur:

Restriction 1:
P-terminals have fixed TEIs and ISDN terminals can have fixed or non-fixed TEIs. Therefore it is not allowed to connect fixed ISDN terminals with TEI=0 or 1 to the PNT; otherwise the PNT layer 2 can not distinguish between ISDN- and P-terminals.

Restriction 2:
A P-terminal expects a fulltime present B-channel. As a consequence we have to assign a higher priority to P-terminals than to ISDN terminals.
For example, a P-terminal with TEI=0 will always become the owner of the B0-channel. In the case that a P-terminal with TEI=0 and an ISDN terminal are connected to the PNT, the ISDN terminal cannot setup a

call requesting for the B0-channel because the P-terminal 'owns' this channel. So, in this case, the ISDN terminal should setup a call using the B1-channel.

At this point, the problem definition of the PNT layer 3 is completed. In the next chapter, the specification of this layer 3 function can be started.

CHAPTER 4

SPECIFICATION OF THE PNT LAYER 3

The specification of the PNT layer 3 function will be in SDL. What is SDL?
SDL stands for Specification and Description Language and is defined by the CCITT. It is a formal specification language specially designed for specifying telecommunication systems.
Why SDL? The main reason for using SDL as the specification language for the PNT is the fact that it has been developed, standardized and used by the CCITT and therefore accepted in the world of telecommunication. Secondly, it is a formal language that is supported by computer tools and thus can lead to more complete and more consistent specifications. Thirdly, it is an executable specification language that can be used for verification by simulation.

4.1 SDL: CCITT'S SPECIFICATION AND DESCRIPTION LANGUAGE

4.1.1 INTRODUCTION

SDL has been designed by the CCITT as a vehicle to specify and describe the behaviour of telecommunication systems. Its design started as early as in 1972 when study group XI of the CCITT started to design a programming language, a specification and description language and a Man/Machine interaction Language (MML). This work resulted among others in a recommendation in which a graphical method was specified to describe an extended finite state machine. The language was called CCITT Specification and Description Language (SDL).
During the period 1976-1988 the language has been thoroughly extended and revised. The semantics of the language were defined formally, so that it became possible to develop support tools to verify the specifications.

SDL can be used to specify the required behaviour and to describe the actual behaviour of a system. It can also be the base for a methodology of documentation to represent completely a system

specification. Furthermore, it may be used to represent, at various levels of detail, the functional properties of a system.
Specifications and descriptions in SDL can be written in a partially **formal** or in a total **formal** way. Partially formal specifications contain statements in natural language, so a wider range of people can understand it. On the other hand, totally formal specifications only contain formal statements.

SDL is based on an Extended Finite-State Machine (EFSM) model provided with the following properties:

- every combination of input and state yields an output and a next state (an ordinary property of an FSM);
- the FSM is extended with auxiliary storage operated on by auxiliary operations;
- auxiliary operations are: **decision**, which makes a choice between several alternatives depending on parameters associated with inputs and on information in auxiliary storage; **task**, performs actions on auxiliary storage and manipulates input and output parameters;
- interactions between EFSM's are represented by signals which are optionally parameterized;
- communication between EFSM's is asynchronous implemented by FIFO queues.

SDL can be represented in two different forms depending on the same semantic model:

- Graphical Representation (SDL/GR) using graphical symbols;
- Phrase Representation (SDL/PR) using program-like statements.

SDL/GR is better suited for expressing ideas and allows the control flow to be easily visualized, while SDL/PR is preferable for use on a machine, because checks can be placed upon it more easily.

4.1. SDL:CCITT'S SPECIFICATION AND DESCRIPTION LANGUAGE

As SDL is a formal language it is possible to support the language with computer tools. There are a number of tool-packages commercially available. The tool-package that is used to specify the PNT is the SDT (SDL Design Tool) package from TeleLogic Sweden [SDT 89]. The following modules are available at this moment:

- SGE: (SDT Graphical Editor) a graphical editor to edit and maintain SDL descriptions in the Graphical Representation (GR).
- SRG: (SDT Report Generator) a report generator. This report generator is used to print the SDL-diagrams made with SGE.
- SAC: (SDT Analyzer and Converter) this module comprises the syntax and semantic analyzer tools, the syntax conversion tools and the simulator generator. The conversion tools convert the Graphical Representation to the Phrase Representation (PR).
- UTILITY: Utilities to maintain the databases of the SDL descriptions.
- MAINT: Utilities to create and delete SDL databases.

The SDL specification of the PNT has been edited graphically using SGE/PC version 2.02. A communication program (for example Kermit) must be used to transfer the GR files to the VAX. Further, the SAC tool on the VAX system is used to convert this graphical specification into the PR form. Next SAC checks the syntax and semantic of the specification. If SAC does not detect any irregularities, a simulator can be generated.

4.1.2 BASIC CONCEPTS OF SDL

Systems in SDL can be modelled by using building blocks for expressing the structure of a system. The behaviour of these blocks is specified by using processes, extended finite-state machines communicating through signals.

Building blocks
A **system** in SDL is composed of a number of blocks connected by channels. Each block may contain one or more processes, which describe the behaviour of the block. The only means of communication

between processes in two different blocks is by sending signals that are transported by the channels (see Figure 4.1).

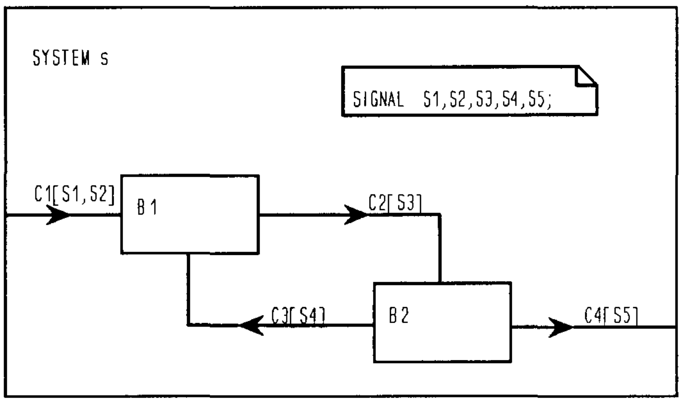


Figure 4.1 *"A SDL system diagram."*

Within a block processes can communicate with one another either by signals or shared variables. The signals between processes in the same block and between a process and the environment of the block are carried by signalroutes (see Figure 4.2).

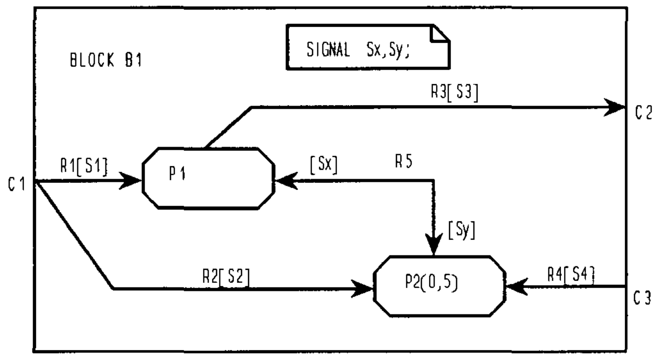


Figure 4.2 *"A SDL block diagram."*

A **channel** connects two different blocks in the system or a block and the environment of the system. It can be unidirectional or bidirectional. When a signal is delivered to a signalroute leading to a block boundary, the signal is given to the channel connected to the signalroute. When a signal arrives at a block from a channel and the

channel is connected to one or more signalroutes, the signal is delivered to the signalroute containing the identifier of the signal in its signal list. The signal list of an incoming channel is partitioned into disjunct sets, one set per signalroute connected to the channel. The signal list of an outgoing channel is partitioned into sets, not necessarily disjunct, one per signalroute connected to the channel.
A **signal** consists of a signal name and possible a number of parameters carried with the signal. Signals are thus typed, whereas channels and signalroutes are not typed. Signals are used for synchronization between processes and for carrying information from one process to another.

Processes

A **process** is an extended finite-state machine, defining the behaviour of a part of a system. Each process has one input queue in which input signals for the process are stored until the process is able to receive them. Thus, there is no separate signal queue per signalroute or signal type. Figure 4.3 illustrates an example of a process.

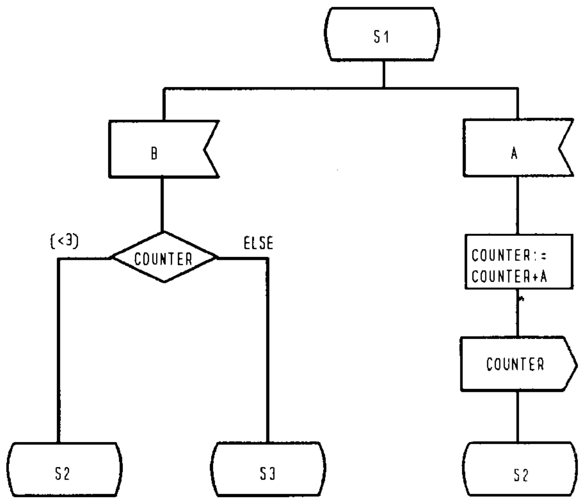


Figure 4.3 *"SDL process state transitions."*

Data handling

Variables can be declared in a process and are local to that process. Only the owner process can change the value of the variables. Processes in the same block can share variables. The process owning the shared variable must declare it as a revealed variable. The process wanting to share the variable must declare a viewed variable with the same identifier and the same sort as the one being shared.

The viewing process can not modify the value of the viewed variable. A process can export a variable to all other processes, not necessarily in the same block. The exporting process must declare the variable as exportable and the importing process must declare a variable with the same name and sort as importable.

A more detailed description of the SDL concepts is beyond the scope of this report. The interested reader is referred to [BELI 89].

4.2 PNT PROCESSES

4.2.1 PNT SYSTEM DIAGRAM

The first stage in the specification of the PNT is the definition of the PNT *system diagram*. Because the PNT layer 3 function is under consideration, the system diagram will consist of one *block* called PNT_LAYER_3. See also figure 4.4.

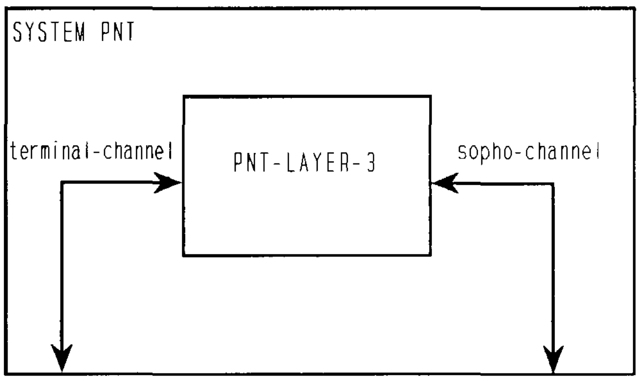


Figure 4.4 *"The PNT system diagram."*

Two *channels* are connected between the block PNT_LAYER_3 and the environment. One channel to convey messages to/from PNT-layer-2-SOPHO-side, and one channel to convey messages to/from PNT-layer-2-ISDN-side.

4.2. PNT PROCESSES

4.2.2 PNT BLOCK DIAGRAM

The PNT layer 3 function is twofold:

- Supporting communication between SOPHO-S and ISDN terminals. An interworking function is needed which performs a protocol conversion between the ISDN protocol 1TR6 and the SOPHO-S protocol TMP;

- Supporting communication between SOPHO-S and P-terminals. The PNT layer 3 must include a message relay function to deal with this kind of communication.

The PNT_LAYER_3 block diagram will consist of 4 processes in order to support both functions described above. Figure 4.5 illustrates this PNT_LAYER_3 block diagram.

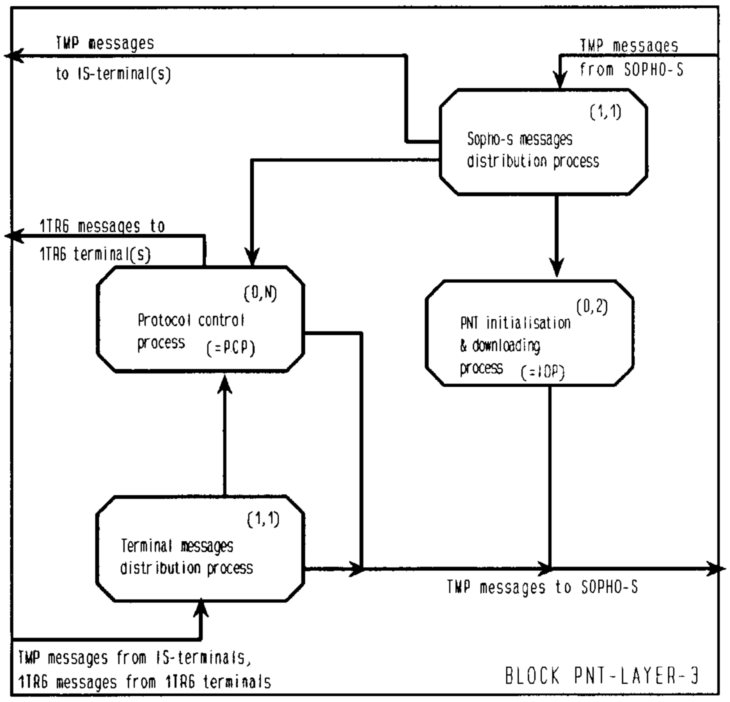

TMP messages to IS-terminal(s)
TMP messages from SOPHO-S
(1,1) Sopho-s messages distribution process
1TR6 messages to 1TR6 terminal(s)
(0,N) Protocol control process (=PCP)
(0,2) PNT initialisation & downloading process (=IDP)
(1,1) Terminal messages distribution process
TMP messages to SOPHO-S
TMP messages from IS-terminals, 1TR6 messages from 1TR6 terminals
BLOCK PNT-LAYER-3

Figure 4.5 *"The PNT_LAYER_3 block diagram."*

The Protocol Control Process (PCP) is the main process. Conversion between TMP messages and 1TR6 messages is carried out by this process. For each new call a separate PCP process instance will be created and after call completion this process instance will be removed again.

When a SOPHO-S digital extension line is taken into service, the SOPHO-S terminal (the PNT in this case) connected to this extension interface will be initialized and downloaded. A separate Initialization & Downloading Process (IDP) will deal with this initialization and downloading. The reason for not including these matters in the PCP process, is that the PNT initialization has nothing to do with the PCP conversion function. Two IDP instances will be active; one for each B-channel.

The "SOPHO-S messages distribution process" distributes the incoming messages from SOPHO-S. A TMP-message destined for a P-terminal will be passed directly to the PNT-layer-2-ISDN-side (message relay). A TMP message destined for a 1TR6 terminal will be passed to the conversion process PCP. Finally, messages concerning initialization and downloading will be passed to the IDP process.

The "terminal messages distribution process" distributes the messages from the terminals. A TMP message from a P-terminal will be passed directly to the PNT-layer-2-sopho-side (message relay). Messages from 1TR6 terminals will be passed to the conversion process PCP.

The SDL specification of the 4 processes from figure 4.5 is carried out by a detailed inspection of, on the one hand the TMP specification and on the other hand the 1TR6 layer 3 specification. The 1TR6 specification is given in SDL [1TR6 90]. The TMP specification, however, was incomplete and given in CCS [MIL 80]. A new consistent TMP specification in SDL has been made, [TMP 90], because it is convenient to have both protocols already specified in SDL when generating a SDL specification of the interworking function of these two protocols.

As a first stage, only the basic call control, i.e. messages to setup, maintain or break off a call, is considered in this report. A protocol conversion that also includes supplementary services, such as "automatic ring-back", "enquiry", "add-on conference" and "follow-me" should be subject for further study. Nevertheless in section 4.7 more will be said about the feasibility of implementing these supplementary services.

4.3 PNT INITIALIZATION PROBLEM

As a direct consequence of the rather different natures of the TMP protocol and the 1TR6 protocol, the following problem arised during the specification of the PNT layer 3 IDP process.

Situation:

When SOPHO-S takes a digital extension line into service, a port (or B-channel) projection has to be made via Operational Maintenance (OM) procedures.

Note:
As stated before in section 2.3, a port is directly related to a B-channel. Therefore it is allowed to speak of B0- and B1-channel in stead of port A and port B.

Four different projections are possible:

B0-channel:	B1-channel;
DATA	DATA (the LAM projection)
VOICE	VOICE
VOICE	DATA (the SOPHO-SET 375D projection).
DATA	VOICE

Consequently, the OM-procedures decide whether the B-channel will be used for data communication or for voice communication.
Once this projection has been made, a terminal corresponding to this projection should be connected to the extension line.
The projection cannot be changed by the terminal, but only by intervention of the OM-procedures.
How can SOPHO-S determine whether a correct type of terminal is connected? (Correct type means corresponding to the projection).
For each port/B-channel, SOPHO-S sends an initialization message to the terminal and the terminal must react to this message with either a voice-confirmation or a data-confirmation. Next, SOPHO-S compares this answer with the projection and in the case that the two do not match, SOPHO-S will not continue the initialization procedure and an alarm will be generated. It is important that this situation will

not occur! A terminal that does not correspond to the projection, should not be connected to the extension line.

<u>PNT problem:</u>

Each type of terminal connected to SOPHO-S performs a standard confirmation according to its fixed function. For example a LAM will always confirm both B-channels with data.
However the PNT cannot perform a standard fixed confirmation, because various terminal configurations can be connected to the PNT. How should the PNT confirm after an initialization message is received from SOPHO-S? Consider two possible solutions for this problem:

4.3.1 STATIC SOLUTION

The PNT is informed about the projection by means of a dipswitch setting in the PNT hardware. Figure 4.6 illustrates this solution.

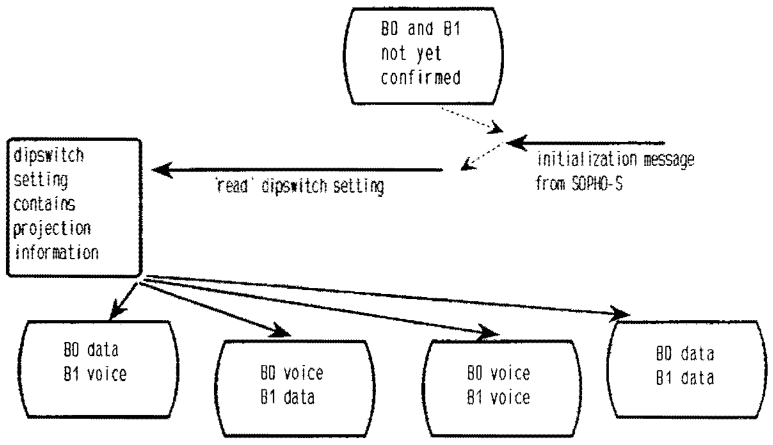


Figure 4.6 *"The static solution."*

In this case the PNT can always send the voice/data confirmation irrespective of the fact whether there are any terminals connected to the PNT!
The PNT is also able to disregard a wrong type of terminal. Wrong type means here not corresponding to the projection information adjusted by the dipswitches.
The person who adjusts the dipswitch setting is responsible for the

correct confirmation. This person should adjust a setting corresponding to the projection information.

Drawbacks:
The PNT has to offer the possibility to adjust a projection; this requires extra hardware.

4.3.2 DYNAMICAL SOLUTION

The PNT determines the types (data or voice) of connected terminals after receiving the initialization message from SOPHO-S. The types of the found terminals define whether the PNT will confirm with data or with voice. See figure 4.7.

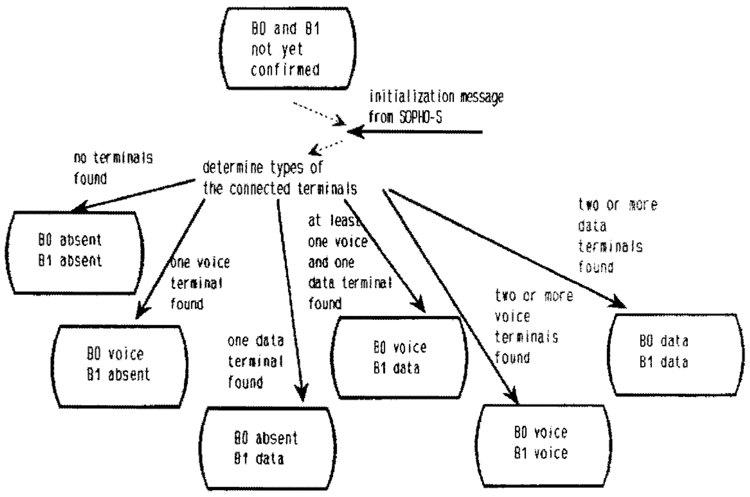

B0 and B1 not yet confirmed
initialization message from SOPHO-S
determine types of the connected terminals
no terminals found
B0 absent B1 absent
one voice terminal found
B0 voice B1 absent
one data terminal found
B0 absent B1 data
at least one voice and one data terminal found
B0 voice B1 data
two or more voice terminals found
B0 voice B1 voice
two or more data terminals found
B0 data B1 data

Figure 4.7 *"The dynamical solution."*

Comment on figure 4.7:

1. Instead of a v/d-confirmation message, it is also allowed to send a *port_absent* message to SOPHO-S in response to the initialization message. In this case SOPHO-S will break off the initialization procedure for this B-channel. In a later stage, the PNT can send a *link_up* message to SOPHO-S in order to restart the initialization.

2. The PNT is not able to distinguish between a B0=voice, B1=data projection and a B0=data, B1=voice projection. As a conse-

quence of this fact, one of these two projections has to be excluded!
A similar agreement has been made for the SOPHO-SET 375D terminal. This terminal always confirms the B0-channel with voice, and the B1-channel with data. Therefore the SOPHO-S projection B0=data, B1=voice should never occur.
The same agreement will also be made for the PNT:

- When the PNT finds **at least** one data- and one voice-terminal, the PNT will always confirm the B0-channel with voice and the B1-channel with data;

- When the PNT finds one voice terminal, the PNT will always confirm the B0-channel with voice;

- When the PNT finds one data terminal, the PNT will always confirm the B1-channel with data;

- The SOPHO-S projection B0 =data, B1=voice should never occur.

The person who connects the terminal(s) to the PNT is responsible for the correct confirmation. This person should only connect those types of terminals, that correspond to the projection.

How can the PNT determine the types of terminals connected to its s-interface? For this purpose the following important property of the ISDN concept will be used:

- Only terminals offering a service, corresponding to the service indicator indicated in the setup message, may respond to this setup message.

The following strategy, based on this property, is proposed in order to determine the types of connected terminals:

A. A service indicator list consisting of all valid service indicator values must be present in the PNT.

B. Send a fake (i.e. there exists no calling party) setup message via the broadcast data link to all connected terminals. In this setup message a service indicator from the service indicator list is indicated.

C. Next, administer all those terminals that respond to this setup message and clear the corresponding point-to-point connections (i.e. there exists no calling party).

D. Send another fake setup message with the next service indicator from the service indicator list.

E. Repeat steps C and D until all service indicators from the service indicator list are checked, or as soon as terminals have respond to <u>two</u> different service indicators.

After step E, the PNT knows the various types of connected terminals and the confirmation procedure of figure 4.7 can be carried out.

The dynamical solution is preferred above the static solution, because it is undesirable that the development of the PNT layer 3 should impose modifications on the PNT hardware. The SDL specification of the dynamical solution can be found in the PNT layer 3 SDL specification in Appendix A.

4.4 PNT S-INTERFACE VERSUS THE ISDN S-INTERFACE

The PNT defines a s-interface within the SOPHO-S environment. The required layer 3 protocol conversion between TMP and the 1TR6 protocol involves some consequences. This fact is inevitable because of the rather different nature of both protocols. Does an ISDN terminal connected to the PNT offer the same functionality as an ISDN terminal connected to an ISDN network?
The s-interface defined by the PNT has the following two restrictions:

<u>The service indicator problem</u>
When an incoming call arrives from SOPHO-S on a data B-channel, the PNT sends a setup message via the broadcast data link. A data service indicator has to be included in this setup message. Because SOPHO-S does not send any further information concerning this data call, the PNT cannot determine what data service indicator should be chosen. As a consequence only data terminals of <u>one</u> type must be present at the PNT's s-interface, so that the PNT is able to assign a data service indicator (the only one) to the incoming call from SOPHO-S.
The same problem also applies for voice.
Hence the PNT can only support one voice service and one data

service at the same time.

No sub-addressing
The s-interface of the PNT cannot support sub-addressing because sub-address information is not sent by SOPHO-S. Consequently all voice (data) terminals connected to the PNT will ring in case of an incoming voice (data) call. A calling party cannot address one specific ISDN terminal connected to the PNT.

Apart from the restrictions described above, the s-interface of the PNT offers the same functionality as the s-interface of a public ISDN network.

An ISDN terminal connected to the PNT offers the same 'basic facilities' as an ISDN terminal connected to a public ISDN network. The three main basic facilities are:

- 'display calling number'
  Information concerning the number of the calling party is sent by SOPHO-S to the PNT. The PNT will use this information to fill in the 'originating address' field of the SETUP message.

- 'call waiting indication'
  When SOPHO-S sends a call waiting indication message, the PNT will send a SETUP message with parameter 'both B-channels occupied'.

- 'metering result'
  SOPHO-S metering information will be placed in an INFO message that will be sent to the terminal by the PNT. However metering information is often sent by SOPHO-S after call completion. In this case the PNT cannot use the metering information, because the call process is already ended.

4.4.1 B-CHANNEL SELECTION PROCEDURE

The PNT B-channel selection procedures will be conform the ISDN Recommendations [Q.931]. In order to set up a call, an ISDN terminal sends a setup message to the PNT. In this setup message, the terminal indicates one of the following:

a) channel is indicated, no acceptable alternative;
b) channel is indicated, any alternative is acceptable;
c) any channel is acceptable.

If no indication is included, alternative c) is assumed.
In cases a) and b), if the indicated channel is available, the PNT selects it for the call.

Note:
In this section, the term 'available channel' means:

1. A free channel;
2. The type (data/voice) of the channel corresponds with the service indicator, which is indicated in the setup message.

In case b), if the PNT cannot grant the preferred channel, it selects the other available B-channel associated with the D-channel.

In case c), the PNT selects any available B-channel associated with the D-channel.

The selected B-channel is indicated in the first message returned by the PNT in response to the SETUP message (i.e. a SETUP ACKNOWLEDGE message).

4.4.2 THE USE OF CALL REFERENCE

The purpose of the call reference is to identify the call at the local terminal-PNT interface to which the particular message applies. Call reference is a matter between PNT and the connected terminals and not between PNT and SOPHO-S. The call reference is the second part of **every message**.

The PNT will maintain a call reference administration conform the ISDN Recommendations [Q.931].

The call reference information element includes the call reference value and the call reference flag. Call reference values are assigned by the originating side of the interface for a call. These values are unique to the originating side only within a particular D-channel layer 2 logical link connection. The call reference value is assigned at the beginning of a call and remains fixed for the lifetime of a call (except in the case of call suspension). After a call ends, or, after a successful suspension, the associated call reference value may be reassigned to a later call. Two identical call reference values on the same D-channel layer 2 logical link connection may be used when each value pertains to a call originated at opposite ends of the link.

The call reference flag can take the values "0" or "1". The call reference flag is used to identify which end of the layer two logical link originated a call reference. The originating side always sets the call reference flag to "0". The destination side always sets the call reference flag to a "1". Hence the call reference flag identifies who allocated the call reference value for this call and the only purpose of the call reference flag is to resolve simultaneous attempts to allocate the same call reference value.

4.5 SIMULATION OF THE PNT SPECIFICATION

One of the goals of using SDL, was the use of the available tools to validate and verify the PNT layer 3 specification. The analyzing and conversion tools, together with an optional simulator generator are organized in the SAC module on the VAX system. Therefore the PNT layer 3 processes, specified using SGE/PC, are uploaded to the VAX. These SDL descriptions of the 4 PNT layer 3 processes can be found in appendix A.
The PNT layer 3 specification is used as input for the SAC tool. This tool performs several tasks in the following order:

1. **Conversion**; the GR diagrams are converted to a PR description file.

2. **Syntax-check**; a syntactical check is performed. When the syntax analyzer detects an error, an error message is generated and added to the error file.

3. **Analyze**; When no errors are detected during the syntactical check, a semantical check is performed.

4. **Generate-simulator**(optional); when the system is semantically correct, a simulator is generated of the SDL description.

4.5. SIMULATION OF THE PNT SPECIFICATION

Why is it attractive to have a simulator of the specification? First of all because at present it is the only tool available to validate and verify the dynamical behaviour of a system. The simulator can be used to validate a system specification against the requirement specification before an implementation is made.

4.5.1 THE USE OF THE SIMULATOR

The PNT layer 3 specification is validated against its environment. A *monitor* can be used to simulate the behaviors of the connected terminals and SOPHO-S. A monitor is a text file in which simulator commands are listed. The monitor contains e.g. signals that terminal(s) or SOPHO-S might send to the PNT.
In section 4.6 the concepts of SDL '88 that are not supported by the simulator generator at the time this report was written are summarized. Despite the many restrictions, only the following are examined more closely:

- viewed/revealed variables are not allowed;
- imported/exported variables are not allowed;
- only procedures without internal states are allowed;
- continuous signals or enabling conditions are not allowed.

<u>Viewed/revealed and imported/exported variables</u>
Sometimes it is necessary for processes to inspect values of other processes. For example, in case of the PNT layer 3, it would be convenient when a PCP process could inspect the sopho_mess_distr process whether a particular B-channel is free by using the imported/exported variable concept of SDL. Also, normally a network has several call control processes and one general process which controls the free/busy administration of switch resources.
It would be convenient to be able to use the viewed/revealed and imported/exported concepts for the simulation. Instead, the PNT layer 3 model uses, in the case of the free/busy b-channel administration, the extra signals *chan_request*, *chan_change* and *rel_chan* to solve the lack of imported/exported variables.
In other cases, the PNT layer 3 has to use variables that are set and changed during the simulation.

Procedures with internal states

In the PNT layer 3 model, the procedure DET_TERM (see appendix A.) creates a PCP process and then starts a timer. The procedure should terminate when this timer expires. Therefore this procedure includes one internal state to wait for this timer to expire.
However, the concept of internal states within a procedure is not supported by the simulator generator. Therefore this DET_TERM procedure cannot be used for simulation.

Continuous signals and enabling conditions

When a setup message from an ISDN terminal arrives in the PNT, destination address digits included in this setup message must be sent to SOPHO-S. These digits are sent to SOPHO-S in digit messages by the PCP process in the dial state. It would be convenient to have a *digits_present* continuous signal indicating the number of present digits. This continuous signal could be used in the dial state in an enabling condition for sending digit messages to SOPHO-S. Instead in the PNT layer 3, an extra signal is used to solve the lack of continuous signal/enabling condition.

Despite the many restrictions of the simulator generator, a simulator of the PNT layer 3 specification has been generated.
A SDT simulator is built as shown in figure 4.8.

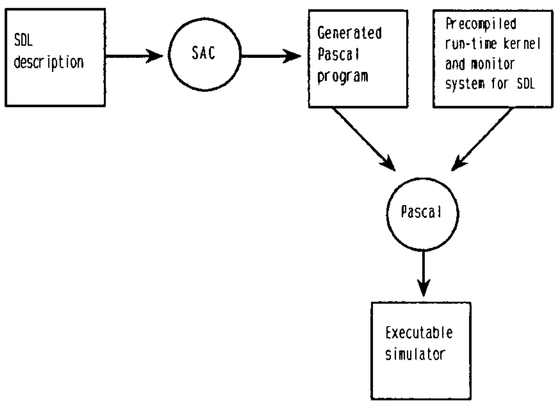


Figure 4.8 *"The production of an executable simulator."*

The precompiled kernel provides the user-interface and the monitor control system. A user interacts, using SDL terminology with the simulator through a user-interface, that has commands to:

- send signal instances from the environment to the system;
- examine the state of active processes;
- examine and add signals to and delete signals from the input queues of processes;
- examine and change the parameter values of signal instances;
- examine and change the values of variables of processes;
- examine the event queue (i.e. the process scheduling queue);
- trace the state transitions of processes;
- set and reset breakpoints;
- start and stop monitors.

4.5.2 A SIMULATION EXAMPLE

The simulator of the PNT layer 3 combines the use of monitors and processes to simulate calls. Terminals and SOPHO-S are simulated using monitors and the PNT layer 3 is simulated using processes. The user can send the following signals to the simulator system:

tmp_mess_from_sopho(<messagename>, <B-channel>)

isdn_mess_from_term(<messagename>, <call-reference>)

tmp_mess_from_term(<messagename>, <B-channel>)

Among the numerous situations that have been simulated, consider as an example the following sequence of monitor commands:

```
OUTPUT-TO TMP_MESS_FROM_SOPHO(INITPORT,0) SOPHO_MESS_DISTR,,
PROCEED-UNTIL 0.1
Comment: Initialization message for B0-channel.
```

OUTPUT-TO ISDN_MESS_FROM_TERM(SETUP,0) TERM_MESS_DISTR,,
PROCEED-UNTIL 0.2
Comment: ISDN terminal A tries to set up a call. However, this request should be refused because the PNT is not yet initialized.

OUTPUT-TO TMP_MESS_FROM_SOPHO(TERM_INIT,0) SOPHO_MESS_DISTR,,
PROCEED-UNTIL 0.3
Comment: This message terminates the initialization of the B0-channel.

OUTPUT-TO TMP_MESS_FROM_SOPHO(INC_CALL,0) SOPHO_MESS_DISTR,,
PROCEED-UNTIL 0.4
Comment: An incoming call arrives from SOPHO-S on the B0-channel.

OUTPUT-TO ISDN_MESS_FROM_TERM(ALERT,1) TERM_MESS_DISTR,,
PROCEED-UNTIL 0.5
Comment: ISDN terminal B reacts to this call.

OUTPUT-TO ISDN_MESS_FROM_TERM(CONN,1) TERM_MESS_DISTR,,
PROCEED-UNTIL 0.6
Comment: ISDN terminal B accepts the call. Consequently, the PNT should send a *answer* message to SOPHO-S.

OUTPUT-TO TMP_MESS_FROM_SOPHO(ANSW_ACK,0) SOPHO_MESS_DISTR,,
PROCEED-UNTIL 0.7
Comment: SOPHO-S acknowledges the answer message.

OUTPUT-TO ISDN_MESS_FROM_TERM(REL,1) TERM_MESS_DISTR,,
Comment: ISDN terminal B breaks off the connection.

Appendix B contains the simulator output. Inspection of this output shows that, in this case, the PNT operates adequate.

Besides the above example, a lot of other simulations have been carried out. An exhaustive simulation (=100% coverage) was of course not possible. However, the various simulations that have been carried out, have resulted in a much more consistent SDL description. Various wrong descriptions have been discovered by the simulations.

4.6 NOT SUPPORTED SDL CONCEPTS

In this section the concepts of SDL '88 that are not supported by SDT are described for SGE and the SAC module. Especially the simulator generator of the SAC module imposes restrictions on the SDL concepts that may used.

4.6.1 SGE, THE GRAPHICAL EDITOR

This section deals only with the PC version of SGE.

- The 'service' concept of SDL '88 is not supported;
- Only block and process *references* are permitted inside a block.

4.6.2 SAC, THE ANALYZER, CONVERTER AND SIMULATOR GENERATOR

The analyzer and converter do impose some minor restrictions in addition to those of SGE.

- No spaces in names;
- No generic systems or transition options;

However, the simulator generator imposes much more strict restrictions on the SDL concepts that have may be used. At the moment this report was written it was not clear which restrictions will remain and which will be solved in the future:

- No channel substructures;
- No procedures, which contain one or more states;
- Parameters to 'create' may not be omitted;
- No continuous signals or enabling conditions;
- No signal refinements;
- No view, reveal, import or export;
- No timer active expressions;

- No conditional expressions;

- No new operators may be introduced in sort definitions;

- Only the following user defined sorts are implemented:
  - syntypes of integer and character, which represent a sequence of consecutive values;

  - newtypes, where literals are enumerated (using the 'operators ordering' clause);

  - STRUCT containing only sorts implemented by SDT;

  - ARRAY with an index sort acceptable as index in a Pascal array and any sort implemented by SDT as component sort;

If there were no restrictions imposed by the simulator generator, then the service concept, internal states within procedures, continuous signals, enabling conditions and viewed, revealed, imported and exported variables would have been used in the PNT layer 3 specification.

4.7 SUPPLEMENTARY SERVICES

Both the 1TR6 protocol and the TMP protocol offer a variety of facilities. A detailed investigation into the problematic nature of facility conversion is subject for further study. However, by comparing the 1TR6 and TMP facilities, the PNT should be able to support the following supplementary services:

- User-to-user signalling;
- Enquiry;
- Do not disturb;
- Follow-me;
- Add on conference;
- Automatic ring back.

CHAPTER 5

IMPLEMENTATION OF THE PNT LAYER 3

As stated before in section 3.5, the PNT will consist of the same hardware as the PNT1 hardware. Moreover, PNT and PNT1 layers 1 and 2 sopho-side (u-side) and ISDN-side (s-side) will also be the same. See figure 5.1.

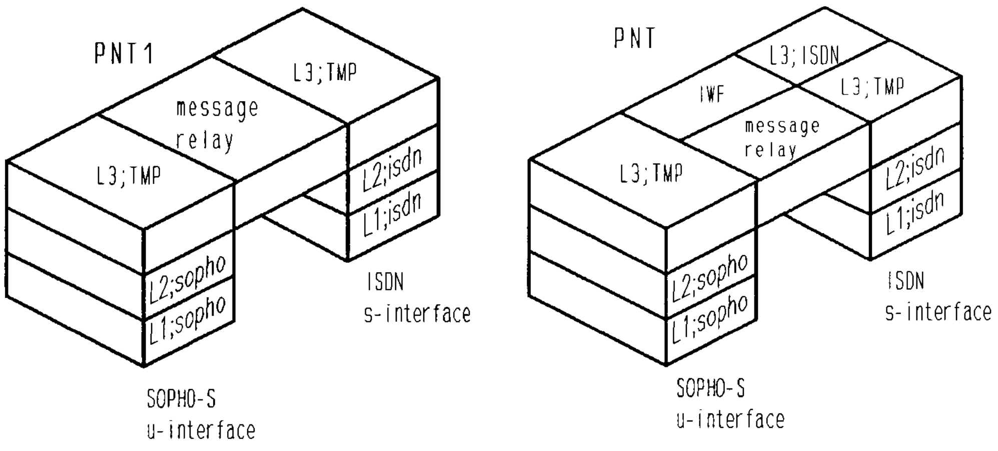


Figure 5.1 *"PNT1 and PNT OSI-models."*

Consequently, the software already written for the PNT1 layers 1,2 s-side and layers 1,2 u-side can also be used for the PNT!
In section 5.2 the layers 2 s- and u-side will be examined more closely, because the PNT layer 3 has to communicate with these layers. But first the PNT hardware will be considered.

5.1 THE PNT HARDWARE

Figure 5.2 illustrates the functional decomposition of the PNT hardware (=PNT1 hardware).

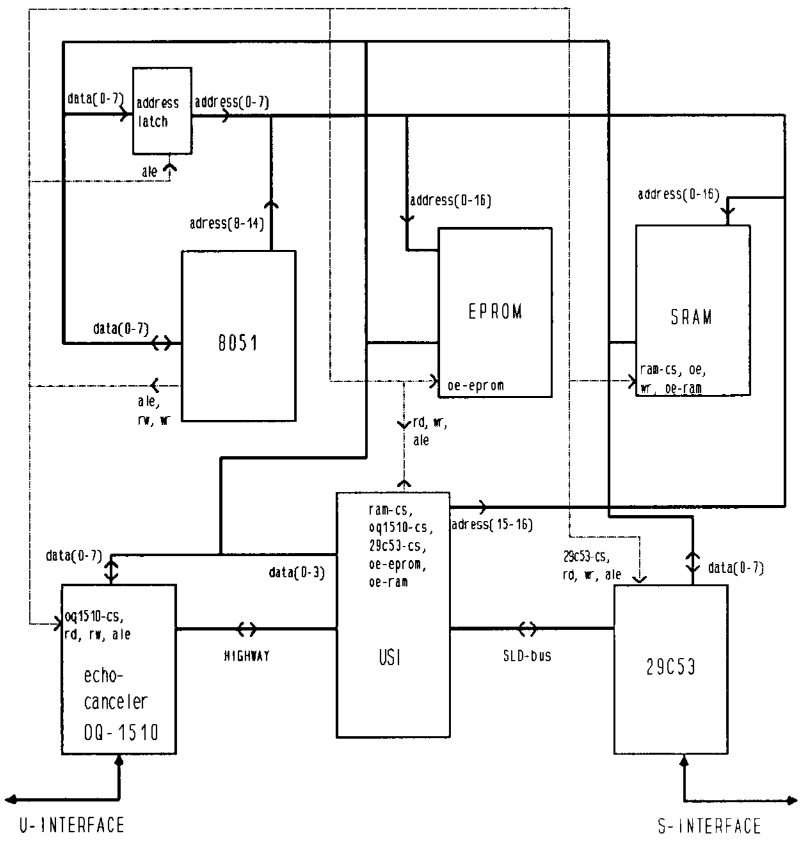


Figure 5.2 *"Functional decomposition of the PNT hardware."*

8051 μ-PROCESSOR

The PNT is equipped with the Intel 8051 single-chip microcomputer. This 40-pin microprocessor includes the following features:

- 4096 bytes program memory on-chip;
- 128 bytes data memory on-chip;
- Four register banks;
- 64 Kilobytes each program and external RAM addressability;
- One microsecond instruction cycle with 12 MHz crystal;
- 32 bidirectional I/O lines organized as four 8-bit ports;

- Multiple mode, high-speed programmable Serial Port;
- Two multiple mode, 16-bit Timer/Counters.

<u>S-INTERFACE CHIP</u>
The 29C53 Digital Loop Controller is an advanced, programmable digital transceiver providing the layer one interface at the s-interface in ISDN basic access applications. It provides access to the two B-channels and the D-channel in accordance with the CCITT Recommendation I.430 and supports both point-to-point and multipoint topologies.
The 29C53 is programmed for PNT purposes:

> The serial SLD-bus normally consists of B0-, B1- and D-channel frames. However within the PNT environment, the SLD-bus conveys only B0 and B1 frames. D-channel information frames are sent to/received from the μ-processor in order to be able to manipulate them.

<u>U-INTERFACE CHIP</u>
The OQ-1510 is a custom designed chip that performs functions analog to the s-interface chip. The chip defines an echo canceller based u-interface. The Highway-bus conveys B0- and B1-channel information. D-channel frames are sent to/received from the μ-processor in order to be able to manipulate them.

<u>USI-CHIP</u>
The U to S Interface chip (USI) is a custom designed gate array developed to perform the following functions:

- Bus-conversion; the B-channel information from the Highway-bus of the u-interface line circuit is transferred to the SLD-bus of the s-interface line circuit and vice versa.
- Memory control;
- Generation of chip select signals.

Considering the functions of the various components of the PNT hardware, it shows that the hardware bridges the difference in the transmission medium of the s-interface and that of the u-interface. The hardware also offers the possibility to manipulate the D-channel information. D-channels frames, retrieved from the interface chips are passed to the μ-processor so they can be processed by the PNT software. D-channel frames, generated by the PNT software, are directed to the interface chips.

5.2 LAYER 2-3 COMMUNICATION

It is necessary to examine the structure of the layers 2 u- and s-side in more detail because layer 3 of the PNT has to communicate with these layers.

5.2.1 PNT DATA LINK LAYER S-SIDE

The PNT data link layer 2-side is implemented conform the ISDN Recommendations [Q.921].

The data link layer consists of the Link Access Procedure on the D-channel, LAPD. The purpose of LAPD is to convey information between layer 3 entities across the ISDN user-network interface using the D-channel. So LAPD is a protocol that operates at the data link layer of the OSI architecture. Figure 5.3 illustrates the data link reference model.

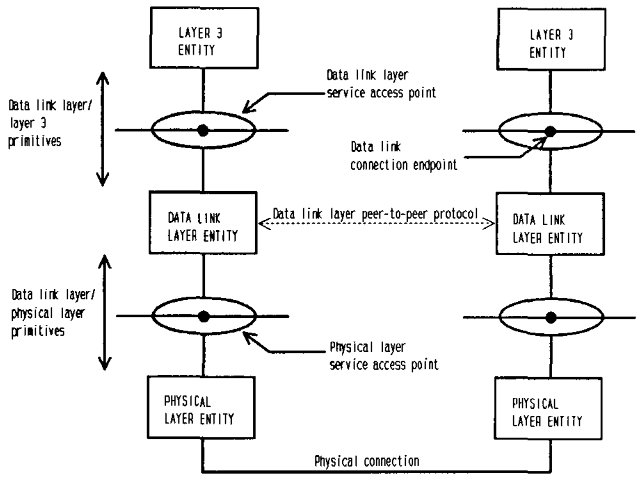

LAYER 3 ENTITY
LAYER 3 ENTITY
Data link layer service access point
Data link layer/ layer 3 primitives
Data link connection endpoint
DATA LINK LAYER ENTITY
Data link layer peer-to-peer protocol
DATA LINK LAYER ENTITY
Data link layer/ physical layer primitives
Physical layer service access point
PHYSICAL LAYER ENTITY
PHYSICAL LAYER ENTITY
Physical connection

Figure 5.3 *"Data link reference model."*

A data link layer Service Access Point (SAP) is the point at which the data link layer provides services to layer 3. The services provided by the data link layer are the combination of the services and functions by both the data link layer and the physical layer. Cooperation between data link layer entities is governed by a peer-to-peer protocol specific to that layer. In order for information to be exchanged between two or more layer 3 entities, an association must be established between the layer 3 entities in the data link layer using a data link layer protocol. This association is called a data link connection. Data link connections are provided by the data link layer between two or more SAPs. Data link layer message units are conveyed between data link layer entities by means of a physical connection.

<u>Services provided to layer 3:</u>
Two forms of information transfer services, provided by the data link layer, are associated with layer 3. The first is based on <u>un</u>acknowledged information transfer at the data link layer while the second service is based on acknowledged information transfer at the data link layer.

<u>Unacknowledged information transfer service:</u>
In this case, the information transfer is not acknowledged at the data link layer. The information transfer is via broadcast data link connections. The characteristics of the unacknowledged information transfer service are summarized in the following:

- Provision of a data link connection between layer 3 entities for unacknowledged information transfer of layer 3 message units;

- No verification of message arrival within the peer data link layer entity.

The only primitive associated with the unacknowledged information transfer service is:
DL-UNIT DATA-REQUEST <parameter = layer 3 message>.

The *DL-UNIT DATA-REQUEST* primitive will be used by the PNT layer 3 to send messages destined for **all** connected 1TR6 terminals (=broadcast information transfer service).

<u>Acknowledged information transfer service:</u>
There is one mode of operation defined: multiple frame. The characteristics of the acknowledged information transfer service are summarized in the following:

- Provision of a data link connection between layer 3 entities for acknowledged information transfer of layer 3 message units;
- Sequence integrity of data link layer message units in the absence of malfunctions;
- Notification to the peer entity in the case of errors, for example, loss of sequence.

The primitives associated with the acknowledged information transfer service are:

i) Data transfer
DL-DATA-REQUEST/INDICATION <parameter = layer 3 message>;

ii) Establishment of multiple frame operation
DL-ESTABLISH-REQUEST/INDICATION/CONFIRM;

iii) Termination of multiple frame operation
DL-RELEASE-REQUEST/INDICATION/CONFIRM;

The ability of the data link layer to execute a service request by layer 3 depends on the internal state of the data link layer. For the layer 3 entity, the internal state of the data link layer is represented by the state of that data link connection endpoint within a data link service access point which is used by this layer 3 entity to invoke a service.

<u>Data link layer representation as seen by layer 3:</u>
The states of a data link connection endpoint may be derived from the internal states of the data link layer entity supporting this type of a data link connection.

- The broadcast data link connection endpoint is always in the information transfer state;
- The point-to-point data link connection endpoint states are:

  - link connection released state;
  - awaiting establish state;
  - awaiting release state;
  - link connection established state.

5.2. LAYER 2-3 COMMUNICATION

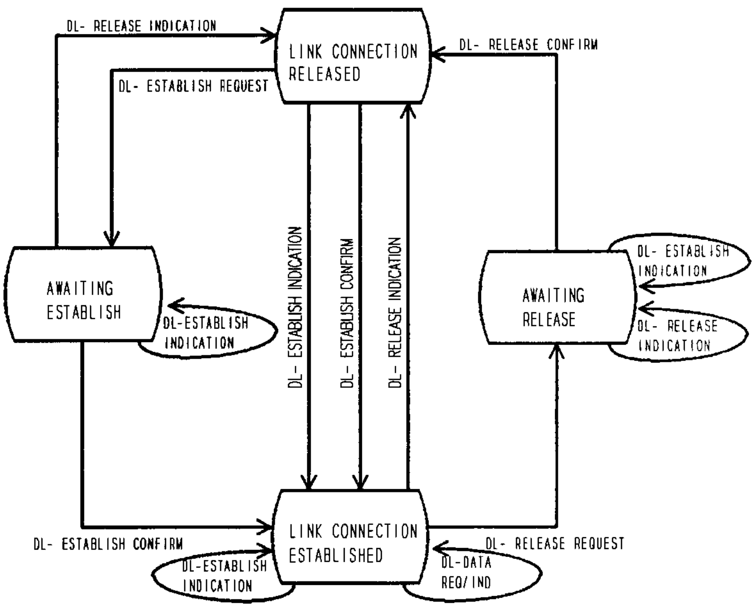


Figure 5.4 *"State transition diagram of a p-to-p data link conn."*

Primitives provide the procedural means to specify conceptually how a data link service user can invoke a service. The possible overall sequences of primitives at a point-to-point data link connection endpoint are defined in the state transition diagram, figure 5.4.
In fact figure 5.4 defines the layer 2-3 communication rules. For example:

- When the data link connection is in the LINK CONNECTION ESTABLISHED state, layer 3 can read a layer 3 message indicated in the primitive *DL-DATA-INDICATION* or layer 3 can put a layer 3 message into the primitive *DL-DATA-REQUEST*.
  On receiving an active *DL-DATA-REQUEST*, layer 2 will send the layer 3 message indicated in this primitive to its peer data link layer entity.

- After call completion, layer 3 ends the call process by activating the primitive *DL-RELEASE-REQUEST*. The data link connection (as seen by layer 3) will be in the AWAITING RELEASE state. On receiving the active *DL-RELEASE-REQUEST*, layer 2 will release the link connection and activates the primitive *DL-RELEASE-CONFIRM*.

On receiving this active *DL-RELEASE-CONFIRM*, layer 3 will regard the data link connection as being in the LINK CONNECTION RELEASED state.

- Etc.

The PNT layer 3 has to maintain an administration for each data link connection, in order to know in which state the data link connection is and which primitives are active! For this purpose the PNT will contain a so called link connection management entity, that deals with this administration.

<u>Note:</u>
For the PNT a maximum of 8 data link connection endpoints can be active simultaneously. Therefore the layer 2-3 primitives will be implemented eightfold. The number of 8 is sufficient for PNT purposes (the PNT can set up data link connections endpoints with 8 logical terminals) and should not be increased unnecessarily (to much impact on RAM).

5.2.2 PNT DATA LINK LAYER U-SIDE

The PNT data link layer is not implemented conform the CCITT data link layer Recommendations Q.921; this layer is a Philips implementation. No layer 2 data link connection states are defined. Therefore layer 3 can maintain a rather small data link layer administration consisting of only two primitives:

DL-DATA-REQUEST/INDICATION <parameter = layer 3 message>.

The reason that no data link connection states are defined, is merely the fact that there is a <u>fixed</u> data link connection between SOPHO-S and the PNT. This data link connection cannot be established nor released, it is always active. The only service provided by this data connection endpoint is the acknowledged information transfer service (so, no unacknowledged information transfer service). Layer 3 communication with layer 2 is restricted to:

- Reading layer 3 messages indicated in the *DL-DATA-INDICATION* primitive;

- Putting layer 3 messages into the *DL-DATA-REQUEST* primitive.

5.3 PL/M-51

The PNT software is written using the Intel PL/M-51 language. PL/M-51 is the Programming Language for the MCS-51 family of embedded controllers.
PL/M is a high-level language for programming various families of microprocessors and microcontrollers. It was designed by Intel Corporation to meet the software requirements of computers in a wide variety of systems and applications work. Following are some of the characteristics of PL/M:

- It has a block structure and control constructs that aid structured programming;
- It includes facilities such as data structures, structured arrays and pointer-based dynamic variables;
- It is a typed language, that is, the compiler performs data type compatibility checking to detect logic errors in programs at compilation time.

The two types of statements in PL/M are declarations and executable statements. A simple example of a declare statement is:

```
DECLARE   WIDTH   BYTE;
```

An example of an executable statement is:

```
CLEARANCE = WIDTH + 2;
```

PL/M is a block-structured language; every statement in a program is part of at least one block. A block is a well-defined group of statements beginning with a DO statement or a procedure declaration and ending with an END statement. The compilation unit in PL/M-51 is a module, which is a labeled simple DO block (The DO block is described as simple because it is just one of four DO-blocks).
A module can contain other blocks but is never itself contained

within another block.
Every PL/M program consists of one or more modules, separately compiled, each consisting of one or more blocks.

PL/M has two kinds of blocks: DO blocks and procedure definition blocks. A procedure definition block is a set of statements beginning with a procedure declaration and ending with an END statement. PL/M has four types of DO statements which are used to construct four kinds of DO blocks:

A simple DO block merely groups, as a unit, a set of statements to be executed sequentially (except for the effect of GOTOs or CALLs).

An iterative DO statement introduces an iterative DO block and causes the executable statements within the block to be executed repeatedly.

The DO WHILE statement contains a condition and causes the executable statements in the block to be executed repeatedly as long as the condition is true.

The DO CASE block, which is introduced by a DO CASE statement, uses the value of the given expression to select a statement to be executed.

In PL/M, a variable can be a scalar (a single quantity), an array, or a structure. A scalar variable or constant always has as type: **BYTE, WORD, or BIT** (so, no character-types, string-types, boolean-types, etc.).
An array is a list of scalars; a structure is an aggregate of scalars, arrays, or both.

A PL/M variable has a name, a type, and a memory-space. The 8051 μ-processor has more than one memory-space (five locations in all), so an address by itself is not enough to specify in PL/M-51 where a variable resides. The memory-space in which it resides must be declared by affixing a suffix to the declaration. The suffix can be any one of the following:

- MAIN refers to the directly addressable on-chip RAM;
- AUXILIARY refers to the off-chip RAM;

- REGISTER refers to the memory-mapped hardware registers;
- IDATA refers to the indirectly addressable on-chip RAM;
- CONSTANT refers to ROM.

PL/M is used to create blocks of code containing declarations followed by executable statements. These blocks are ordered and nested in such a way as to simplify and clarify the flow of data and control. A collection of these blocks performing a single function, or a small set of related functions, is usually compiled as one module.
Beyond the advantages of modularity, simplicity, and clarity, the nesting of blocks serves another very basic purpose: names declared at an outer level are known to all statements of all nested blocks as well.

<u>Extended scope</u>
To extend the scope means to make names available for use in modules other than the one where they are defined. For example, scope can be extended for variables, labels, or procedures. The following declaration defines a procedure:

```
SUMMER: PROCEDURE (A, B) BIT PUBLIC;
   DECLARE (A, B) BYTE;
   ...
   ...

END SUMMER;
```

Any other module may invoke SUMMER if it first declares:

```
SUMMER: PROCEDURE (A, B) BIT EXTERNAL;
   /* A,B can be any names */
   DECLARE (A, B) BYTE;
   /* but these names must match them */
   /* and each type must match its public definition */
END SUMMER;
```

A more detailed description of the PL/M concepts can be found in [PLM 88].

5.4 MAPPING THE SDL PROCESSES ON SOFTWARE STRUCTURES

The next step in the process of implementing the PNT layer 3 SDL specification, is the mapping of the 4 SDL processes on software structures. First, consider the use of data request/indication primitives by the 4 processes. Table 5.1 is derived from figure 4.5.

The protocol control process does not inspect any primitives directly, because process input is received via one of the distribution processes. Also the initialization & downloading process does not check any primitives directly, because process input is received via the "SOPHO-S messages distribution process".

PNT layer 3 processes:	primitives for process input:	primitives for process output:
SOPHO-S messages distribution process	dl-data-ind-u-side	dl-data-req-s-side
Terminal messages distribution process	dl-data-ind-s-side	dl-data-req-u-side
Protocol control process	---- (input via distribution processes)	dl-data-req-u-side dl-data-req-s-side dl-unit-data-req
Initialization & downloading process	---- (input via SOPHO-S mess. distr. proc.)	dl-data-req-u-side

Table 5.1 *"The use of primitives by the PNT layer 3 processes."*

Figure 5.5 illustrates the PNT layer 3 software module configuration together with the relating primitives.

The PCP and IDP processes are controlled by the two message distribution processes. Therefore a "Layer 3 control" module is created consisting of the implementations of the sopho_mess_distr process

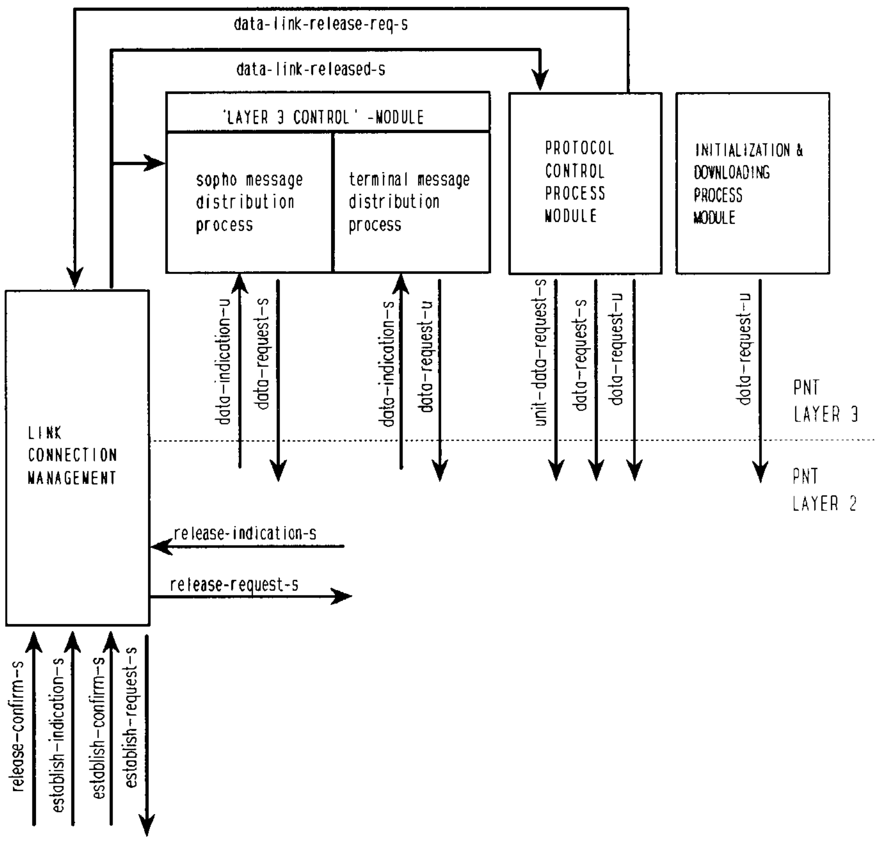


Figure 5.5 *"Layer 3 software modules with relating primitives."*

and the term_mess_distr process.

The PCP and IDP processes will be implemented in separate modules.

The translation of the various SDL descriptions into program code appeared to be a rather straightforward task. The SDL description offered a well defined framework for the software implementation. The translation did not introduce any new problems, because all the SDL entities could be easily replaced by software entities. The translation only involved practical matters that had to be solved, as a direct consequence of the use of informal SDL descriptions. For example:

- The SDL description uses the informal task 'choose free call reference'. The translator has to choose a data structure in order to be able to maintain a call reference administration.

- The SDL description of the PCP process uses the informal decision 'selection digits present?'. The translator has to implement a message decoding function, that must be capable of decomposing the setup message searching for possible present digits.

Except for this kind of practical matters, the translation of the SDL descriptions did not introduce any new problems. This experience is also supported by the knowledge that in the near future a new SDT tool, the SPG module, will become available. This SDL Program Generator will be capable of a automatic generation of a computer program of the SDL description. Planned languages to be supported are C and Ada.

In order to illustrate the ease of the translation, consider the following example:

Figure 5.6 shows a part of the SDL description of the PCP process, connected state.

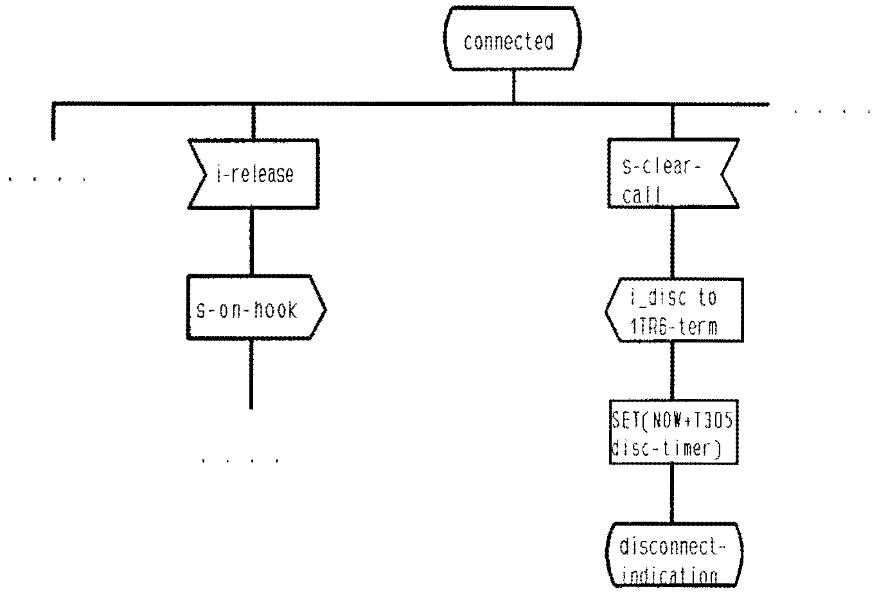


Figure 5.6 *"Part of the SDL description of the PCP process."*

The PL/M code in table 5.1 corresponds to the above SDL description.

```
connected_process: PROCEDURE;
DO;
   IF (pcp_action = tr6_message_present)
   THEN
   DO;
      IF (tr6_messagebyte = i_release)
      THEN
      DO;
         CALL send_on_hook_message;
         ...
         ...

   END;
   ELSE
   DO;
      IF (pcp_action = sopho_message_present)
      THEN
      DO;
         IF (sopho_messagebytel = s_clear_call)
         THEN
         DO;
            CALL send_disconnect_message;
            CALL start_t305;
            protocol_control_instance(pcp_
              instance_nbr).state = disconnect_indication;
         END;
         ELSE
                ...
                ...
                ...
```

Table 5.2 "*Part of the PL/M code of the PCP process.*"

By comparing figure 5.6 and the corresponding PL/M code, it shows that the translation of the SDL description into PL/M code is indeed a rather straightforward process.

5.4.1 PNT LAYER 3 SOFTWARE MODULES

THE PROTOCOL CONTROL PROCESS MODULE

Because several PCP processes can be active at the same time, PCP process instances are implemented as an ARRAY of STRUCTURES:

```
protocol_control_instance(pcp_instance_nbr) STRUCTURE
                (
                .active,
                .state,
                .start_state,
                .tei_val,
                .cr,
                .b_channel
                )
```

Variables relating to the PCP instances can be administered by using this type of data structure.

A separate procedure is implemented for every PCP process state. These state procedures use the variable *pcp_instance_nbr* as parameter, so that one and the same state procedure can be used for state processing by all pcp instances!

The two message distribution processes can use the following public procedures, defined within the PCP module:

- *Return_free_pcp_instance*
  (To create a pcp instance);

- *Return_free_call_reference*
  (This procedure returns a free call reference value);

- *Cr_tei_comb_present*
  (The term_mess_distr process uses this procedure to determine to which PCP instance the 1TR6 message has to be scheduled);

- *PCP_state_processing*
  (This procedure executes one PCP state transition).

THE INITIALIZATION & DOWNLOADING PROCESS MODULE

A maximum of 2 (2 B-channels) IDP process instances can be active. The following data structure is used to represent these instances:

```
init_downl_instance(idp_instance_nbr) STRUCTURE
                 (
                 .state,
                 .initport_received)
```

Again the IDP process states are implemented in separate procedures. There is only one public procedure defined in the IDP module: *idp_state_processing*. The sopho_mess_distr process can call this public procedure in order to force an idp process state transition.

LAYER 3 CONTROL MODULE

Besides a layer 3 initialize procedure, this module also contains the main loop procedure of the PNT layer 3, *ly3_control*. The 3 major functions of this procedure are:

- Handling of possible present 1TR6 messages; carried out by the procedure *sopho_mess_distr_process*.
- Handling of possible present sopho messages; carried out by the procedure *term_mess_distr_process*.
- Handling of the layer 3 timers.

The *ly3_control* procedure has to be placed in the, so called PNT *scheduler_entries_list*. This list consists of procedures that have to be carried out in an endless loop:

```
scheduler_entries_list(
                          .scan_main_power_status,
                          .activate timers,
                          .ly1_control_u,
                          .ly1_control_s,
                          .ly2_control_u,
                          .ly2_control_s,
                          .handle_dl_primitives,
                          .ly3_control)          <----  added entry
```

The public procedure *ly3_initializer* has to be added to the PNT *initialize_entries_list*.
It is beyond the scope of this report to treat the various software modules in more detail. Moreover, each procedure is provided with the necessary comment explaining the function of the procedure.

5.4.2 GENERAL REMARKS ON THE SOFTWARE

- Procedures with formal parameters are not used in the PNT software. Although the use of formal parameters leads to a better structured program, in big programs it also leads to an enormous impact on the stack! Therefore, in order to avoid stack overflows, the PNT software does not use procedures with formal parameters.

- The PNT software uses no CASE statements. Instead IF-THEN-ELSE constructs are used. The reason for not using CASE statements is the increased impact on RAM in comparison to the IF-THEN-ELSE construct.

- In sections 4.5 and 4.6 the SDL concepts, not supported by the SDT simulator generator were discussed. Because of these shortcomings, the PNT layer 3 SDL description had to be modified in order to generate a simulator. In particular the lack of viewed/revealed and imported/exported variables resulted in extra signals and process states (see section 4.5.1).
  PL/M procedures being in different modules can share common variables. **Therefore, during the** SDL-to-PL/M translation, the **introduced modifications have been deleted again!**
  For example, in the case of the free/busy b-channel administration, the necessarily introduced signals and process states are not translated. Instead the b-channel administration is implemented in a data structure, which can be directly manipulated by the different process procedures.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 USE OF SDL AS SPECIFICATION LANGUAGE

The experiences with the PNT layer 3 model have clearly shown that SDL is very well suited to be used as specification language. Compared to informal methods, SDL has the following advantages:

- Unambiguous, consistent specification of the behaviour of a system;

- Comprehensible documents when using the graphical representation;

- It may result in (semi)automatic methods to implement a new system;

- It has a low level of abstraction. Therefore, it is easy to learn;

- Validation and verification of a design is supported by the simulator.

However, there are also some drawbacks:

- Designers have to get used to formal methods;

- Due to the low level of abstraction, one has to be very careful not to implement a system instead of specifying it.

6.1.1 ABOUT THE SDL SIMULATOR

The process of validating and verifying a design is simplified by

the possibility to generate a simulator. Several mistakes in the PNT layer 3 specification were detected using the simulator. The use of a simulator has surely made the specification much more consistent.
However, the simulator generator has also some major drawbacks. The simulator generator imposes strict restrictions on the SDL concepts that have may be used. At the moment this report was written it was not clear which restrictions will remain and which will be solved in the future. Because of these restrictions, the PNT layer 3 specification had to be modified in order to be able to generate a simulator. Especially the lack of viewed/revealed and imported/exported variables, continuous signals and enabling conditions resulted in extra signals and process states.

6.2 FROM SDL SPECIFICATION TO PL/M IMPLEMENTATION

The implementation phase, i.e. the translation of the SDL description into PL/M code, appeared to be a rather straightforward process. The SDL description offered a well defined framework for the software implementation. The translation did not introduce any new problems, because the SDL structures could be easily replaced by software structures. No strange tricks had to be carried out to convert the SDL structures into PL/M code.
The main point of the PNT layer 3 development process lied with the SDL specification, and not with the SDL-to-PL/M translation. The translation only involved practical matters that had to be solved, as a direct consequence of the use of <u>informal</u> SDL descriptions.
The experience that the translation did not introduce any new problems, is also supported by the knowledge that in the near future the SDL Program Generator tool will become available.

6.3 ABOUT THE PNT LAYER 3

The subject of the project was to investigate the possibility of connecting ISDN terminals via a PNT to SOPHO-S. When the PNT layer 3 protocol conversion, as specified in appendix A is used, it is indeed possible to connect ISDN terminals to the PNT. Moreover, the PNT layer 3 specification even allows a combination of ISDN terminals and P-terminals to be connected to the PNT.

6.3. ABOUT THE PNT LAYER 3

The s-interface defined by the PNT is conform the ISDN Recommendations except for two matters. First, the PNT only supports one voice- and one data service at the same time. Next the PNT cannot support sub-addressing. On the other hand, the PNT does support basic facilities such as 'display calling number' and 'call waiting indication'.
Within the SOPHO-S environment, one may only connect a terminal to an extension line, which corresponds with the projection of that particular extension line. This fact together with the knowledge that SOPHO-S does not send projection information, caused the PNT initialization problem. The PNT layer 3 is provided with the necessary intelligence to solve this problem. The implemented dynamical solution is based on the assumption that only terminal-configurations, corresponding the projection, may be connected to the PNT's s-interface.

6.4 FOR FURTHER STUDY

Due to the limited space of time, it was not possible to test the software implementation (circa 3500 program lines) of the PNT layer 3. Further work, that has to be carried out, must include a thorough testing of this software.
At this moment, the PNT layer 3 specification/implementation includes only the basic call control. A comparison of the 1TR6- and SOPHO-S supplementary services resulted in the assumption, that the PNT should be able to support most of these facilities. However, further study is required to deal with the necessary protocol conversion.

LIST OF ABBREVIATIONS AND GLOSSARY

BCS	Business Communication Systems
CCITT	Comité Consultatif Internationale de Télégraphique et Téléphone
CCS	Calculus for Communicating Systems
CE	Connection Endpoint
DCT	Data Communication & Textswitching
DM	Data Modules
DPNSS	Digital Private Network Signalling System
EFSM	Extended Finite State Machine
FCS	Frame Check Sequence
FSM	Finite State Machine
GR	Graphical Representation of SDL
HDLC	High level Data Link Control
IDP	Initialization & Downloading Process
ISDN	Integrated Services Digital Network
ISO	International Standardization Organization
ISPBX	Integrated Services Private automatic Branch eXchange
IWF	InterWorking Function
LAM	Line Adapter Module
LAPD	Link Access Procedure on the D-channel
L2	Layer 2, Data Link Layer
MML	Man Machine interaction Language
NT	Network Termination
OSI	Open Systems Interconnection
PABX	Private Automatic Branch eXchange
PCP	Protocol Control Process
PL/M	Programming Language for the Intel MCS-51 family of embedded controllers
PNT	Private Network Termination
PNT1	PNT with layers 1 & 2 ISDN and layer 3 TMP
PR	Phrase Representation of SDL
PTDSN	Philips Telecommunicatie & Data Systemen Nederland
SAC	SDT Analyzer and Converter
SAPI	Service Access Point Identifier
SDL	Specification and Description Language
SDT	SDL Design Tool
SGE	SDT Graphical Editor

LIST OF ABBREVIATIONS AND GLOSSARY

SOPHO-S switches	Telephony switches of the SOPHOMATION line of Philips
SOPHOMATION	Synergetic Open PHilips Office autoMATION
SPG	SDT Program Generator
SRG	SDT Report Generator
TA	Terminal Adapter
TE	Terminal Equipment
TEI	Terminal Endpoint Identifier
TMP	Terminal Message Protocol
1TR6	German version of the ISDN recommendations

BIBLIOGRAPHY

[1TR6 90] Deutsche Bundespost, "1 TR 6, D-Kanal-Protokoll Schicht 2 und 3", Ausgabe 1.90, 1990.

[BELI 89] Belina F. and Hogrefe D., "The CCITT-Specification and Description Language SDL", Computer Networks and ISDN Systems, North-Holland, (16):311-341, 1989.

[CONF 89] Proceedings of the IFIP TC6/ICCC Joint Conference on "ISDN in Europe", The Hague, The Netherlands, 25-27 April, 1989. North-Holland, 1989.

[MIL 80] Milner R., "A Calculus of Communication Systems", Lecture Notes in Computer Science, vol. 92, Springer-Verlag, 1980.

[PLM 88] Intel, "PL/M-51 on Dos systems.", Intel Corporation, 1988.

[Q.921] CCITT, "Recommendations Q.921 - Digital subscriber signalling system NO. 1 Data Link Layer", Fascicle VI.10, Blue Book, 1988.

[Q.931] CCITT, "Recommendations Q.931 - ISDN user-network interface layer 3 specification for basic call control", Fascicle VI.8, Blue Book, 1988.

[SDT 89] TeleLOGIC, "SDT user's manual", Sweden, 1989.

[SSR 89] Saracco R., Smith J. and Reed R., "Telecommunications Systems Engineering using SDL", North-Holland, 1989.

[TANE 81] Tanenbaum A.S., "Computer networks", Prentice-Hall, 1981.

[TMP 90] Willemsen E.F.M., SDL specification of the Terminal Message Protocol, Philips BCS DCT DM 0.13.01.07.01, 1990.

APPENDIX A

SDL DESCRIPTION OF THE PNT LAYER 3

SYSTEM PNT   1 (3)

```
SIGNAL TMP_mess_from_sopho(name_type_1,chan_no),
TMP_mess_from_term(name_type_2),
ISDN_mess_from_term(name_type_3,call_reference),
digits_in_buffer;
```

```
NEWTYPE name_type_1
LITERALS
answ_ack,clear_call,datacheck,date_time,
download_class,download_dte_info,download_funkey,
download_ring_rhythm,download_ring_time,enable_status_request,
inc_call,initport,led_control,om_op_command,pos_cpc,
pts,send_digits,set_mode,status_monitoring,term_inc_call,
term_init,meter_result,displ_cw,cw_off,cng_nbr,displ_nbr;
OPERATORS
        ORDERING
ENDNEWTYPE name_type_1;
```

```
NEWTYPE name_type_2
LITERALS
answer,call_request,class_download_conf,
clear_call_and_reconnect,datacheck_result,digit,digit1,
dte_download_conf,enquiry,facility_status_request,
funkey_download_conf,inc_call_term,inc_disr,initport_conf_d,
initport_conf_v,om_op_message,on_hook,oper_conf,
pct_status_request,restart_dialling,ring_download_conf,
setmode_conf,stop_ring_download,link_up,port_absent,
download_request;
OPERATORS
        ORDERING
ENDNEWTYPE name_type_2;
```

```
NEWTYPE name_type_3
LITERALS
setup,alert,conn,conn_ack,
disc,rel,rel_ack,det,info,fac,fac_ack,fac_rej,
user_info,fac_canc,fac_reg,res,susp,fac_inf,fac_sta;
OPERATORS
        ORDERING
ENDNEWTYPE name_type_3;
```

```
NEWTYPE channel_number_type
LITERALS
B0,B1;
OPERATORS
ORDERING
ENDNEWTYPE channel_number_type;

NEWTYPE state_name
LITERALS
idle,call_present,connect_requested,
release_request,release_ack,
disconnect_indication,check_terminals;
OPERATORS
ORDERING
ENDNEWTYPE state_name;

SYNTYPE chan_no=integer
CONSTANTS 0:1
ENDSYNTYPE chan_no;

SYNTYPE call_reference=integer
CONSTANTS 0:100
ENDSYNTYPE call_reference;
NEWTYPE portnumber_type
ARRAY(chan_no,integer)
ENDNEWTYPE portnumber_type;

NEWTYPE IS_term_type
ARRAY(chan_no,boolean)
ENDNEWTYPE IS_term_type;

NEWTYPE pid_type
ARRAY(chan_no,pid)
ENDNEWTYPE pid_type;

NEWTYPE v_d_a_type
ARRAY(chan_no,v_d_a_sort)
ENDNEWTYPE v_d_a_type;

NEWTYPE flag_active_IDP_type
ARRAY(chan_no,boolean)
ENDNEWTYPE flag_active_IDP_type;

NEWTYPE cause_type
LITERALS
chan_occupied,chan_not_yet_initialized;
OPERATORS ORDERING
ENDNEWTYPE cause_type;

NEWTYPE v_d_a_sort
LITERALS
absent,voice,data;
OPERATORS
ORDERING
ENDNEWTYPE v_d_a_sort;
```

SYSTEM PNT   2 (3)

```
SIGNAL s_answ_ack,s_clear_call,s_datacheck,s_date_time,
s_download_class,s_download_dte_info,s_download_funkey,
s_download_ring_rhythm,s_download_ring_time,s_enable_status_request,
s_inc_call,s_initport,s_led_control,s_om_op_command,s_pos_cpc,
s_pts,s_send_digits,s_set_mode,s_status_monitoring,s_term_inc_call,
s_term_init,s_meter_result,s_displ_cw,s_cw_off,s_cng_nbr,s_displ_nbr,
s_answer,s_call_request,s_class_download_conf,s_clear_call_and_reconnect,
s_datacheck_result,s_digit,s_digitl,s_dte_download_conf,s_enquiry,
s_facility_status_request,s_funkey_download_conf,s_inc_call_term,
s_inc_disr,s_initport_conf_d,s_initport_conf_v,s_om_op_message,s_on_hook,
s_oper_conf,s_pct_status_request,s_restart_dialling,s_ring_download_conf,
s_setmode_conf,s_stop_ring_download,s_link_up,s_port_absent,
s_download_request,chan_request(chan_no),rel_chan(chan_no),
chan_change(chan_no,pid),B_chan_assigned,B_chan_not_assigned(cause_type),
idp_started(chan_no),idp_finished(chan_no);
```

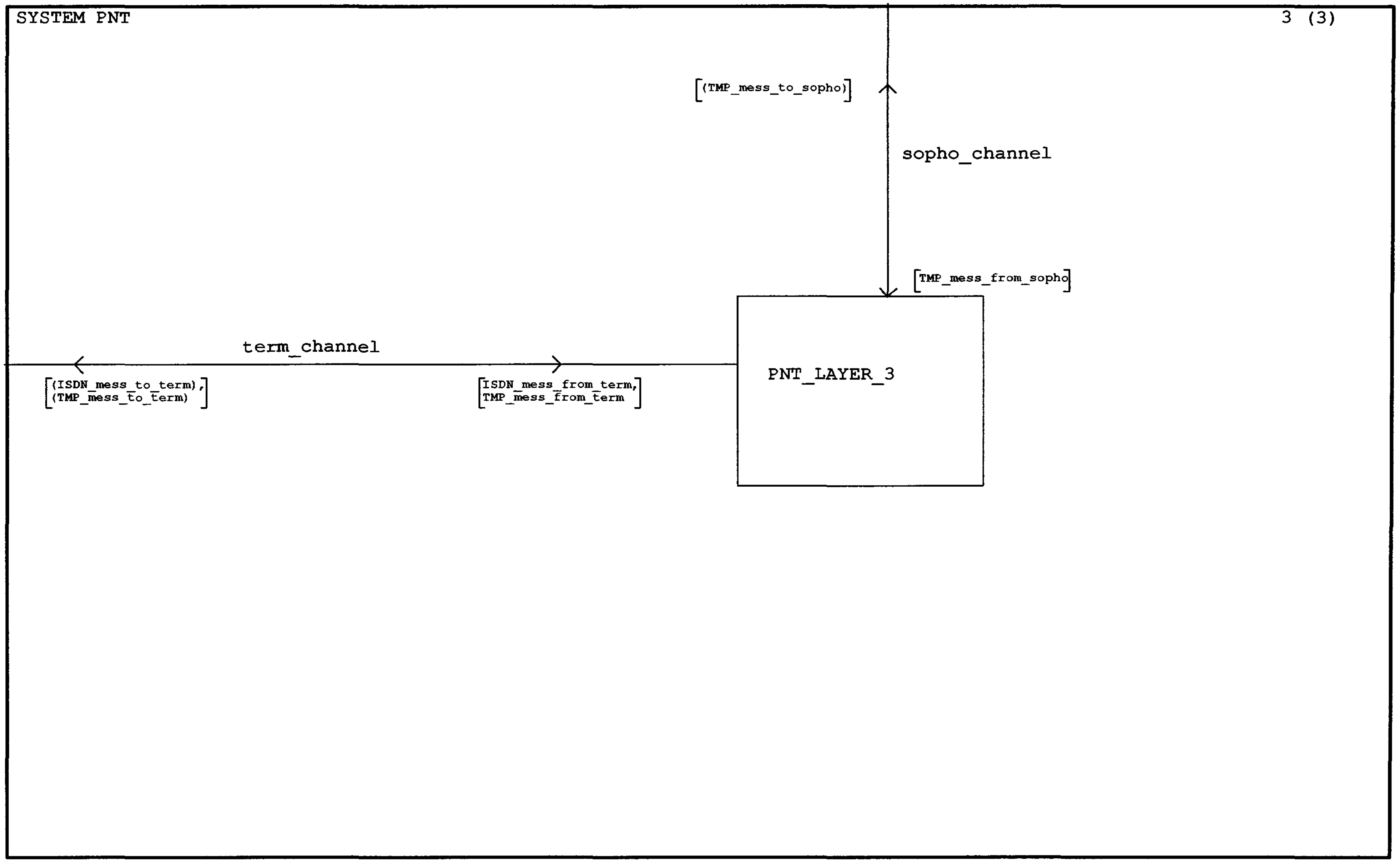
```
SIGNALLIST ISDN_mess_from_term=i_setup,i_alert,i_conn,i_conn_ack,
i_disc,i_rel,i_rel_ack,i_det,i_info,i_fac,i_fac_ack,i_fac_rej,
i_user_info,i_fac_canc,i_fac_reg,i_res,i_susp,i_fac_inf,i_fac_sta;
```

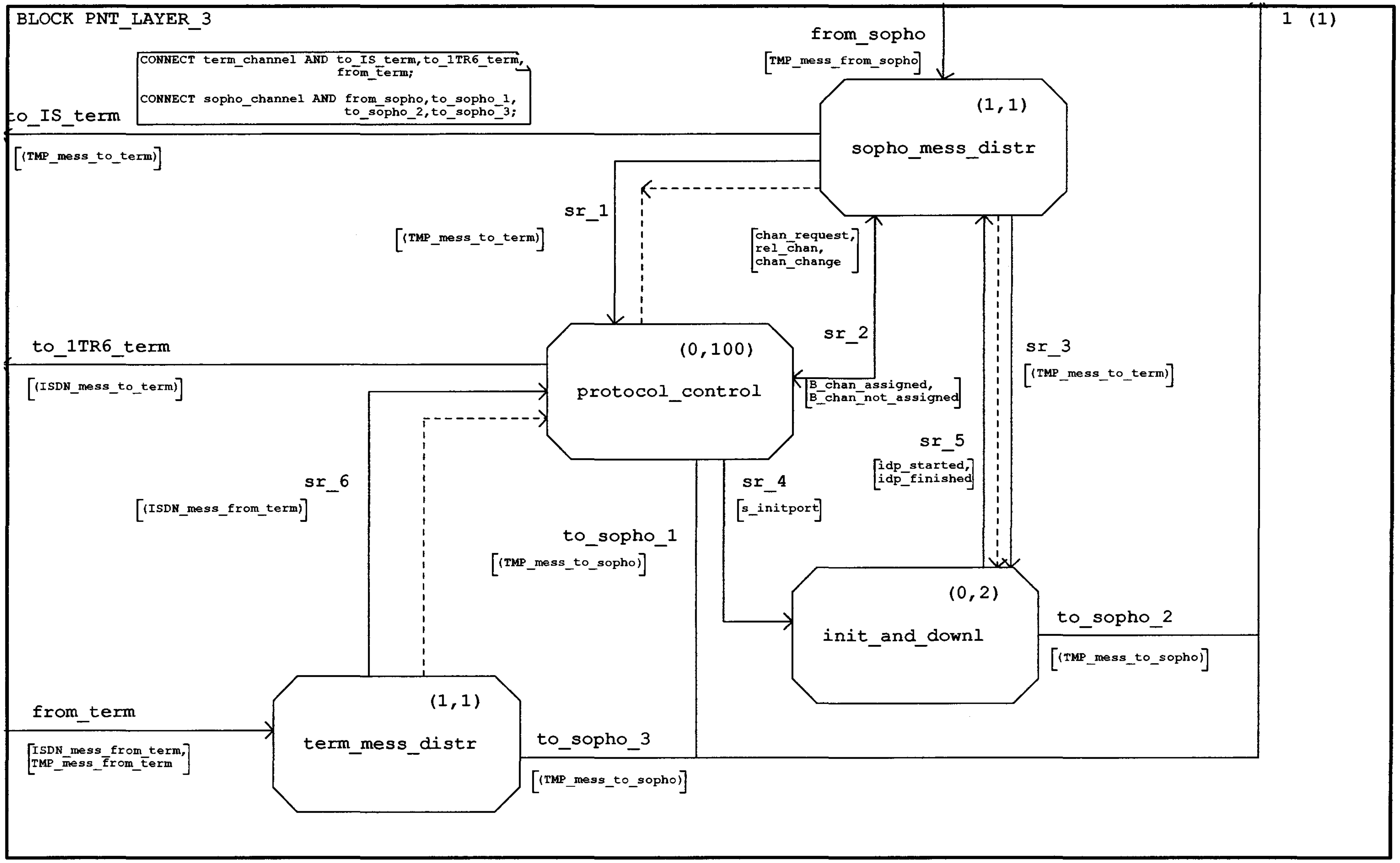
```
SIGNALLIST ISDN_mess_to_term=i_setup,i_alert,i_conn,i_conn_ack,
i_disc,i_rel,i_rel_ack,i_det,i_info,i_fac,i_fac_ack,i_fac_rej,
i_user_info,i_call_sent,i_canc_ack,i_canc_rej,i_con_con,i_reg_ack,
i_reg_rej,i_res_ack,i_res_rej,i_setup_ack,i_stat,i_susp_ack,
i_susp_rej,i_canc_ind,i_inf_ack,i_inf_rej,i_reg_ind,i_sta_ack,
i_sta_rej;
```

```
SIGNAL
i_setup,i_alert,i_conn,i_conn_ack,
i_disc,i_rel,i_rel_ack,i_det,i_info,i_fac,i_fac_ack,i_fac_rej,
i_user_info,i_call_sent,i_canc_ack,i_canc_rej,i_con_con,i_reg_ack,
i_reg_rej,i_res_ack,i_res_rej,i_setup_ack,i_stat,i_susp_ack,
i_susp_rej,i_canc_ind,i_inf_ack,i_inf_rej,i_reg_ind,i_sta_ack,
i_sta_rej,i_susp,i_res,i_fac_reg,i_fac_canc,i_fac_inf,i_fac_sta;
```

```
signallist TMP_mess_to_term=s_answ_ack,s_clear_call,s_datacheck,
s_date_time,s_download_class,s_download_dte_info,s_download_funkey,
s_download_ring_rhythm,s_download_ring_time,s_enable_status_request,
s_inc_call,s_initport,s_led_control,s_om_op_command,s_pos_cpc,
s_pts,s_send_digits,s_set_mode,s_status_monitoring,s_term_inc_call,
s_term_init,s_meter_result,s_displ_cw,s_cw_off,s_cng_nbr,s_displ_nbr;
```

```
SIGNALLIST TMP_mess_to_sopho=s_answer,s_call_request,
s_class_download_conf,s_clear_call_and_reconnect,s_datacheck_result,
s_digit,s_digitl,s_dte_download_conf,s_enquiry,s_facility_status_request,
s_funkey_download_conf,s_inc_call_term,s_inc_disr,s_initport_conf_d,
s_initport_conf_v,s_om_op_message,s_on_hook,s_oper_conf,
s_pct_status_request,s_restart_dialling,s_ring_download_conf,
s_setmode_conf,s_stop_ring_download,s_link_up,s_port_absent,
s_download_request;
```


SYSTEM PNT
3 (3)
[(TMP_mess_to_sopho)]
sopho_channel
[TMP_mess_from_sopho]
term_channel
PNT_LAYER_3
[(ISDN_mess_to_term),
(TMP_mess_to_term)]
[ISDN_mess_from_term,
TMP_mess_from_term]


BLOCK PNT_LAYER_3
1 (1)
CONNECT term_channel AND to_IS_term,to_1TR6_term, from_term;
CONNECT sopho_channel AND from_sopho,to_sopho_1, to_sopho_2,to_sopho_3;
from_sopho
[TMP_mess_from_sopho]
(1,1)
sopho_mess_distr
to_IS_term
[(TMP_mess_to_term)]
sr_1
[(TMP_mess_to_term)]
[chan_request, rel_chan, chan_change]
sr_2
[B_chan_assigned, B_chan_not_assigned]
sr_3
[(TMP_mess_to_term)]
(0,100)
protocol_control
to_1TR6_term
[(ISDN_mess_to_term)]
sr_5
[idp_started, idp_finished]
sr_4
[s_initport]
sr_6
[(ISDN_mess_from_term)]
to_sopho_1
[(TMP_mess_to_sopho)]
(0,2)
init_and_downl
to_sopho_2
[(TMP_mess_to_sopho)]
from_term
[ISDN_mess_from_term, TMP_mess_from_term]
(1,1)
term_mess_distr
to_sopho_3
[(TMP_mess_to_sopho)]

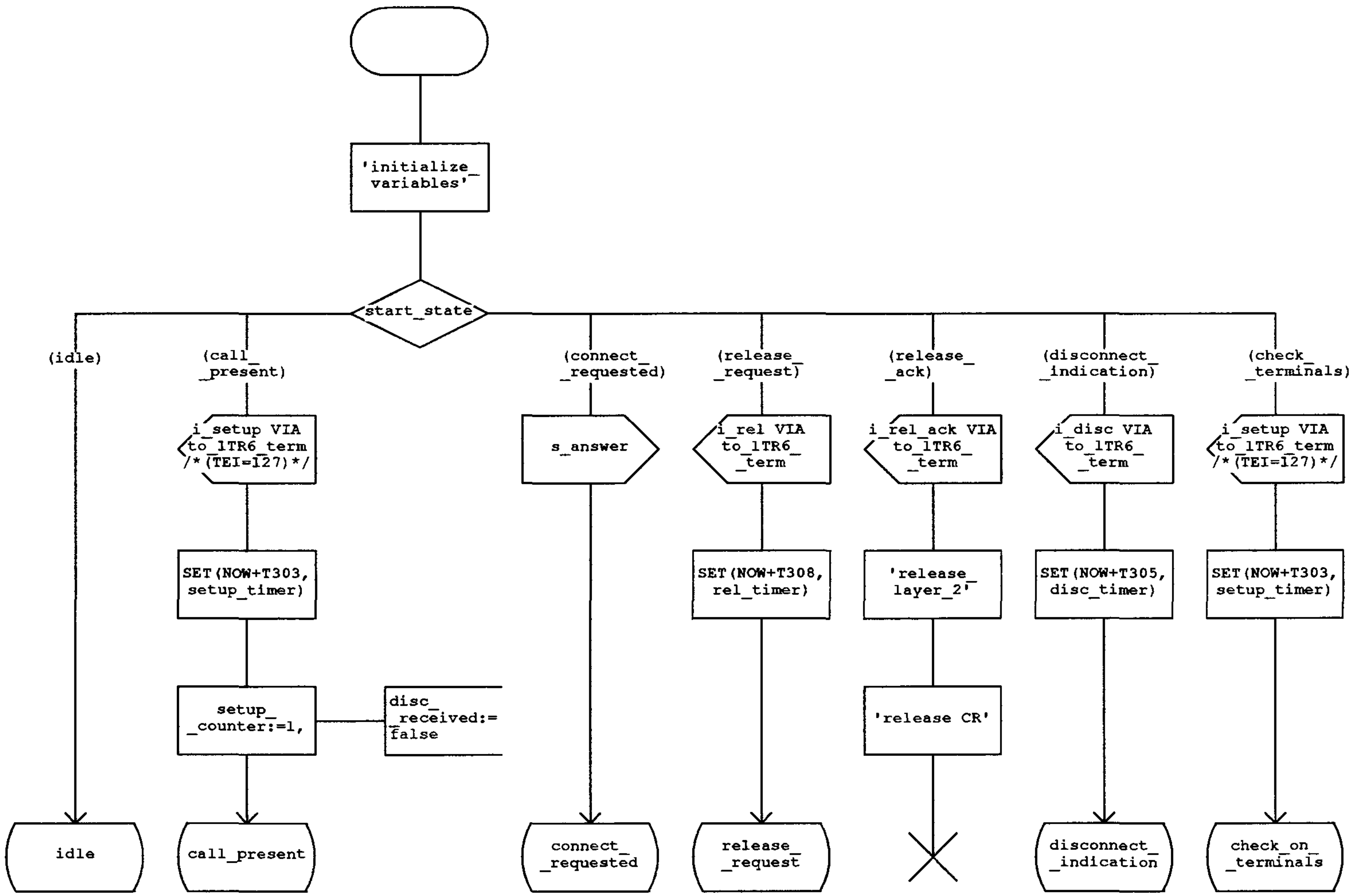
PROCESS protocol_control   declarations (23)

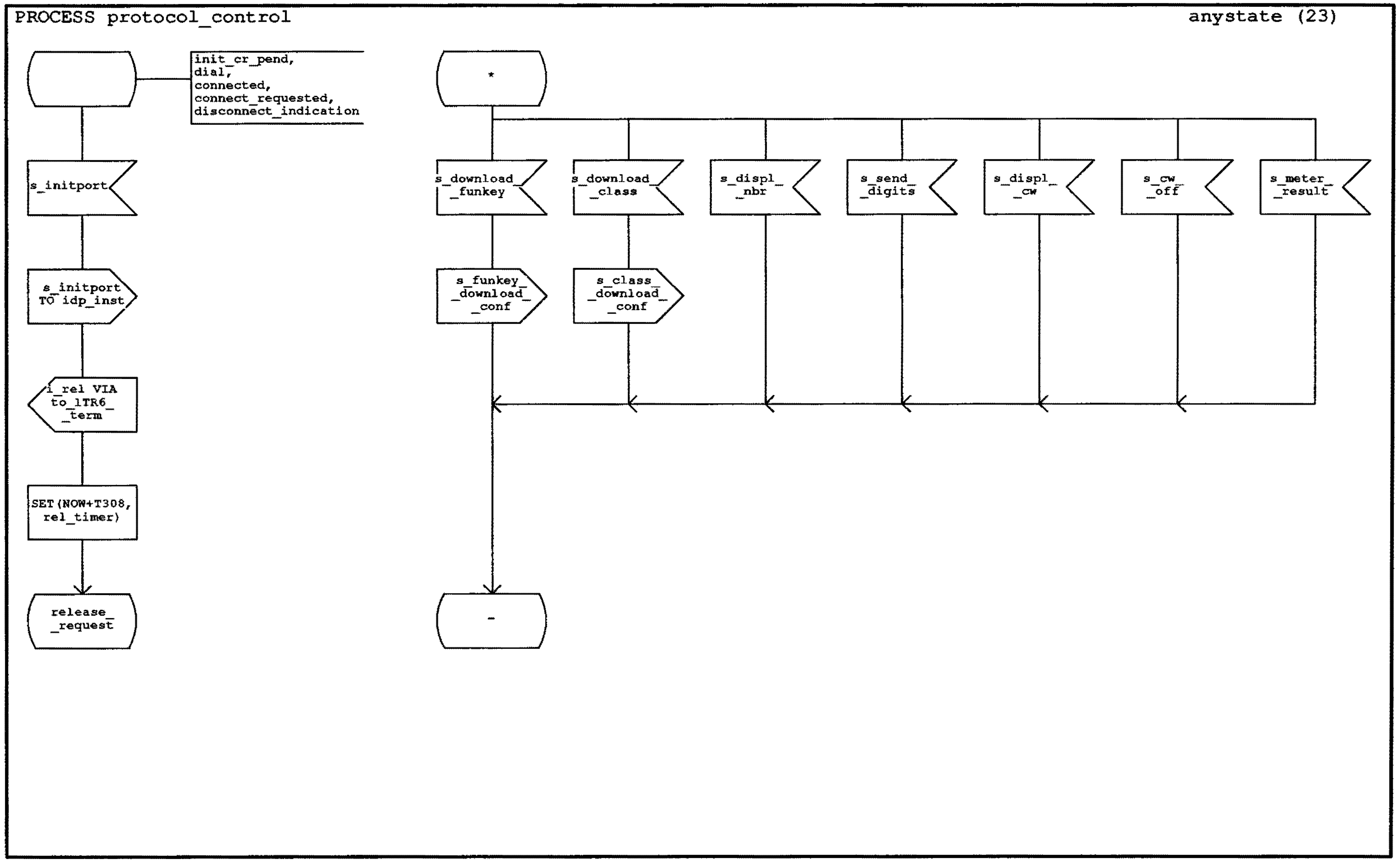
```
(0,100);
FPAR
CR,TEI integer,
start_state  state_name;
SIGNALSET
(TMP_mess_to_term),
(ISDN_mess_from_term),
B_chan_assigned,
B_chan_not_assigned,
digits_in_buffer;
```

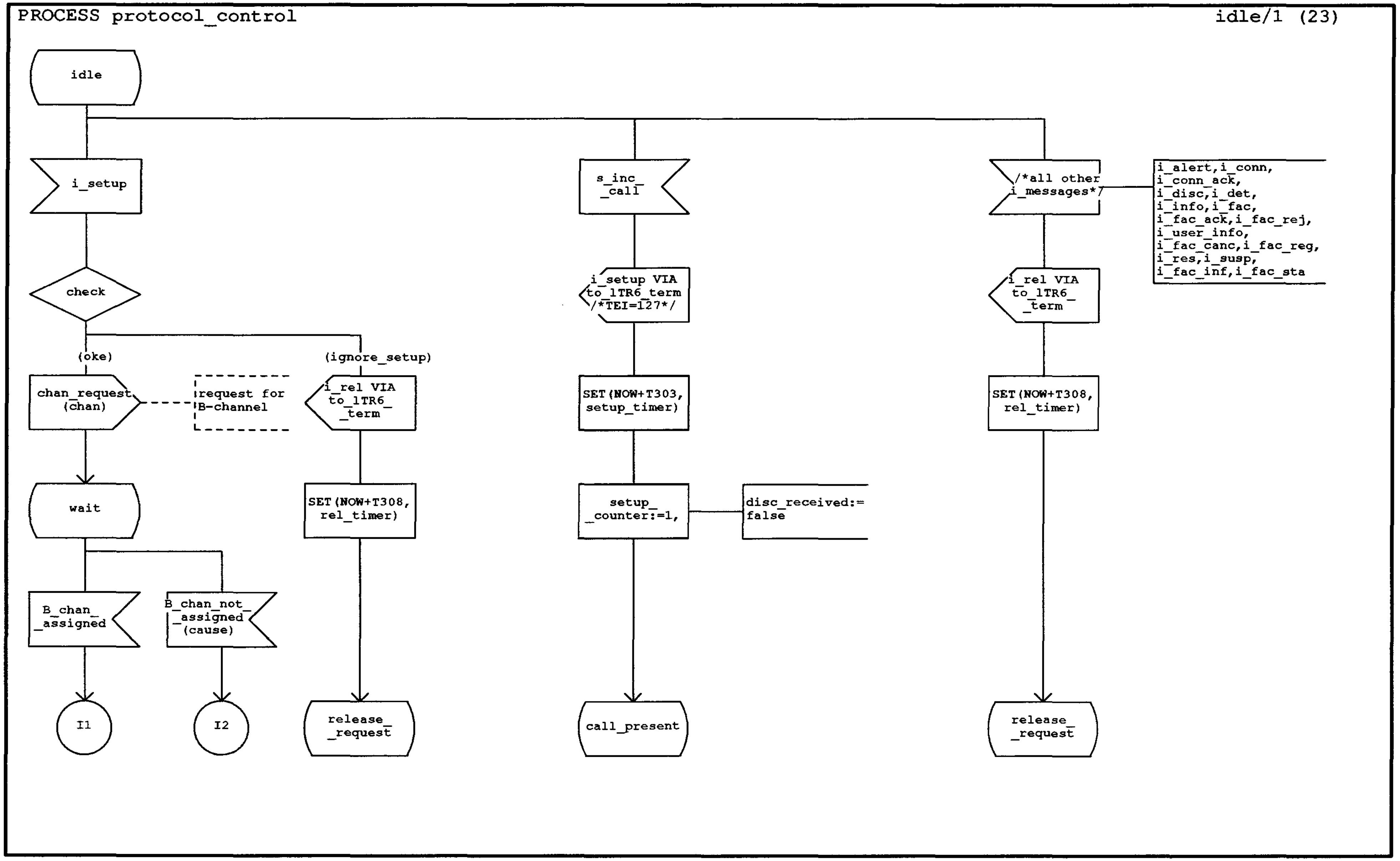
```
NEWTYPE check_type
LITERALS
oke,ignore_setup;
OPERATORS ORDERING
ENDNEWTYPE check_type;

NEWTYPE type_of_call_request
LITERALS
data_call,voice_call;
OPERATORS ORDERING
ENDNEWTYPE type_of_call_request;

NEWTYPE reason_for_clearing_type
LITERALS
a,b;
OPERATORS ORDERING
ENDNEWTYPE reason_for_clearing_type;

NEWTYPE cpc_type
LITERALS
        om_session_started,connect_with_party_added_on,
        connect_with_held_party,network_reached,
        connect_without_held_party,connect_when_free,
        redirected_call,terminal_called,enquiry_nack,
        dialling_ack,facility_registered,
        party_cleared;
OPERATORS
        ORDERING
ENDNEWTYPE cpc_type;
```

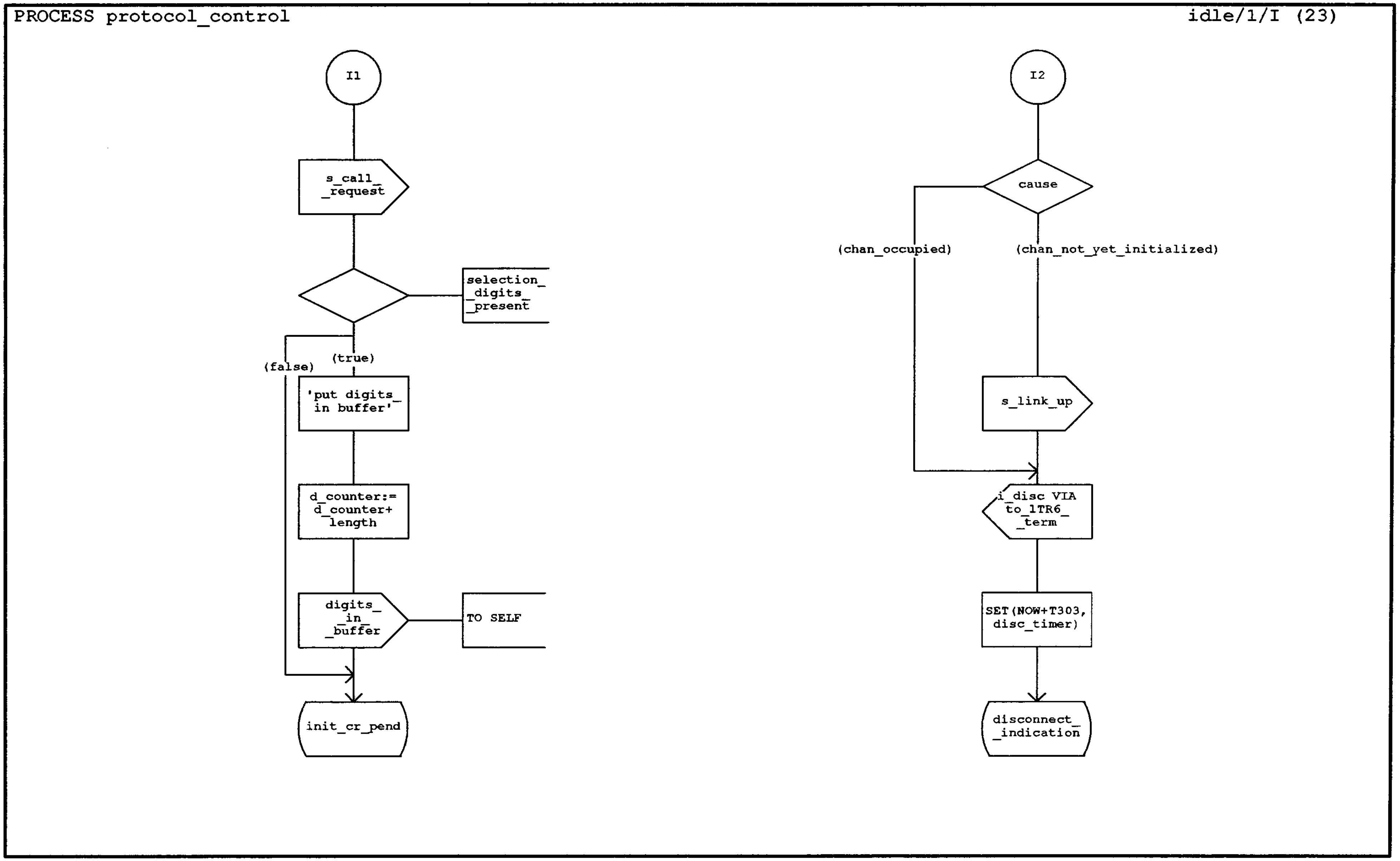
```
SYNONYM T305                 duration =120;
SYNONYM T308                 duration =4;
SYNONYM T303                 duration =1.5;
SYNONYM Tnonx                duration =4;
DCL
setup_counter                integer :=0,
length                       integer :=2,
d_counter                    integer :=0,
disc_received                boolean :=false,
selection_digits_present     boolean :=true,
mode_nonxmode                boolean :=false,
signals_in_process_queue     boolean,
fac_accepted_and_B_chan_free boolean :=true,
other_CR_TEI_present         boolean :=false,
two_times                    boolean :=false,
all_serv_ind_checked         boolean :=false,
process_occupies_B_chan      boolean :=false,
active_nonxmode_timer        boolean :=false,
active_setup_timer           boolean :=false,
chan                         chan_no,
idp_inst                     pid,
check                        check_type,
call_request_is              type_of_call_request,
reason_for_clearing,
choice                       reason_for_clearing_type,
cpc_value                    cpc_type,
cause                        cause_type;

TIMER
rel_timer,disc_timer,setup_timer,nonxmode;
```

PROCESS protocol_control   init (23)

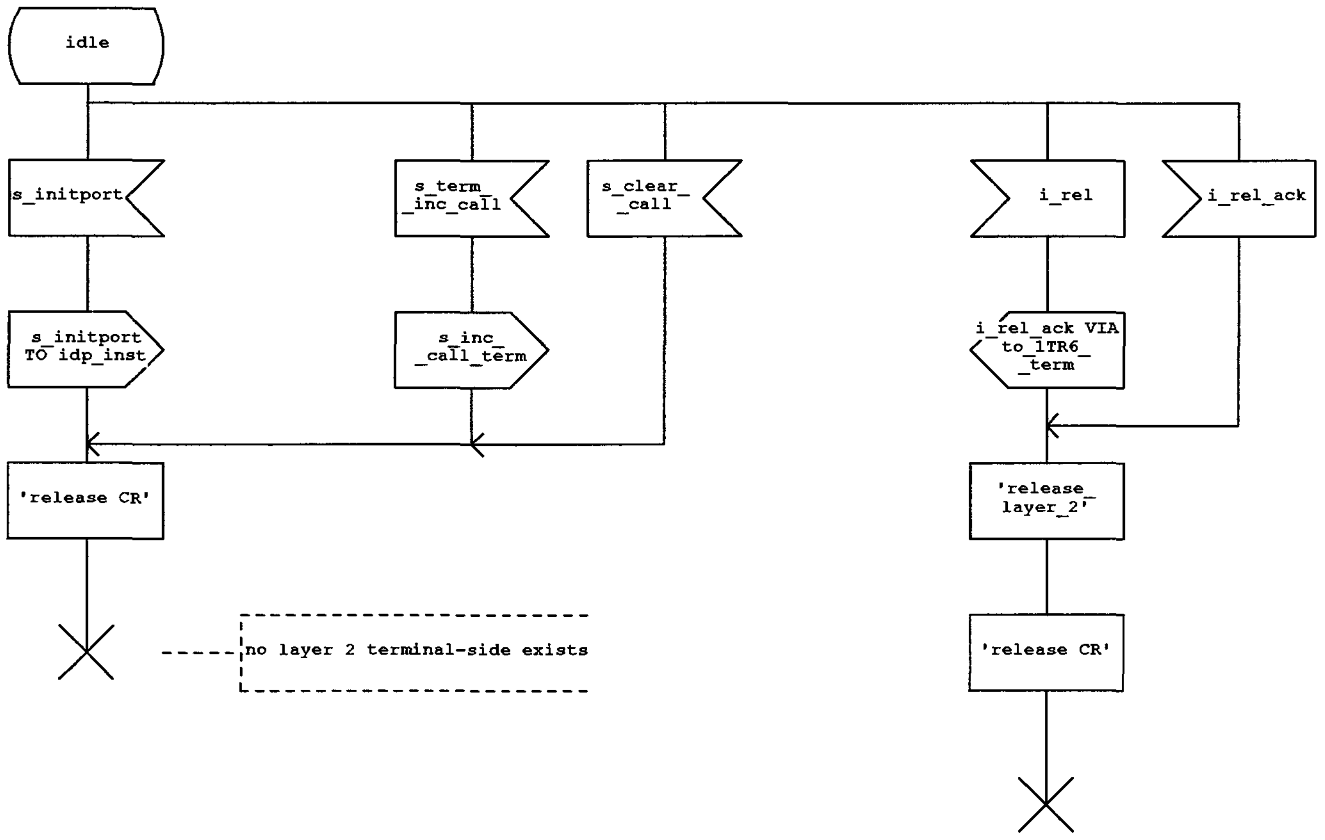


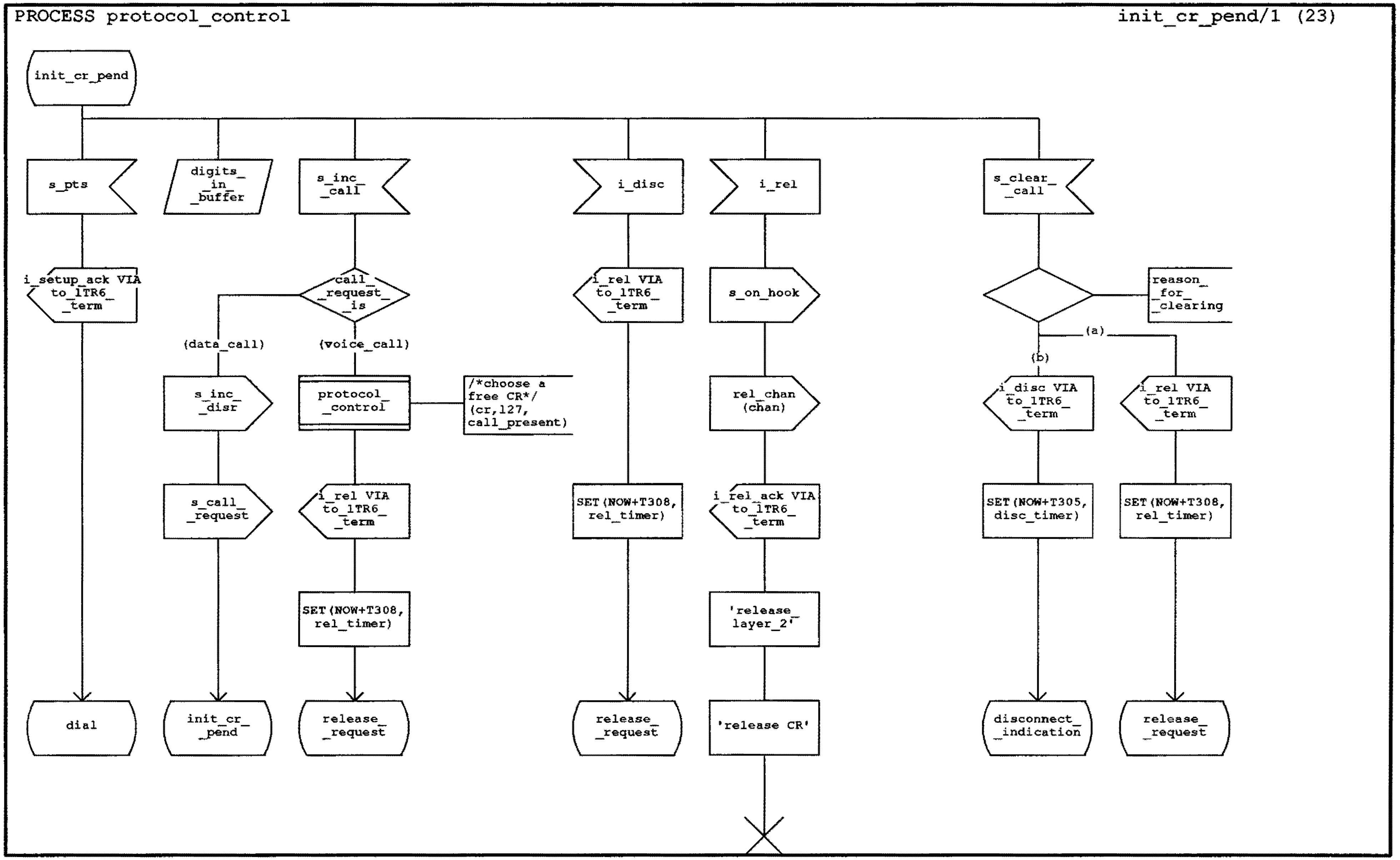

PROCESS protocol_control
anystate (23)
init_cr_pend,
dial,
connected,
connect_requested,
disconnect_indication
s_initport
s_initport TO idp_inst
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request
*
s_download_ _funkey
s_download_ _class
s_displ_ _nbr
s_send_ _digits
s_displ_ _cw
s_cw_ _off
s_meter_ _result
s_funkey_ _download_ _conf
s_class_ _download_ _conf
-


PROCESS protocol_control
idle/1 (23)
idle
i_setup
check
(oke)
(ignore_setup)
chan_request (chan)
request for B-channel
i_rel VIA to_1TR6_ _term
wait
SET(NOW+T308, rel_timer)
B_chan_ _assigned
B_chan_not_ _assigned (cause)
I1
I2
release_ _request
s_inc_ _call
i_setup VIA to_1TR6_term /*TEI=127*/
SET(NOW+T303, setup_timer)
setup_ _counter:=1,
disc_received:= false
call_present
/*all other i_messages*/
i_alert,i_conn, i_conn_ack, i_disc,i_det, i_info,i_fac, i_fac_ack,i_fac_rej, i_user_info, i_fac_canc,i_fac_reg, i_res,i_susp, i_fac_inf,i_fac_sta
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request

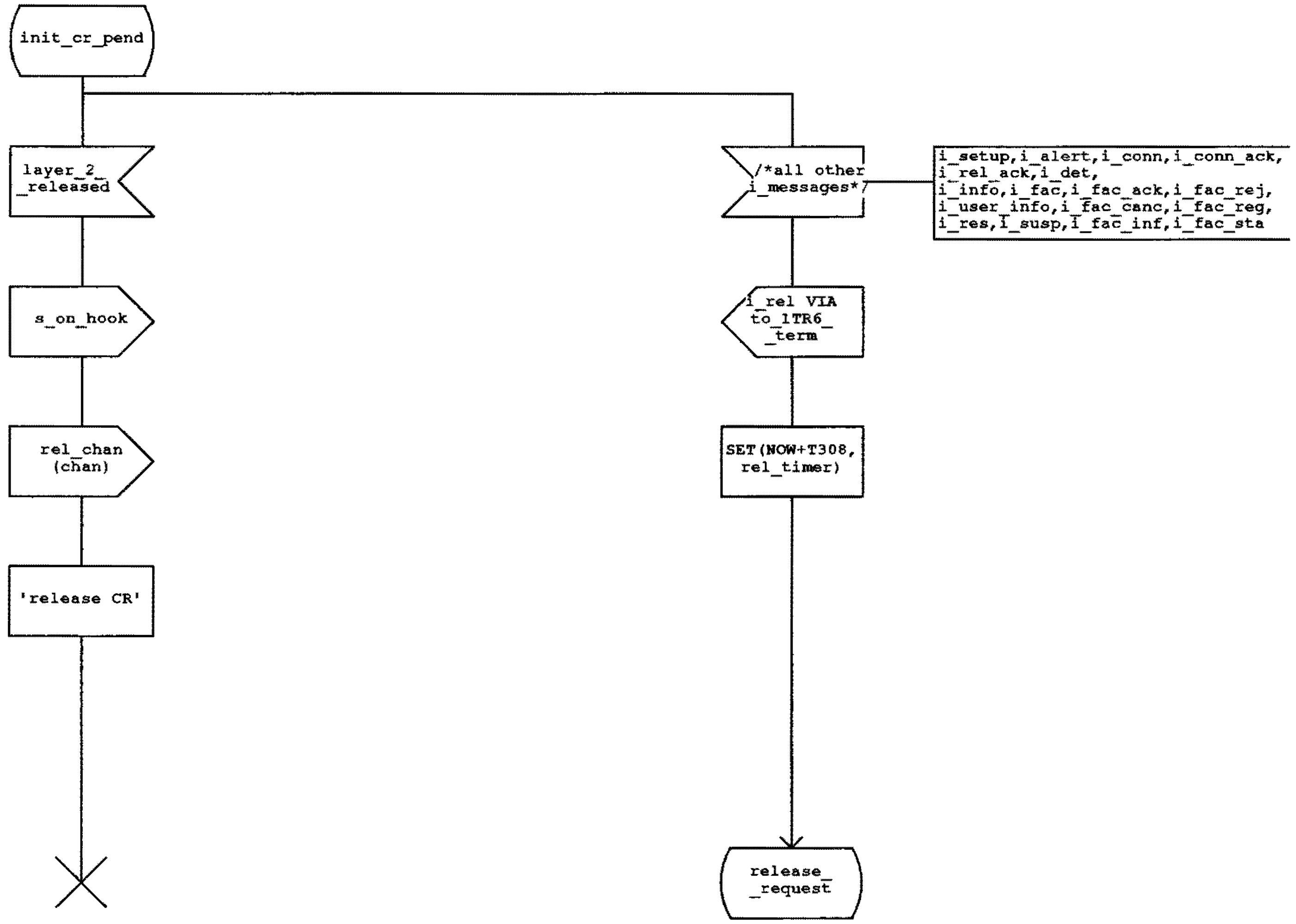

PROCESS protocol_control
idle/1/I (23)
I1
s_call_
_request
selection_
digits
_present
(false)
(true)
'put digits_
in buffer'
d_counter:=
d_counter+
length
digits_
_in_
_buffer
TO SELF
init_cr_pend
I2
cause
(chan_occupied)
(chan_not_yet_initialized)
s_link_up
i_disc VIA
to_1TR6_
_term
SET(NOW+T303,
disc_timer)
disconnect_
_indication

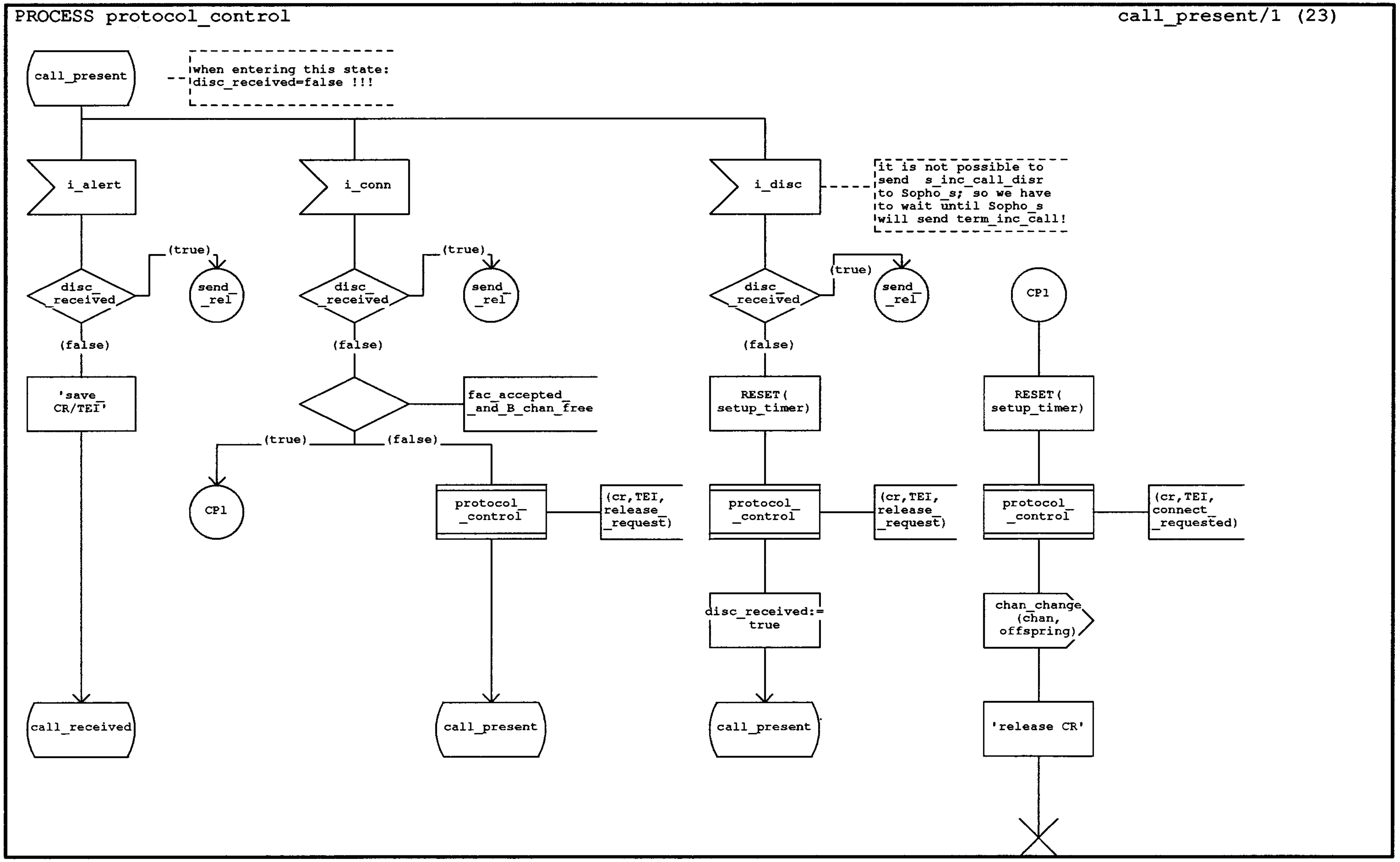
PROCESS protocol_control idle/2 (23)

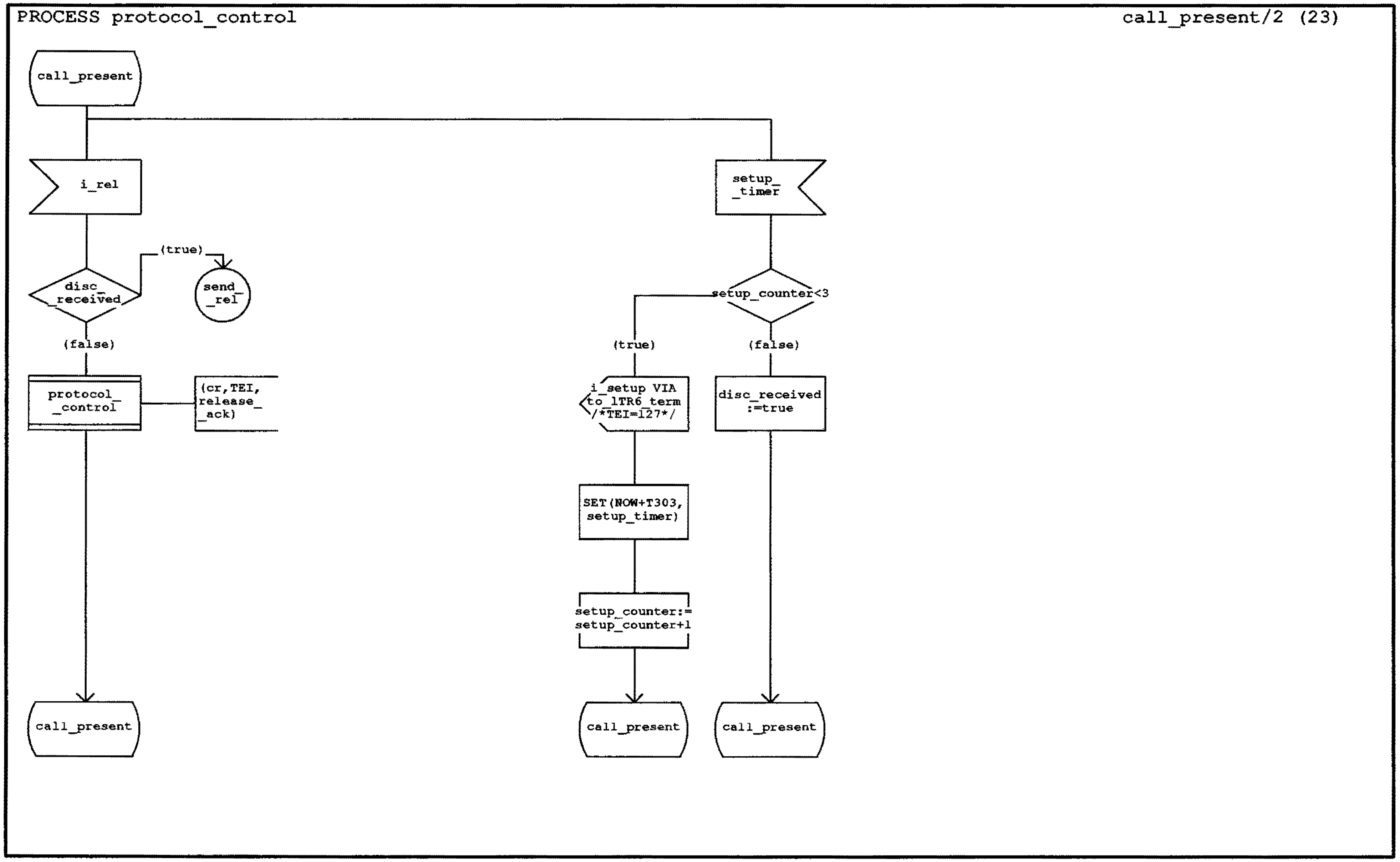


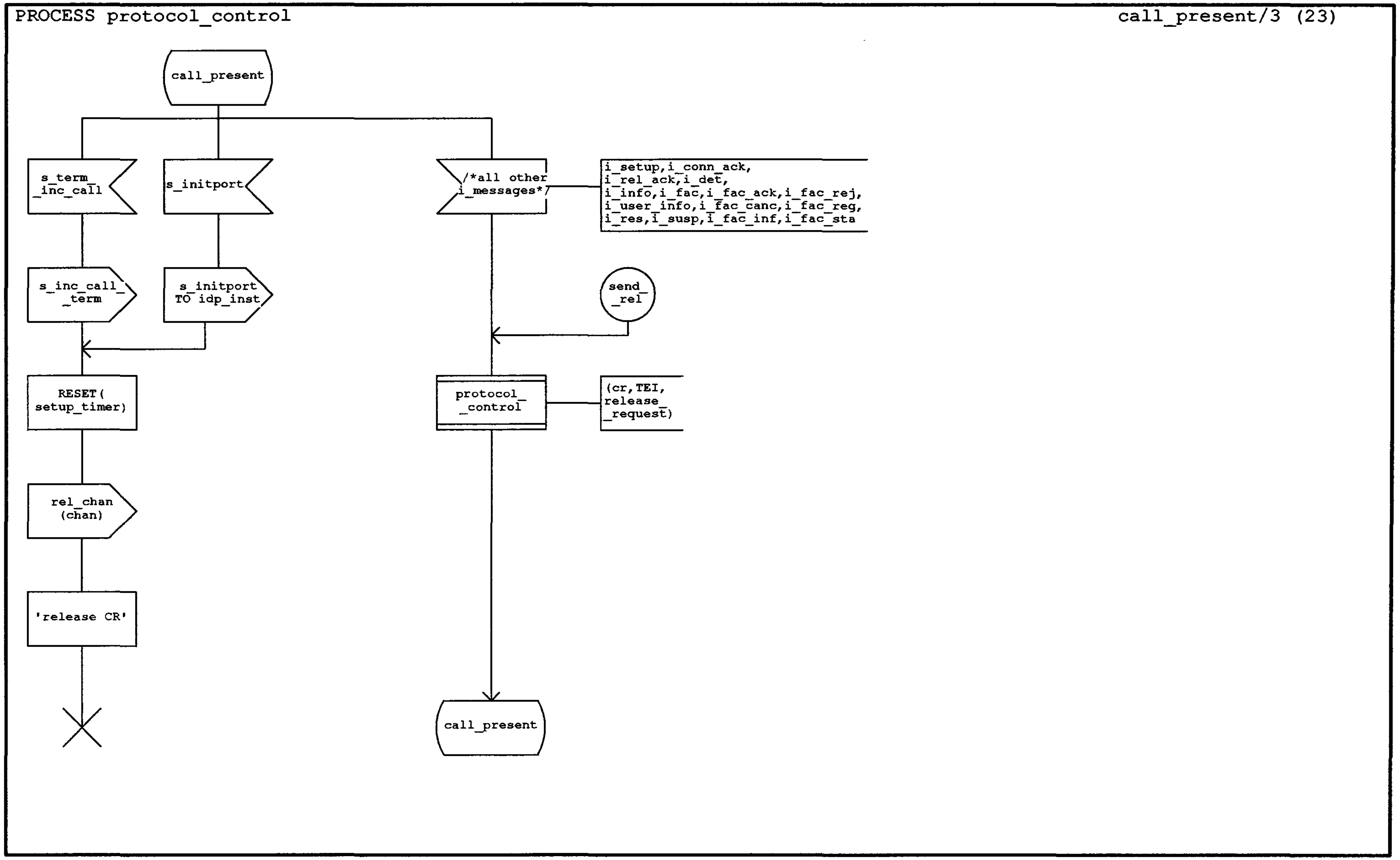

PROCESS protocol_control
init_cr_pend/1 (23)
init_cr_pend
s_pts
i_setup_ack VIA to_1TR6_ _term
dial
digits_ _in_ _buffer
s_inc_ _call
call_ _request_ _is
(data_call)
(voice_call)
s_inc_ _disr
s_call_ _request
init_cr_ _pend
protocol_ _control
/*choose a free CR*/ (cr,127, call_present)
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request
i_disc
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request
i_rel
s_on_hook
rel_chan (chan)
i_rel_ack VIA to_1TR6_ _term
'release_ layer_2'
'release CR'
s_clear_ _call
reason_ _for_ _clearing
(a)
(b)
i_disc VIA to_1TR6_ _term
SET(NOW+T305, disc_timer)
disconnect_ _indication
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request

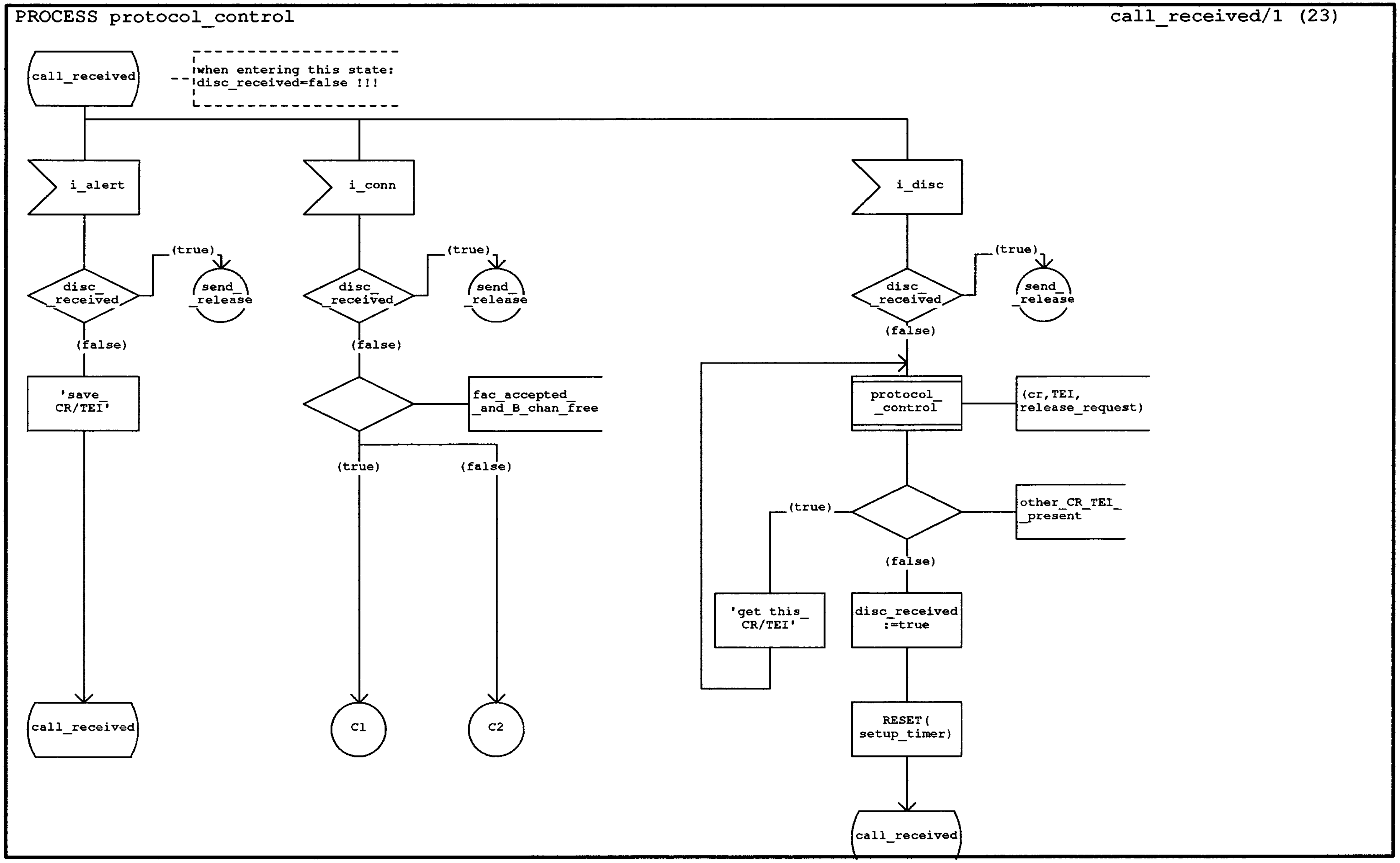
PROCESS protocol_control init_cr_pend/2 (23)



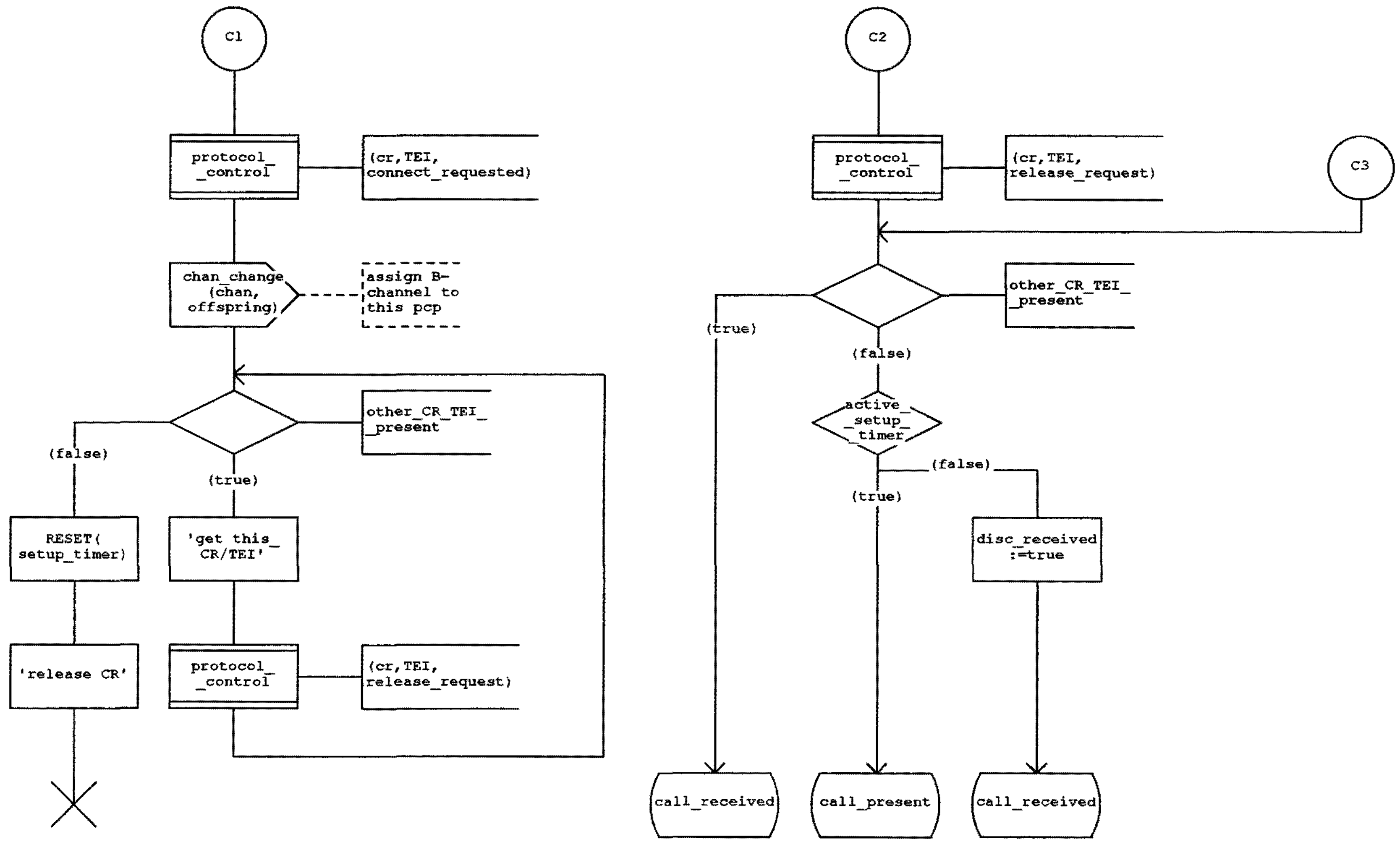

PROCESS protocol_control
call_present/1 (23)
call_present
when entering this state:
disc_received=false !!!
i_alert
i_conn
i_disc
it is not possible to
send s_inc_call_disr
to Sopho_s; so we have
to wait until Sopho_s
will send term_inc_call!
disc_received
(true)
send_rel
(false)
'save_CR/TEI'
fac_accepted_and_B_chan_free
CP1
protocol_control
(cr,TEI, release_request)
RESET(setup_timer)
(cr,TEI, connect_requested)
disc_received:= true
chan_change (chan, offspring)
'release CR'
call_received
call_present

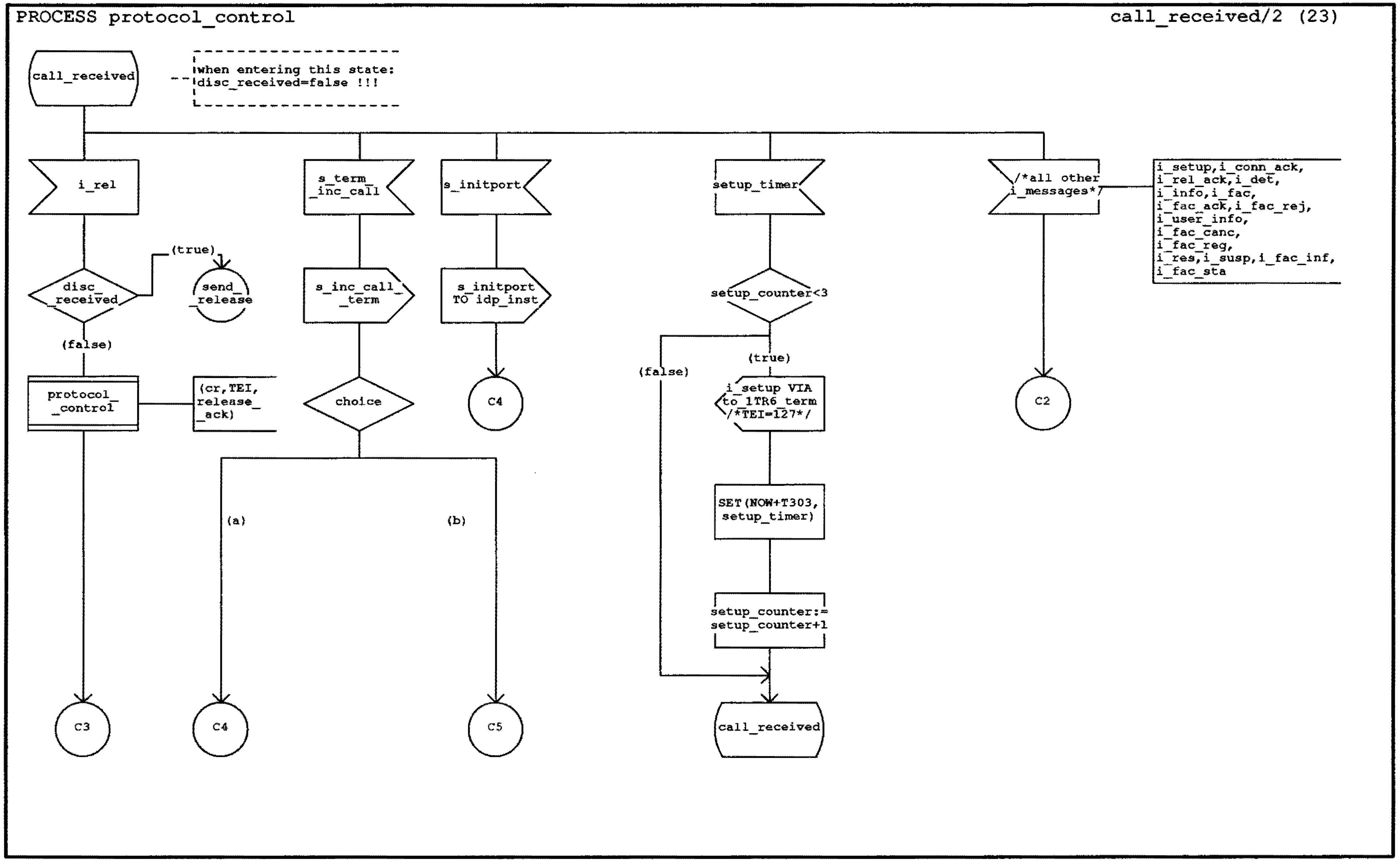

PROCESS protocol_control
call_present/2 (23)
call_present
i_rel
disc_received
(true)
send_rel
(false)
protocol_control
(cr,TEI,release_ack)
call_present
setup_timer
setup_counter<3
(true)
(false)
i_setup VIA to_1TR6_term /*TEI=127*/
SET(NOW+T303, setup_timer)
setup_counter:= setup_counter+1
disc_received :=true
call_present
call_present


PROCESS protocol_control
call_present/3 (23)
call_present
s_term_inc_call
s_initport
/*all other i_messages*/
i_setup,i_conn_ack,
i_rel_ack,i_det,
i_info,i_fac,i_fac_ack,i_fac_rej,
i_user_info,i_fac_canc,i_fac_reg,
i_res,i_susp,i_fac_inf,i_fac_sta
s_inc_call_term
s_initport TO idp_inst
send_rel
RESET(setup_timer)
protocol_control
(cr,TEI,release_request)
rel_chan (chan)
'release CR'
call_present

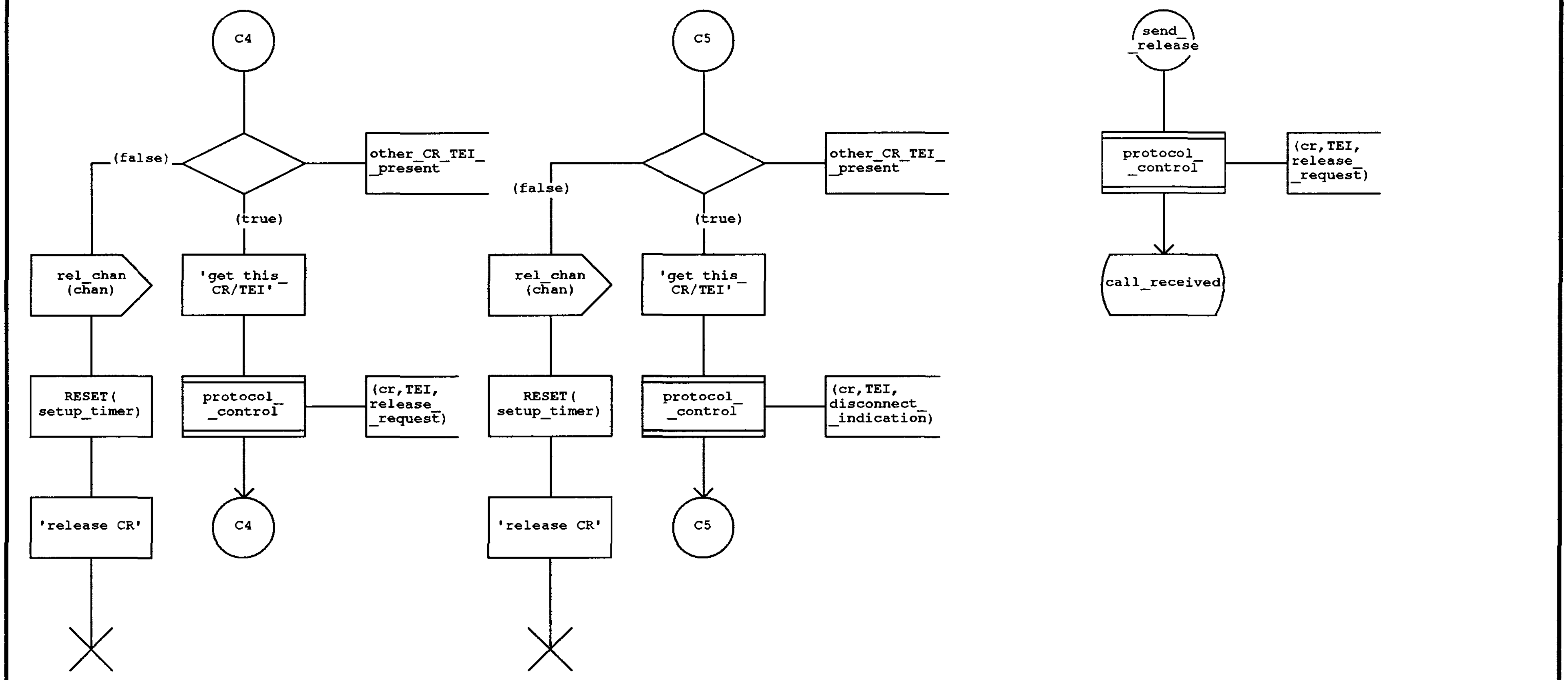

PROCESS protocol_control
call_received/1 (23)
call_received
when entering this state:
disc_received=false !!!
i_alert
i_conn
i_disc
disc_
_received
(true)
send_
_release
(false)
'save_
CR/TEI'
fac_accepted_
_and_B_chan_free
(true)
(false)
protocol_
_control
(cr,TEI,
release_request)
other_CR_TEI_
_present
'get this_
CR/TEI'
disc_received
:=true
RESET(
setup_timer)
C1
C2
call_received

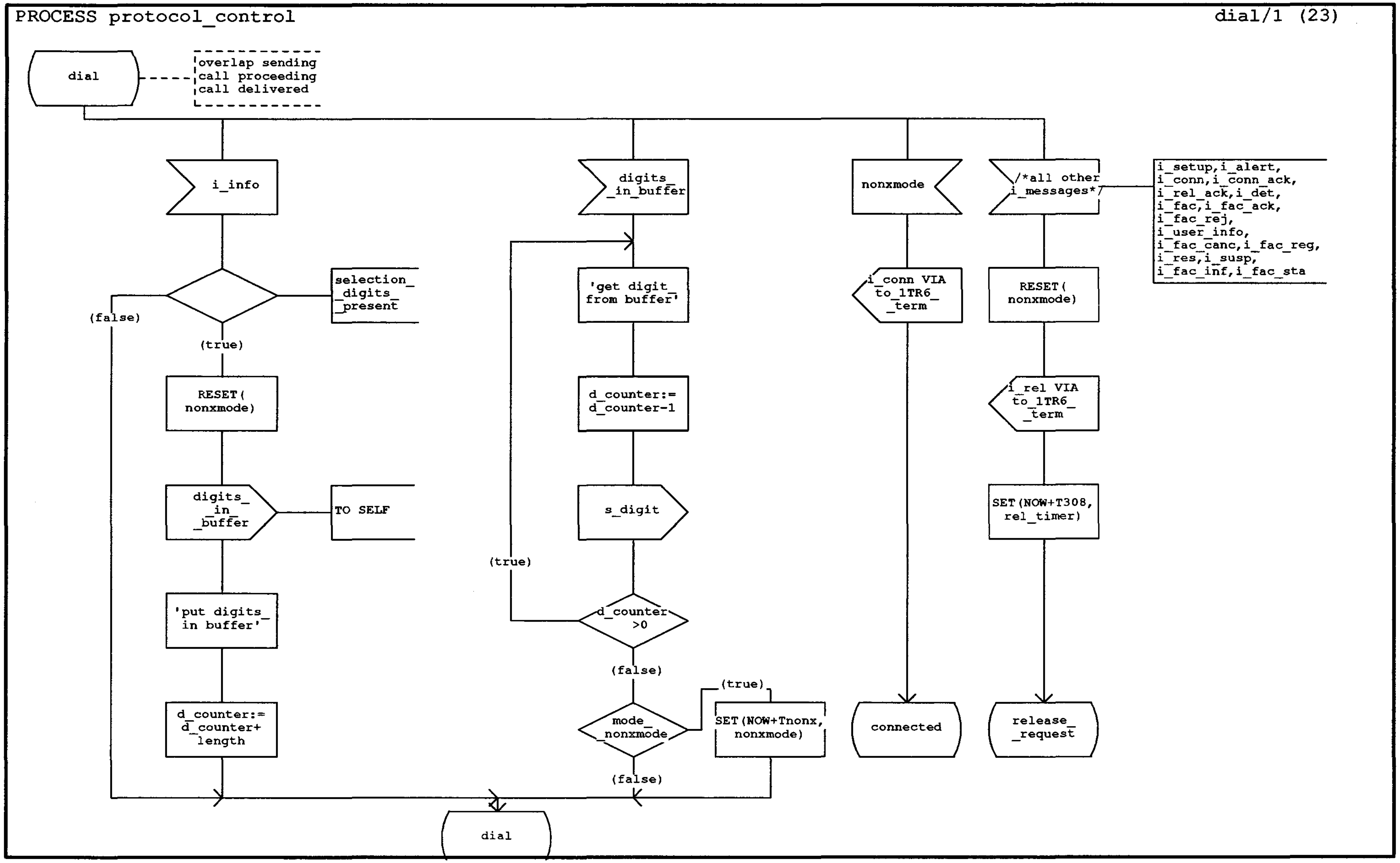
PROCESS protocol_control call_received/1/C (23)

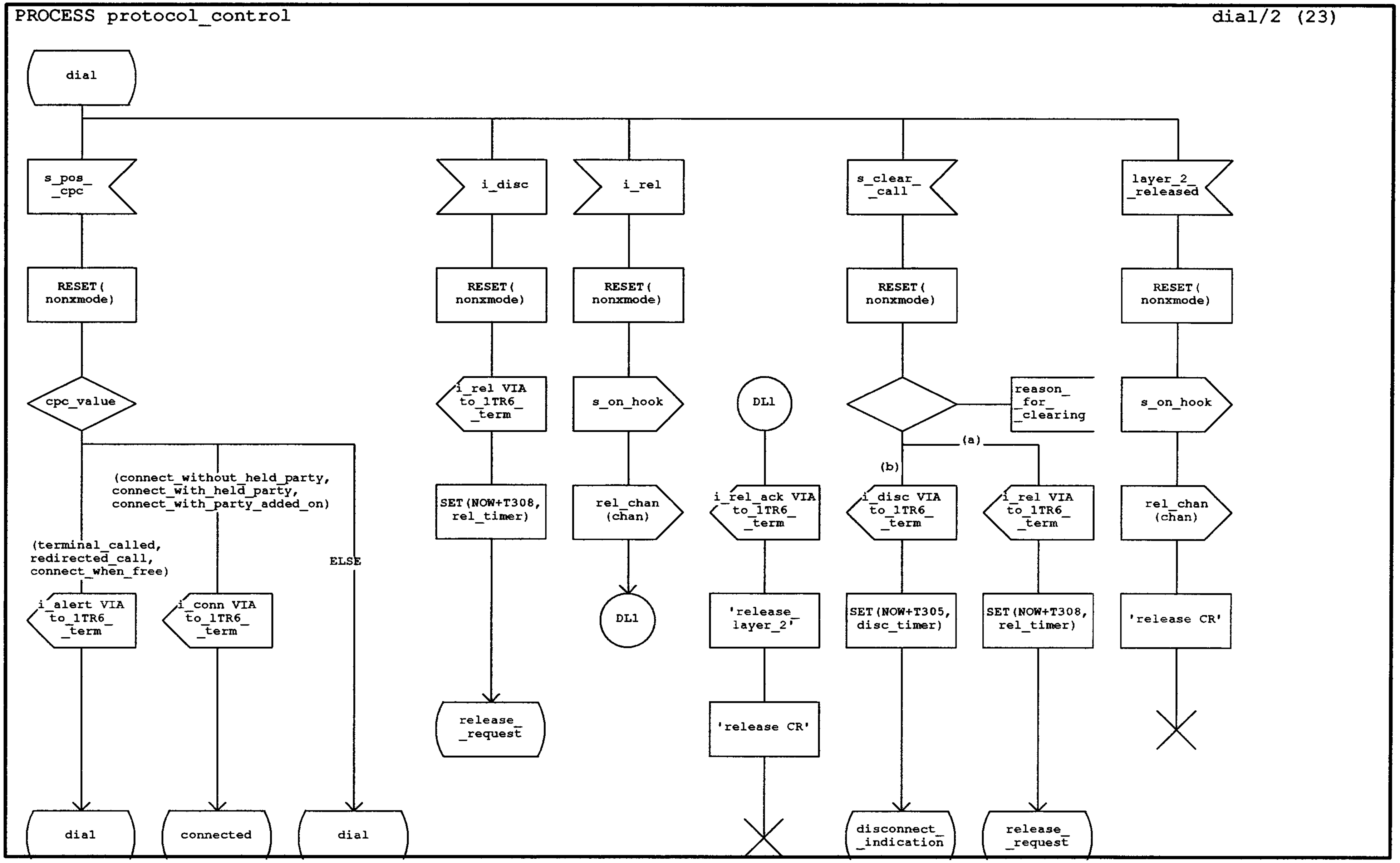


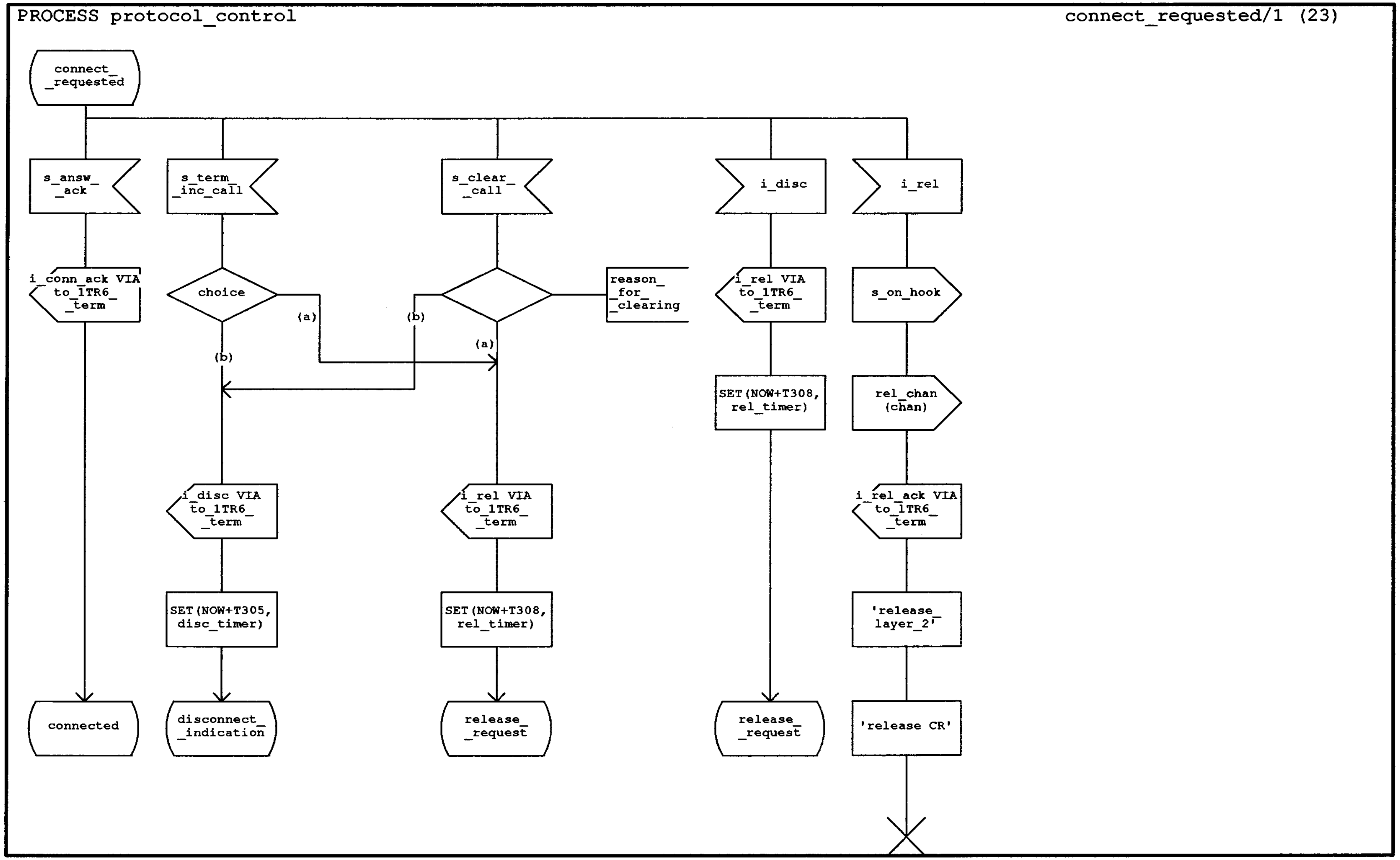

PROCESS protocol_control
call_received/2 (23)
call_received
when entering this state:
disc_received=false !!!
i_rel
s_term_inc_call
s_initport
setup_timer
/*all other i_messages*/
i_setup,i_conn_ack,
i_rel_ack,i_det,
i_info,i_fac,
i_fac_ack,i_fac_rej,
i_user_info,
i_fac_canc,
i_fac_reg,
i_res,i_susp,i_fac_inf,
i_fac_sta
disc_received
(true)
send_release
(false)
protocol_control
(cr,TEI,release_ack)
s_inc_call_term
choice
(a)
(b)
s_initport TO idp_inst
C4
setup_counter<3
(true)
(false)
i_setup VIA to_1TR6_term /*TEI=127*/
SET(NOW+T303, setup_timer)
setup_counter:= setup_counter+1
call_received
C2
C3
C4
C5

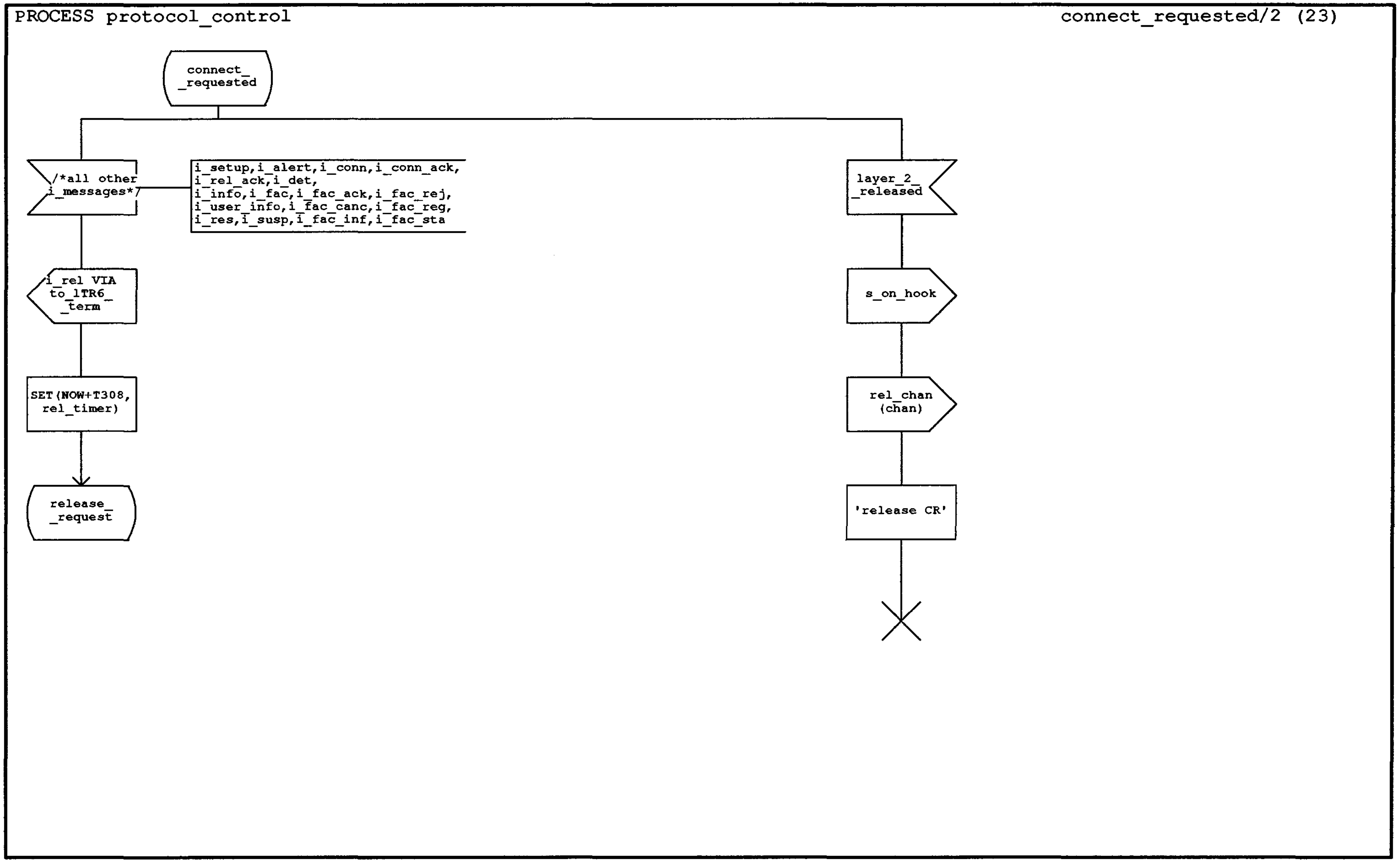
PROCESS protocol_control call_received/2/C (23)

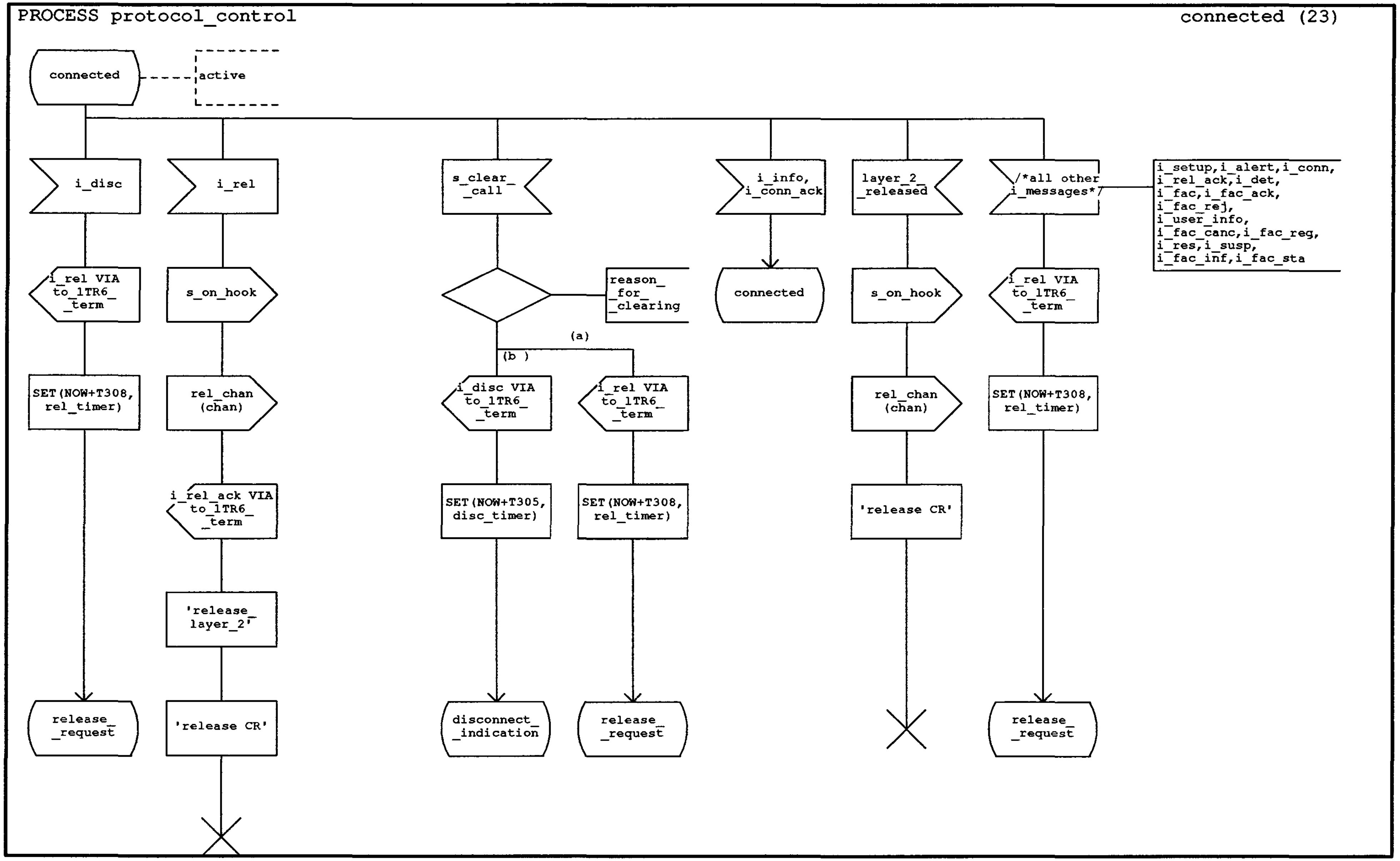


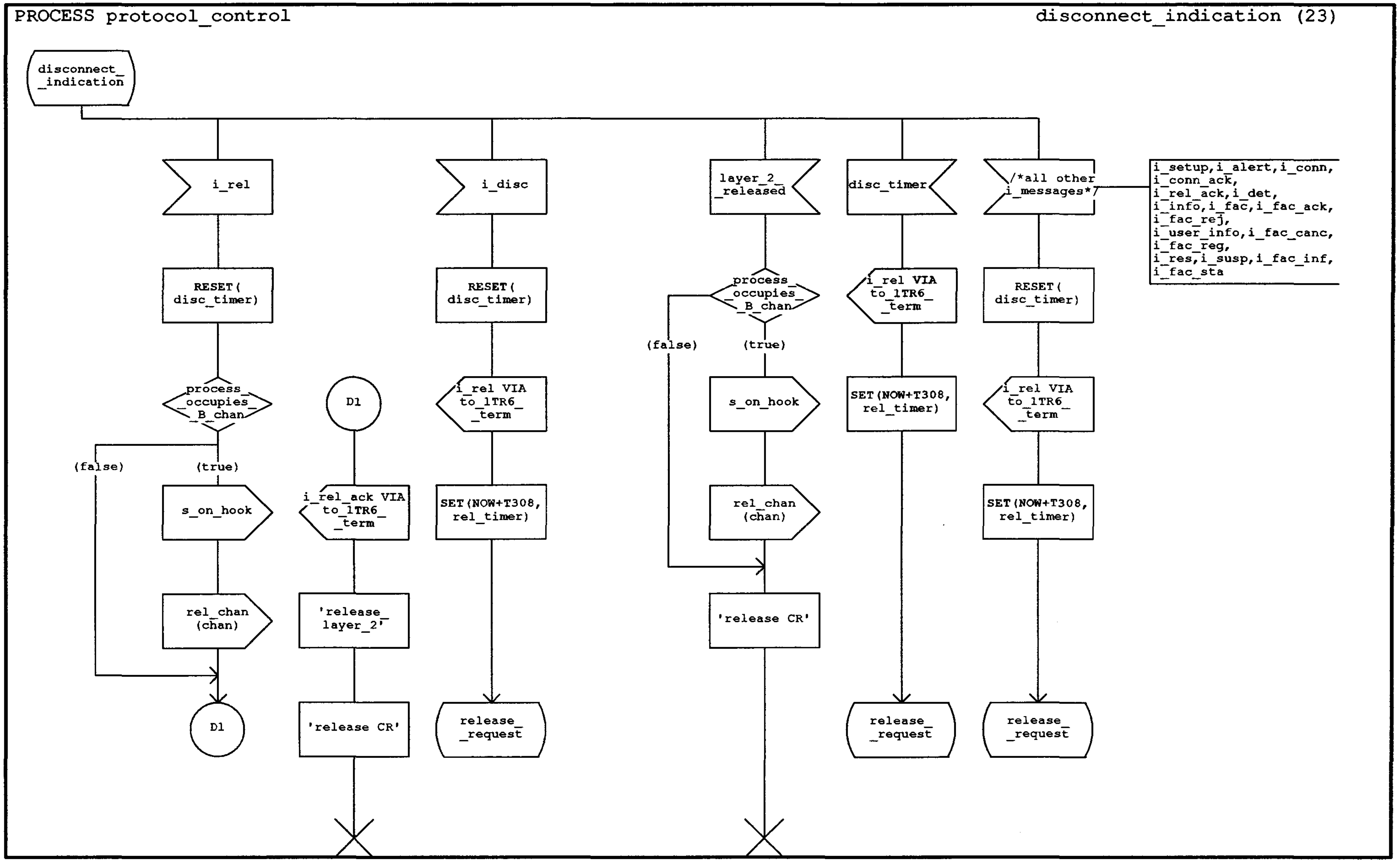

PROCESS protocol_control
dial/1 (23)
dial
overlap sending
call proceeding
call delivered
i_info
selection_
digits
_present
(false)
(true)
RESET(
nonxmode)
digits_
_in_
_buffer
TO SELF
'put digits_
in buffer'
d_counter:=
d_counter+
length
digits_
_in_buffer
'get digit_
from buffer'
d_counter:=
d_counter-1
s_digit
(true)
d_counter
>0
(false)
mode_
_nonxmode
(true)
SET(NOW+Tnonx,
nonxmode)
(false)
dial
nonxmode
i_conn VIA
to_1TR6_
_term
connected
/*all other
i_messages*/
i_setup,i_alert,
i_conn,i_conn_ack,
i_rel_ack,i_det,
i_fac,i_fac_ack,
i_fac_rej,
i_user_info,
i_fac_canc,i_fac_reg,
i_res,i_susp,
i_fac_inf,i_fac_sta
RESET(
nonxmode)
i_rel VIA
to_1TR6_
_term
SET(NOW+T308,
rel_timer)
release_
_request

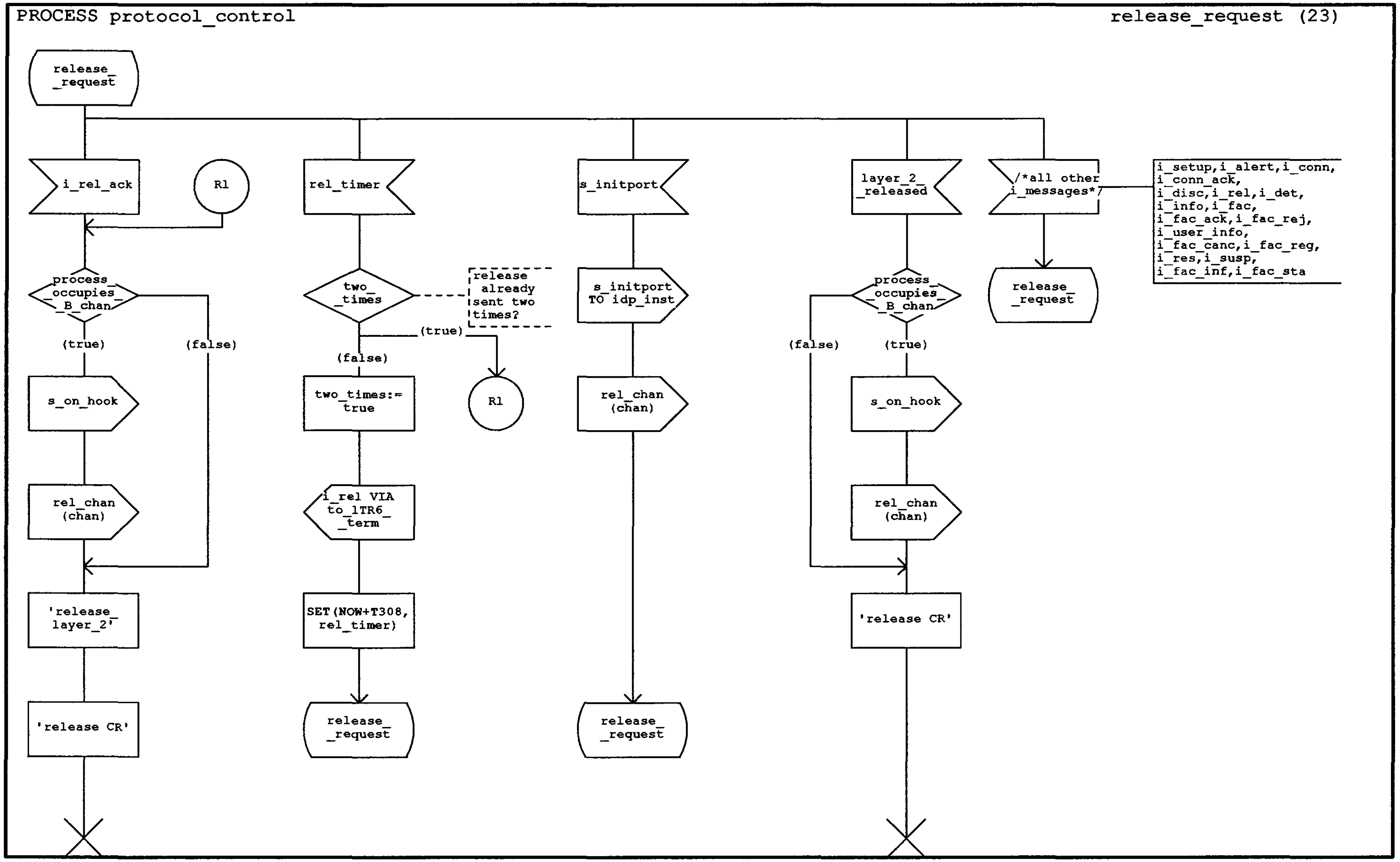

PROCESS protocol_control
dial/2 (23)
dial
s_pos_ _cpc
RESET(nonxmode)
cpc_value
(terminal_called, redirected_call, connect_when_free)
(connect_without_held_party, connect_with_held_party, connect_with_party_added_on)
ELSE
i_alert VIA to_1TR6_ _term
i_conn VIA to_1TR6_ _term
dial
connected
dial
i_disc
RESET(nonxmode)
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request
i_rel
RESET(nonxmode)
s_on_hook
rel_chan (chan)
DL1
DL1
i_rel_ack VIA to_1TR6_ _term
'release_ layer_2'
'release CR'
s_clear_ _call
RESET(nonxmode)
reason_ _for_ _clearing
(a)
(b)
i_disc VIA to_1TR6_ _term
SET(NOW+T305, disc_timer)
disconnect_ _indication
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request
layer_2_ _released
RESET(nonxmode)
s_on_hook
rel_chan (chan)
'release CR'

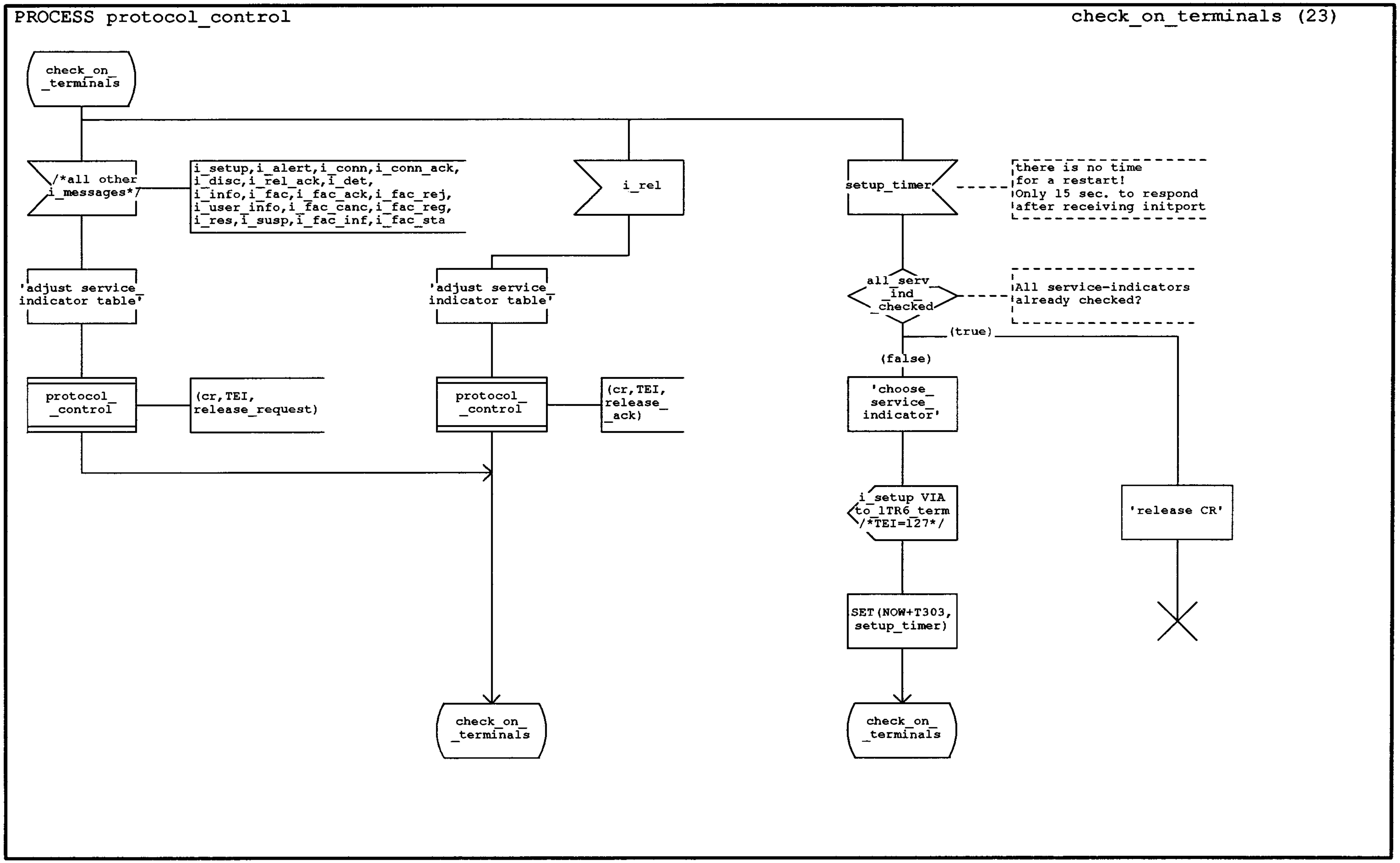

PROCESS protocol_control
connect_requested/1 (23)
connect_
_requested
s_answ_
_ack
s_term_
_inc_call
s_clear_
_call
i_disc
i_rel
i_conn_ack VIA
to_1TR6_
_term
choice
(a)
(b)
(b)
(a)
reason_
for
_clearing
i_rel VIA
to_1TR6_
_term
s_on_hook
SET(NOW+T308,
rel_timer)
rel_chan
(chan)
i_disc VIA
to_1TR6_
_term
i_rel VIA
to_1TR6_
_term
i_rel_ack VIA
to_1TR6_
_term
SET(NOW+T305,
disc_timer)
SET(NOW+T308,
rel_timer)
'release_
layer_2'
connected
disconnect_
_indication
release_
_request
release_
_request
'release CR'

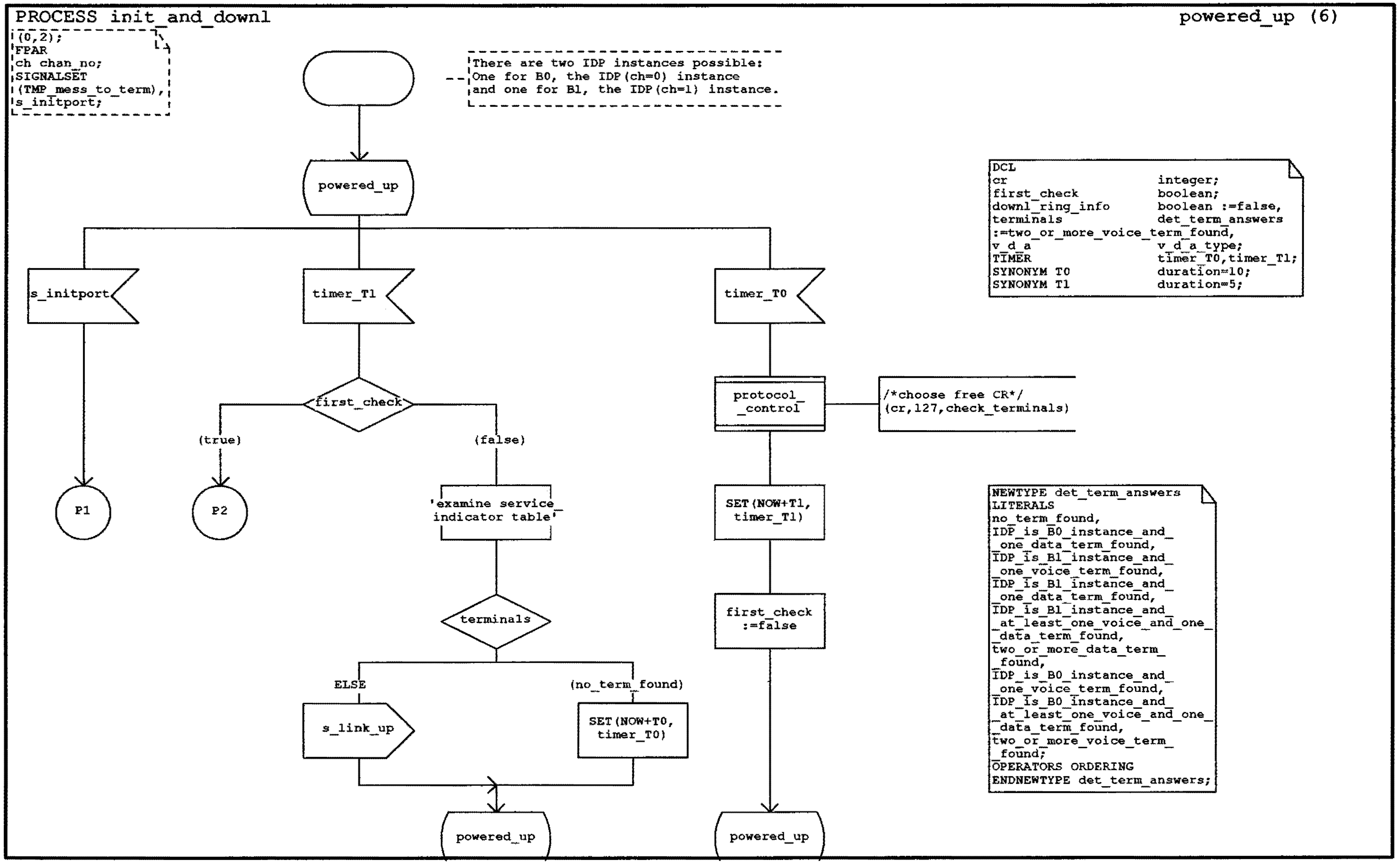

PROCESS protocol_control
connect_requested/2 (23)
connect_
_requested
/*all other
i_messages*/
i_setup,i_alert,i_conn,i_conn_ack,
i_rel_ack,i_det,
i_info,i_fac,i_fac_ack,i_fac_rej,
i_user_info,i_fac_canc,i_fac_reg,
i_res,i_susp,i_fac_inf,i_fac_sta
i_rel VIA
to_1TR6_
_term
SET(NOW+T308,
rel_timer)
release_
_request
layer_2_
_released
s_on_hook
rel_chan
(chan)
'release CR'


PROCESS protocol_control
connected (23)
connected
active
i_disc
i_rel
s_clear_
_call
i_info,
i_conn_ack
layer_2_
_released
/*all other
i_messages*/
i_setup,i_alert,i_conn,
i_rel_ack,i_det,
i_fac,i_fac_ack,
i_fac_rej,
i_user_info,
i_fac_canc,i_fac_reg,
i_res,i_susp,
i_fac_inf,i_fac_sta
i_rel VIA
to_1TR6_
_term
s_on_hook
reason_
for
_clearing
connected
s_on_hook
i_rel VIA
to_1TR6_
_term
(a)
(b)
SET(NOW+T308,
rel_timer)
rel_chan
(chan)
i_disc VIA
to_1TR6_
_term
i_rel VIA
to_1TR6_
_term
rel_chan
(chan)
SET(NOW+T308,
rel_timer)
i_rel_ack VIA
to_1TR6_
_term
SET(NOW+T305,
disc_timer)
SET(NOW+T308,
rel_timer)
'release CR'
'release_
layer_2'
release_
_request
'release CR'
disconnect_
_indication
release_
_request
release_
_request


PROCESS protocol_control
disconnect_indication (23)
disconnect_ _indication
i_rel
RESET(disc_timer)
process_ occupies_ _B_chan
(false)
(true)
s_on_hook
rel_chan (chan)
D1
D1
i_rel_ack VIA to_1TR6_ _term
'release_ layer_2'
'release CR'
i_disc
RESET(disc_timer)
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request
layer_2_ _released
process_ occupies_ _B_chan
(false)
(true)
s_on_hook
rel_chan (chan)
'release CR'
disc_timer
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request
/*all other i_messages*/
i_setup,i_alert,i_conn, i_conn_ack, i_rel_ack,i_det, i_info,i_fac,i_fac_ack, i_fac_rej, i_user_info,i_fac_canc, i_fac_reg, i_res,i_susp,i_fac_inf, i_fac_sta
RESET(disc_timer)
i_rel VIA to_1TR6_ _term
SET(NOW+T308, rel_timer)
release_ _request


PROCESS protocol_control
release_request (23)
release_
_request
i_rel_ack
R1
process_
occupies_
_B_chan
(true)
(false)
s_on_hook
rel_chan
(chan)
'release_
layer_2'
'release CR'
rel_timer
two_
_times
release
already
sent two
times?
(true)
(false)
R1
two_times:=
true
i_rel VIA
to_1TR6_
_term
SET(NOW+T308,
rel_timer)
release_
_request
s_initport
s_initport
TO idp_inst
rel_chan
(chan)
release_
_request
layer_2_
_released
process_
occupies_
_B_chan
(false)
(true)
s_on_hook
rel_chan
(chan)
'release CR'
/*all other
i_messages*/
i_setup,i_alert,i_conn,
i_conn_ack,
i_disc,i_rel,i_det,
i_info,i_fac,
i_fac_ack,i_fac_rej,
i_user_info,
i_fac_canc,i_fac_reg,
i_res,i_susp,
i_fac_inf,i_fac_sta
release_
_request


PROCESS protocol_control
check_on_terminals (23)
check_on_
_terminals
/*all other
i_messages*/
i_setup,i_alert,i_conn,i_conn_ack,
i_disc,i_rel_ack,i_det,
i_info,i_fac,i_fac_ack,i_fac_rej,
i_user_info,i_fac_canc,i_fac_reg,
i_res,i_susp,i_fac_inf,i_fac_sta
i_rel
setup_timer
there is no time
for a restart!
Only 15 sec. to respond
after receiving initport
'adjust service_
indicator table'
'adjust service_
indicator table'
all_serv_
ind
_checked
All service-indicators
already checked?
(true)
(false)
protocol_
_control
(cr,TEI,
release_request)
protocol_
_control
(cr,TEI,
release_
_ack)
'choose_
service_
indicator'
i_setup VIA
to_1TR6_term
/*TEI=127*/
'release CR'
SET(NOW+T303,
setup_timer)
check_on_
_terminals
check_on_
_terminals



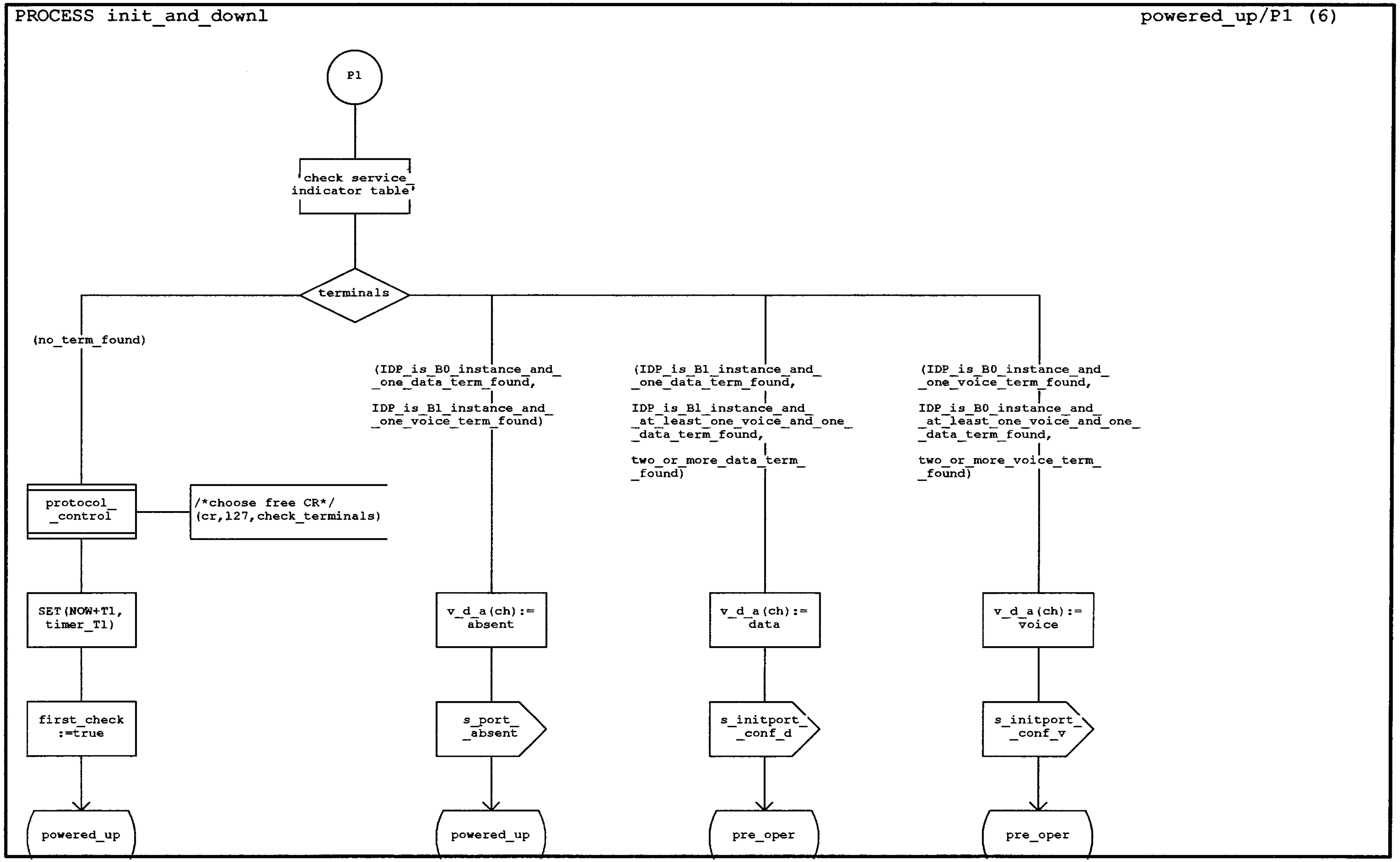
```
(0,2);
FPAR
ch chan_no;
SIGNALSET
(TMP_mess_to_term),
s_initport;
```

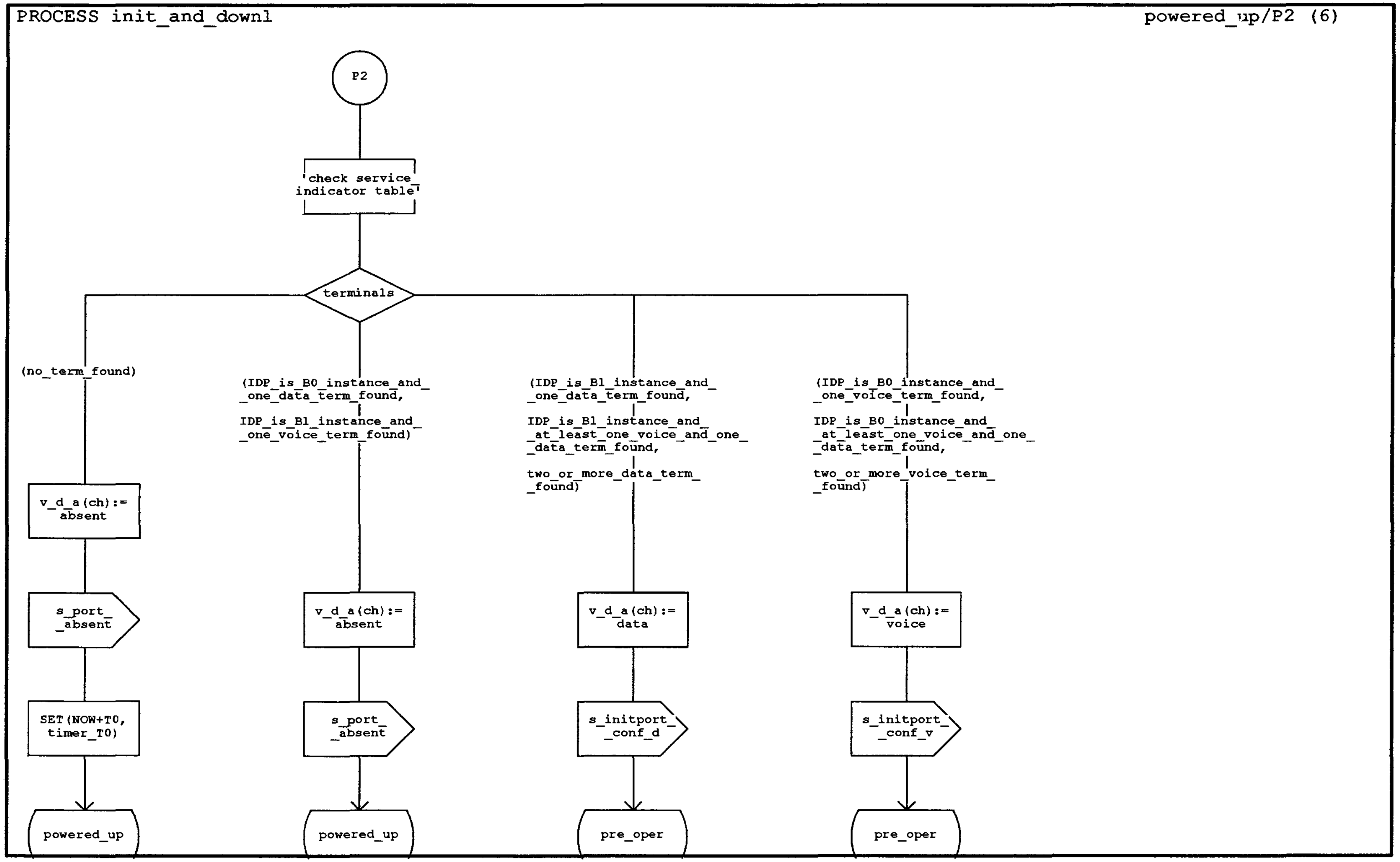
```
There are two IDP instances possible:
One for B0, the IDP(ch=0) instance
and one for B1, the IDP(ch=1) instance.
```

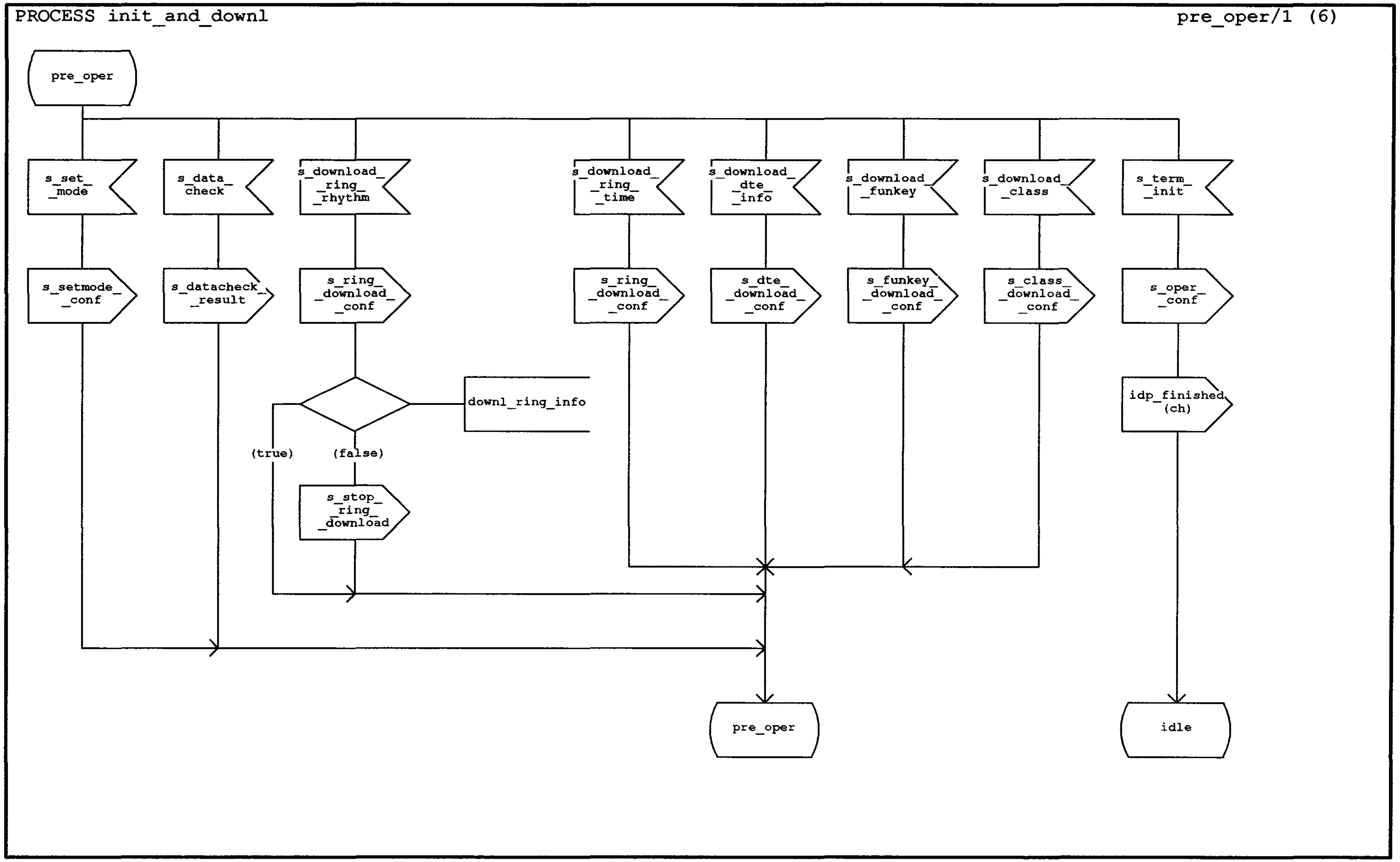
```
/*choose free CR*/
(cr,127,check_terminals)
```

```
DCL
cr                        integer;
first_check               boolean;
downl_ring_info           boolean :=false,
terminals                 det_term_answers
:=two_or_more_voice_term_found,
v_d_a                     v_d_a_type;
TIMER                     timer_T0,timer_T1;
SYNONYM T0                duration=10;
SYNONYM T1                duration=5;
```

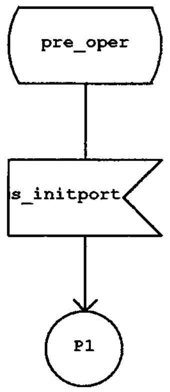
```
NEWTYPE det_term_answers
LITERALS
no_term_found,
IDP_is_B0_instance_and_
_one_data_term_found,
IDP_is_B1_instance_and_
_one_voice_term_found,
IDP_is_B1_instance_and_
_one_data_term_found,
IDP_is_B1_instance_and_
_at_least_one_voice_and_one_
_data_term_found,
two_or_more_data_term_
_found,
IDP_is_B0_instance_and_
_one_voice_term_found,
IDP_is_B0_instance_and_
_at_least_one_voice_and_one_
_data_term_found,
two_or_more_voice_term_
_found;
OPERATORS ORDERING
ENDNEWTYPE det_term_answers;
```


PROCESS init_and_downl
powered_up/P1 (6)
P1
'check service_ indicator table'
terminals
(no_term_found)
(IDP_is_B0_instance_and_ _one_data_term_found,
IDP_is_B1_instance_and_ _one_voice_term_found)
(IDP_is_B1_instance_and_ _one_data_term_found,
IDP_is_B1_instance_and_ _at_least_one_voice_and_one_ _data_term_found,
two_or_more_data_term_ _found)
(IDP_is_B0_instance_and_ _one_voice_term_found,
IDP_is_B0_instance_and_ _at_least_one_voice_and_one_ _data_term_found,
two_or_more_voice_term_ _found)
protocol_ _control
/*choose free CR*/ (cr,127,check_terminals)
SET(NOW+T1, timer_T1)
first_check :=true
powered_up
v_d_a(ch):= absent
s_port_ _absent
powered_up
v_d_a(ch):= data
s_initport_ _conf_d
pre_oper
v_d_a(ch):= voice
s_initport_ _conf_v
pre_oper

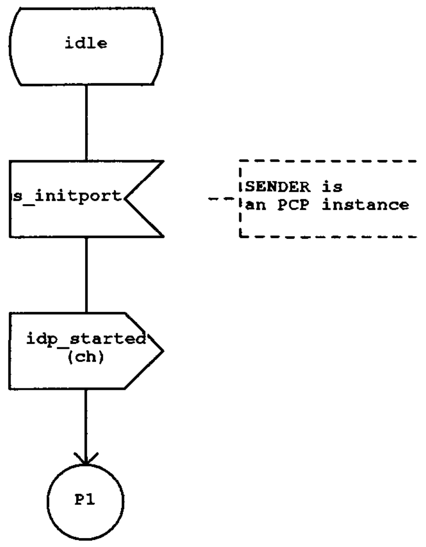

PROCESS init_and_downl
powered_up/P2 (6)
P2
'check service_ indicator table'
terminals
(no_term_found)
(IDP_is_B0_instance_and_ _one_data_term_found, IDP_is_B1_instance_and_ _one_voice_term_found)
(IDP_is_B1_instance_and_ _one_data_term_found, IDP_is_B1_instance_and_ _at_least_one_voice_and_one_ _data_term_found, two_or_more_data_term_ _found)
(IDP_is_B0_instance_and_ _one_voice_term_found, IDP_is_B0_instance_and_ _at_least_one_voice_and_one_ _data_term_found, two_or_more_voice_term_ _found)
v_d_a(ch):= absent
s_port_ _absent
SET(NOW+T0, timer_T0)
powered_up
v_d_a(ch):= absent
s_port_ _absent
powered_up
v_d_a(ch):= data
s_initport_ _conf_d
pre_oper
v_d_a(ch):= voice
s_initport_ _conf_v
pre_oper

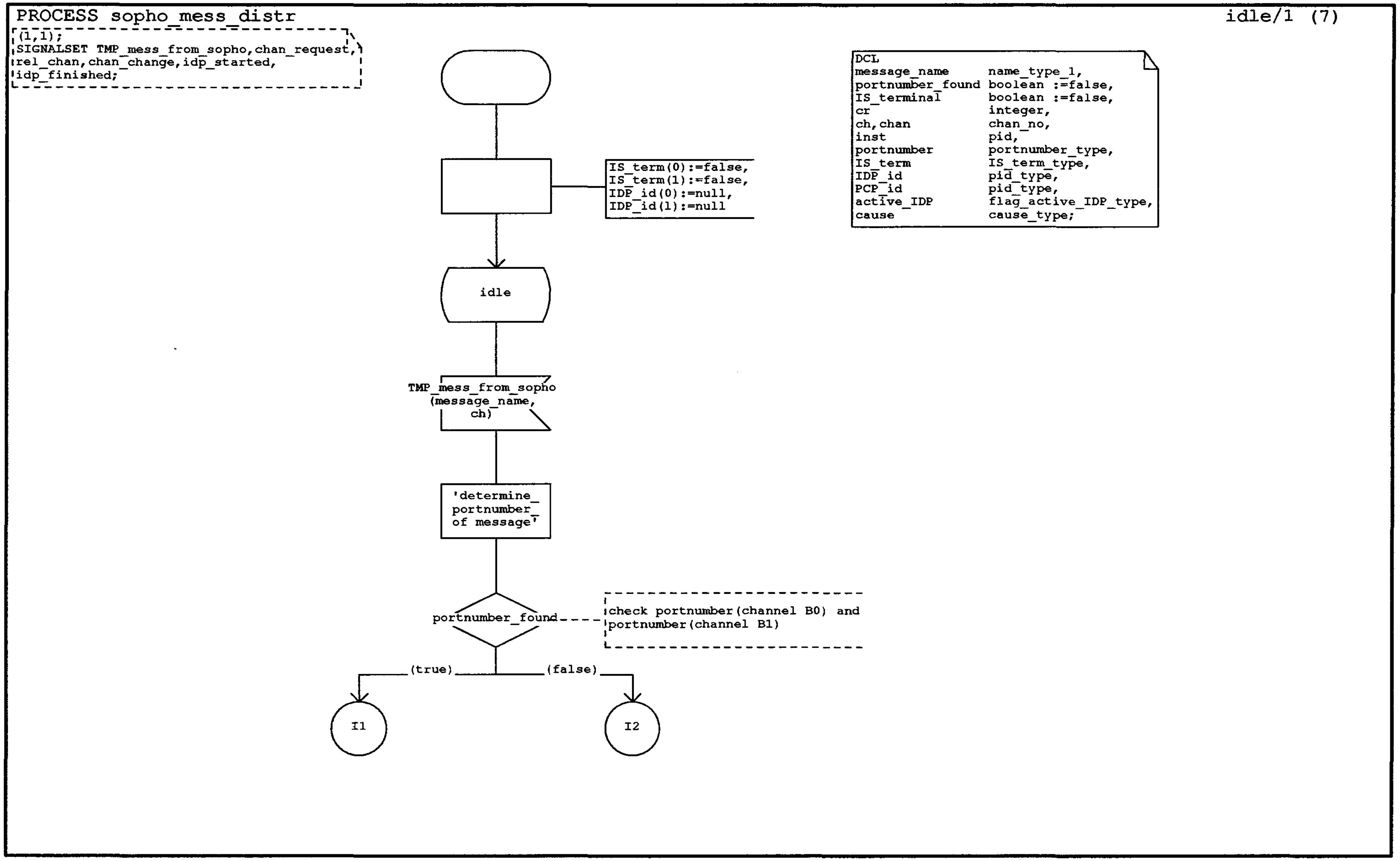

PROCESS init_and_downl
pre_oper/1 (6)
pre_oper
s_set_ _mode
s_data_ check
s_download_ _ring_ _rhythm
s_download_ _ring_ _time
s_download_ _dte_ _info
s_download_ _funkey
s_download_ _class
s_term_ _init
s_setmode_ _conf
s_datacheck_ _result
s_ring_ _download_ _conf
s_ring_ _download_ _conf
s_dte_ _download_ _conf
s_funkey_ _download_ _conf
s_class_ _download_ _conf
s_oper_ _conf
downl_ring_info
(true)
(false)
idp_finished (ch)
s_stop_ _ring_ _download
pre_oper
idle

PROCESS init_and_downl pre_oper/2 (6)

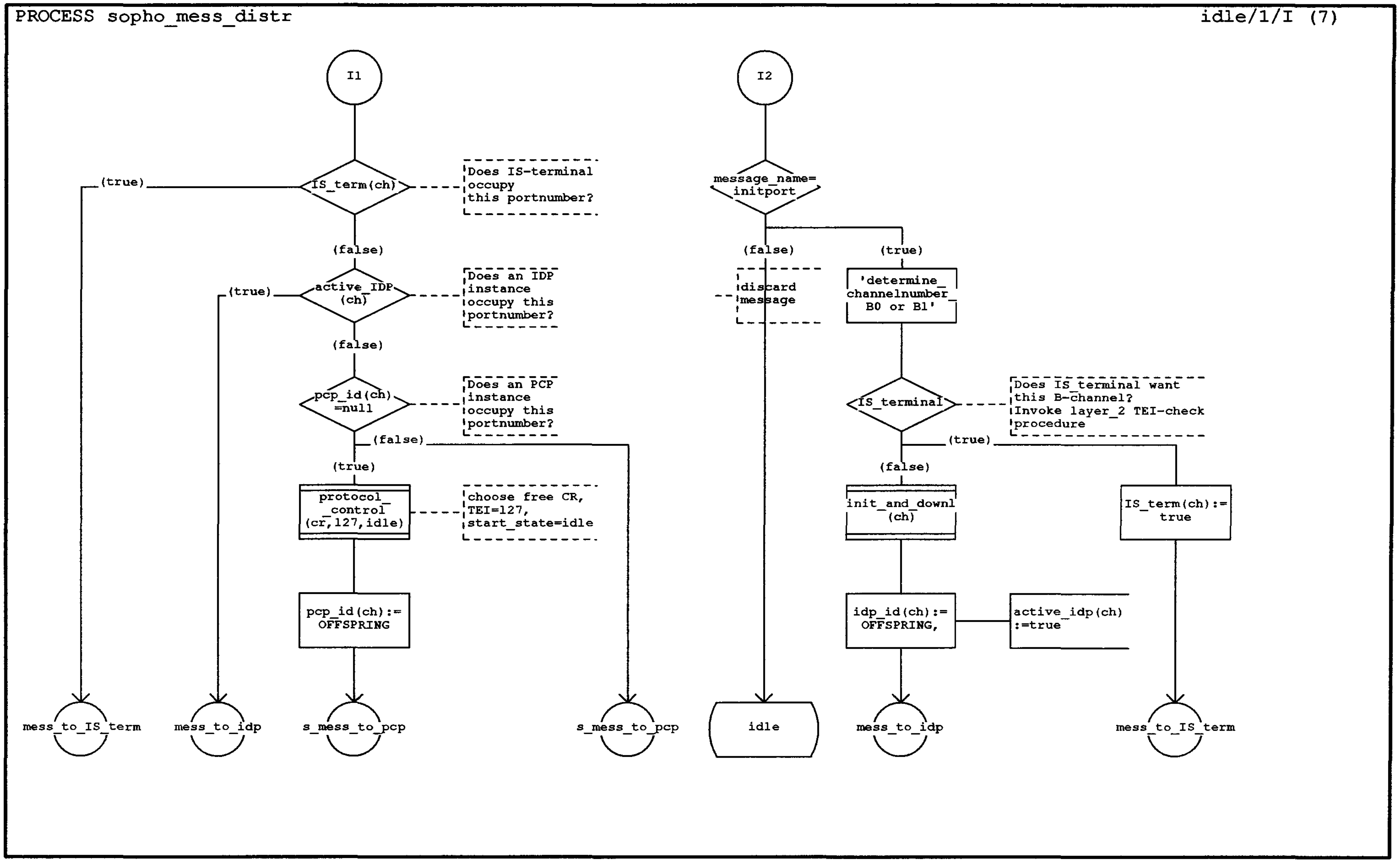


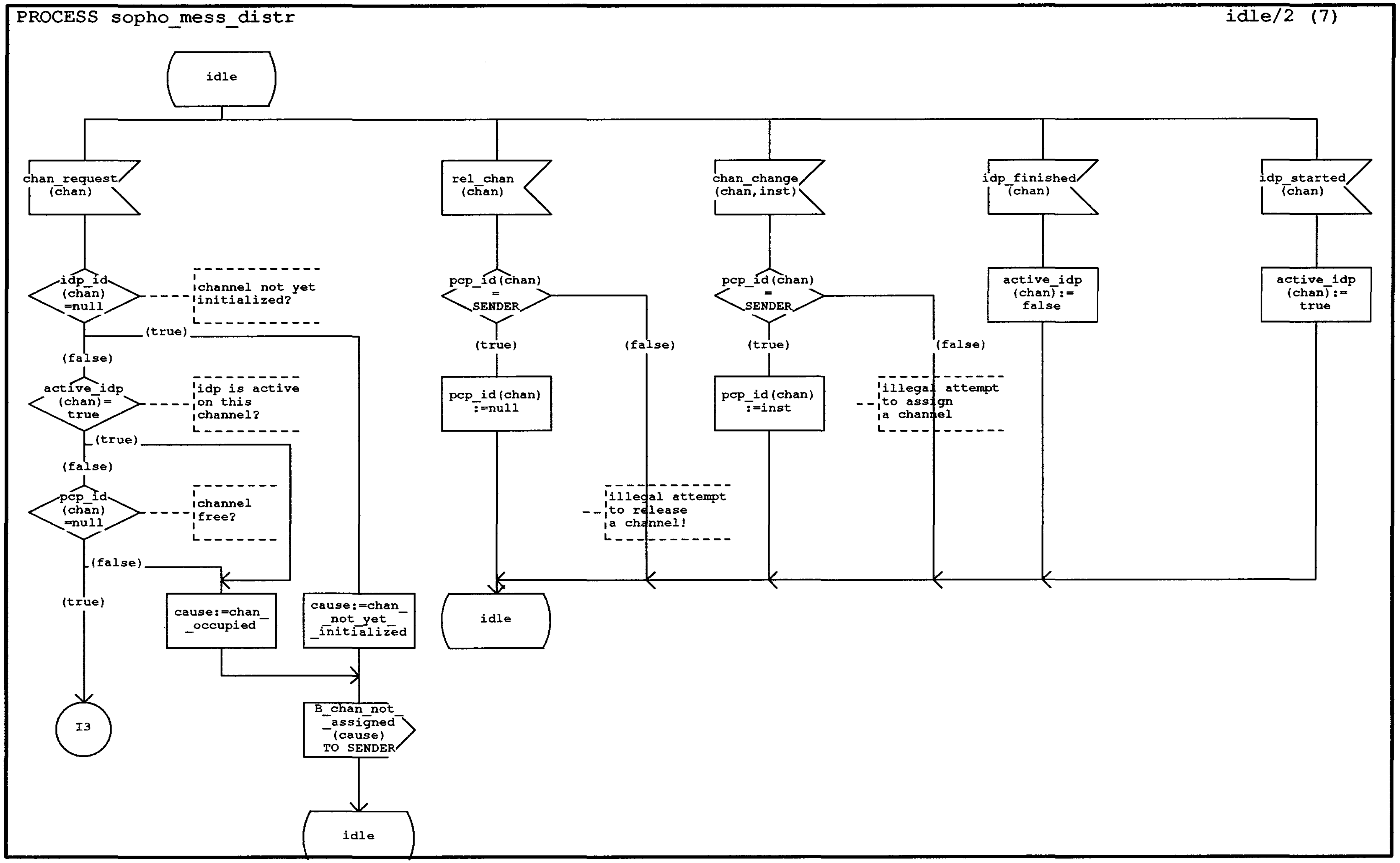
PROCESS init_and_downl   idle (6)



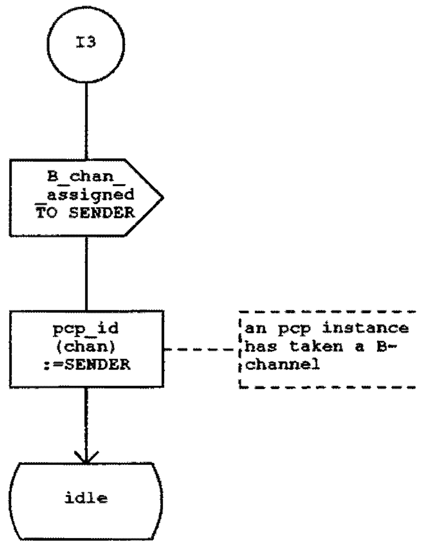


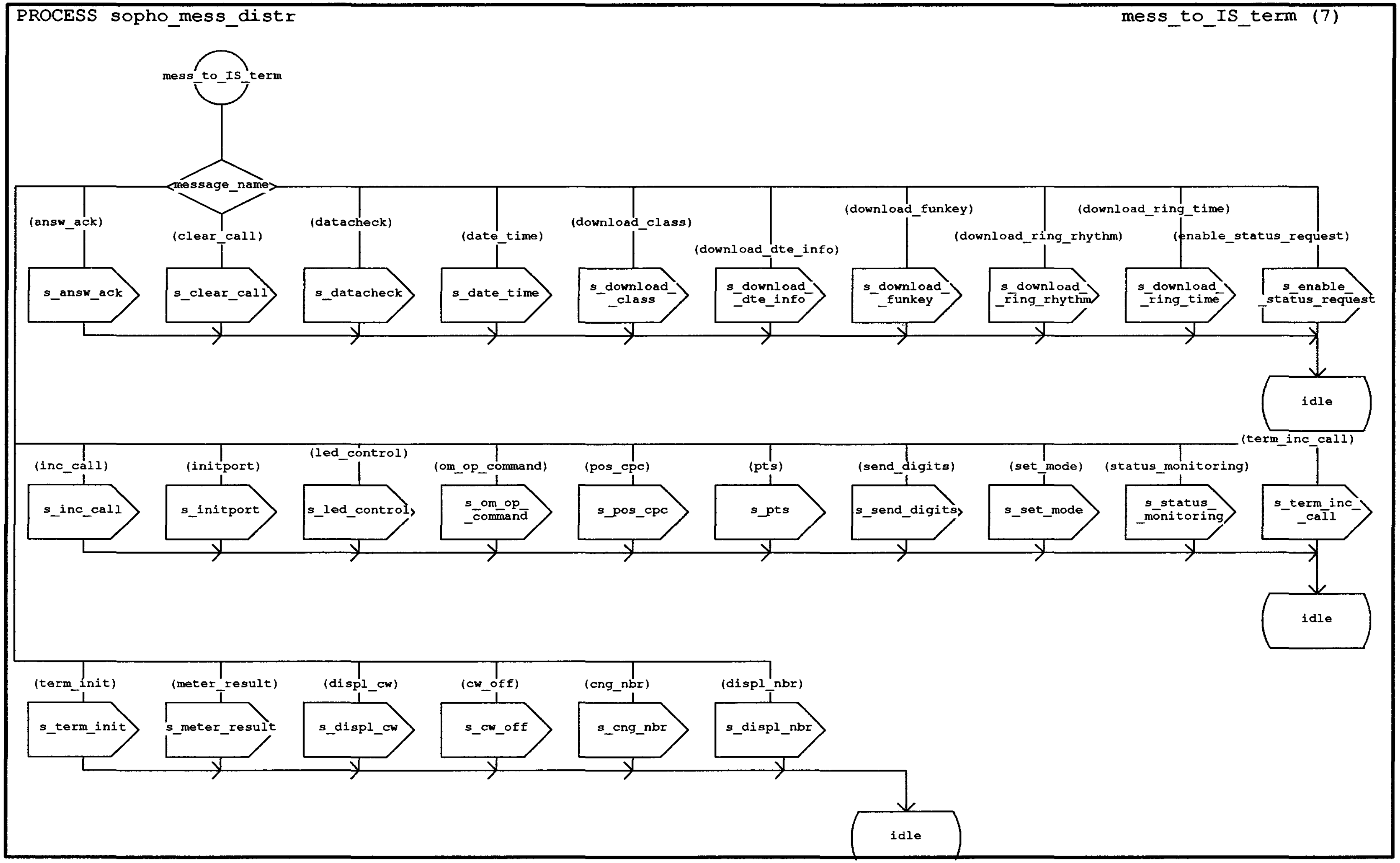
```
DCL
message_name       name_type_1,
portnumber_found   boolean :=false,
IS_terminal        boolean :=false,
cr                 integer,
ch,chan            chan_no,
inst               pid,
portnumber         portnumber_type,
IS_term            IS_term_type,
IDP_id             pid_type,
PCP_id             pid_type,
active_IDP         flag_active_IDP_type,
cause              cause_type;
```

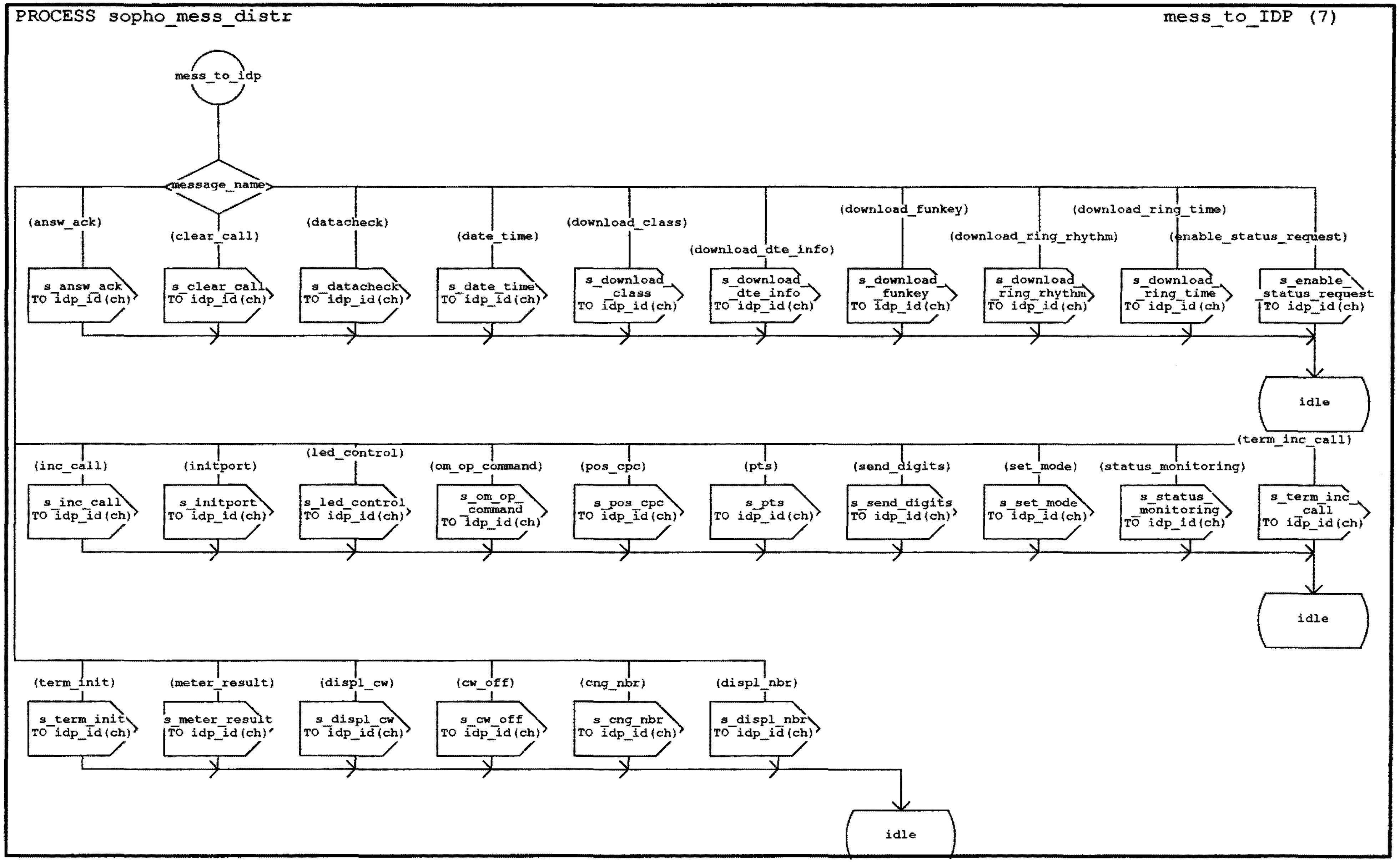

PROCESS sopho_mess_distr
idle/1/I (7)
I1
IS_term(ch)
(true)
(false)
Does IS-terminal occupy this portnumber?
active_IDP (ch)
Does an IDP instance occupy this portnumber?
pcp_id(ch) =null
Does an PCP instance occupy this portnumber?
(true)
(false)
protocol_control (cr,127,idle)
choose free CR, TEI=127, start_state=idle
pcp_id(ch):= OFFSPRING
mess_to_IS_term
mess_to_idp
s_mess_to_pcp
s_mess_to_pcp
I2
message_name= initport
(false)
(true)
discard message
'determine_channelnumber_ B0 or B1'
IS_terminal
Does IS_terminal want this B-channel? Invoke layer_2 TEI-check procedure
(true)
(false)
init_and_downl (ch)
IS_term(ch):= true
idp_id(ch):= OFFSPRING,
active_idp(ch) :=true
idle
mess_to_idp
mess_to_IS_term

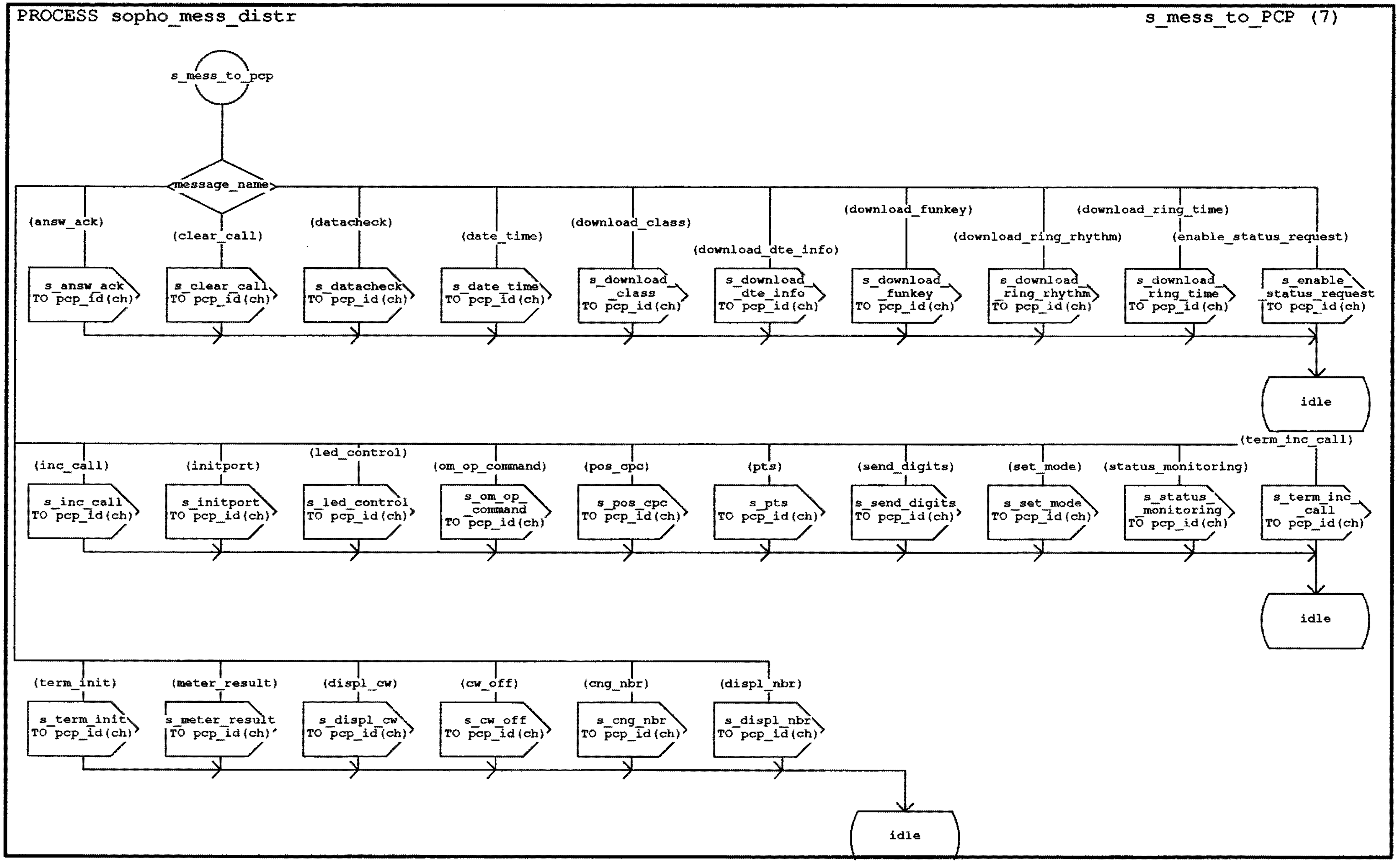

PROCESS sopho_mess_distr
idle/2 (7)
idle
chan_request (chan)
idp_id (chan) =null
channel not yet initialized?
(true)
(false)
active_idp (chan)= true
idp is active on this channel?
(true)
(false)
pcp_id (chan) =null
channel free?
(false)
(true)
cause:=chan_ _occupied
cause:=chan_ _not_yet_ _initialized
B_chan_not_ _assigned (cause) TO SENDER
idle
I3
rel_chan (chan)
pcp_id(chan) = SENDER
(true)
(false)
pcp_id(chan) :=null
illegal attempt to release a channel!
idle
chan_change (chan,inst)
pcp_id(chan) = SENDER
(true)
(false)
pcp_id(chan) :=inst
illegal attempt to assign a channel
idp_finished (chan)
active_idp (chan):= false
idp_started (chan)
active_idp (chan):= true

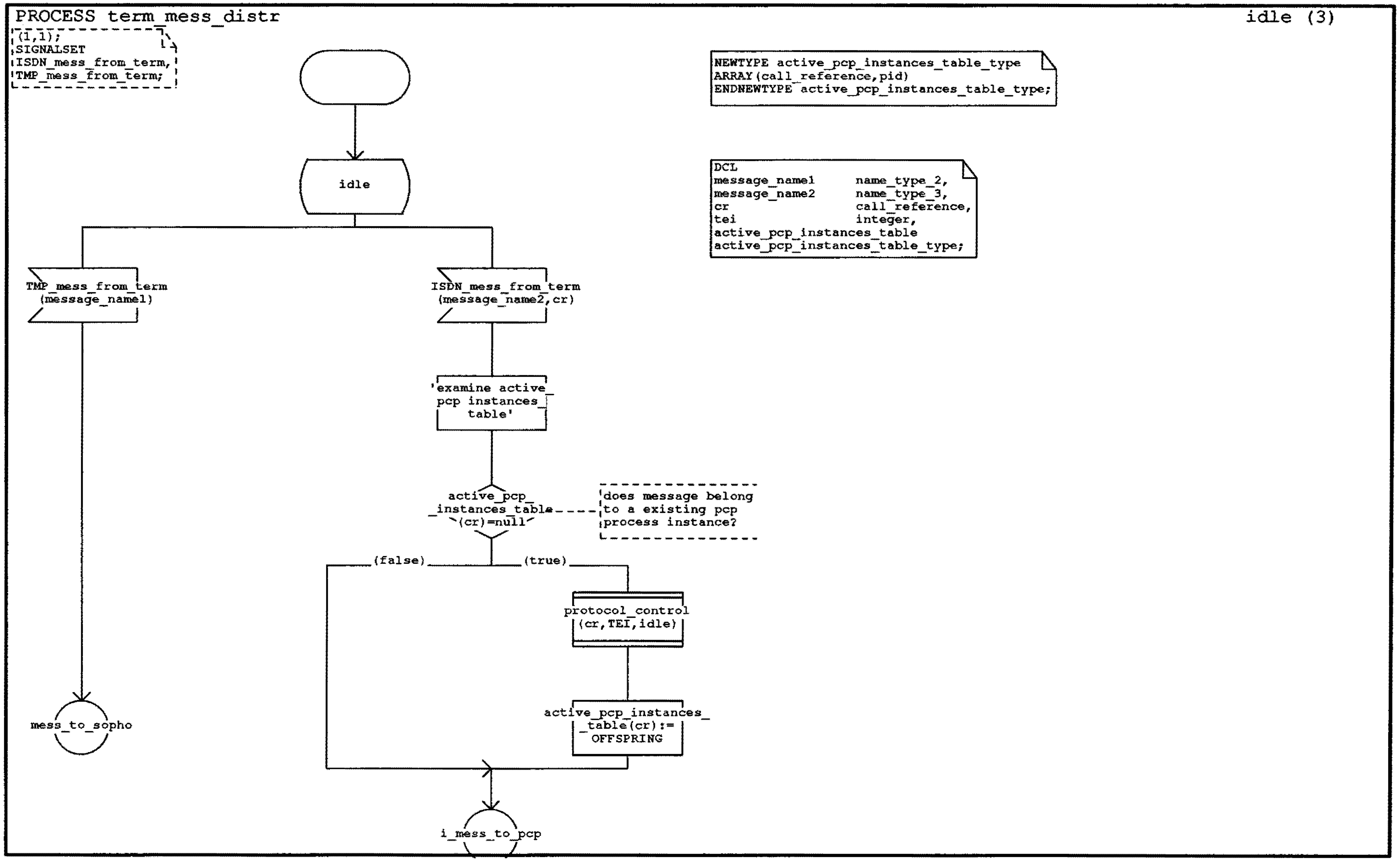
PROCESS sopho_mess_distr idle/2/I (7)

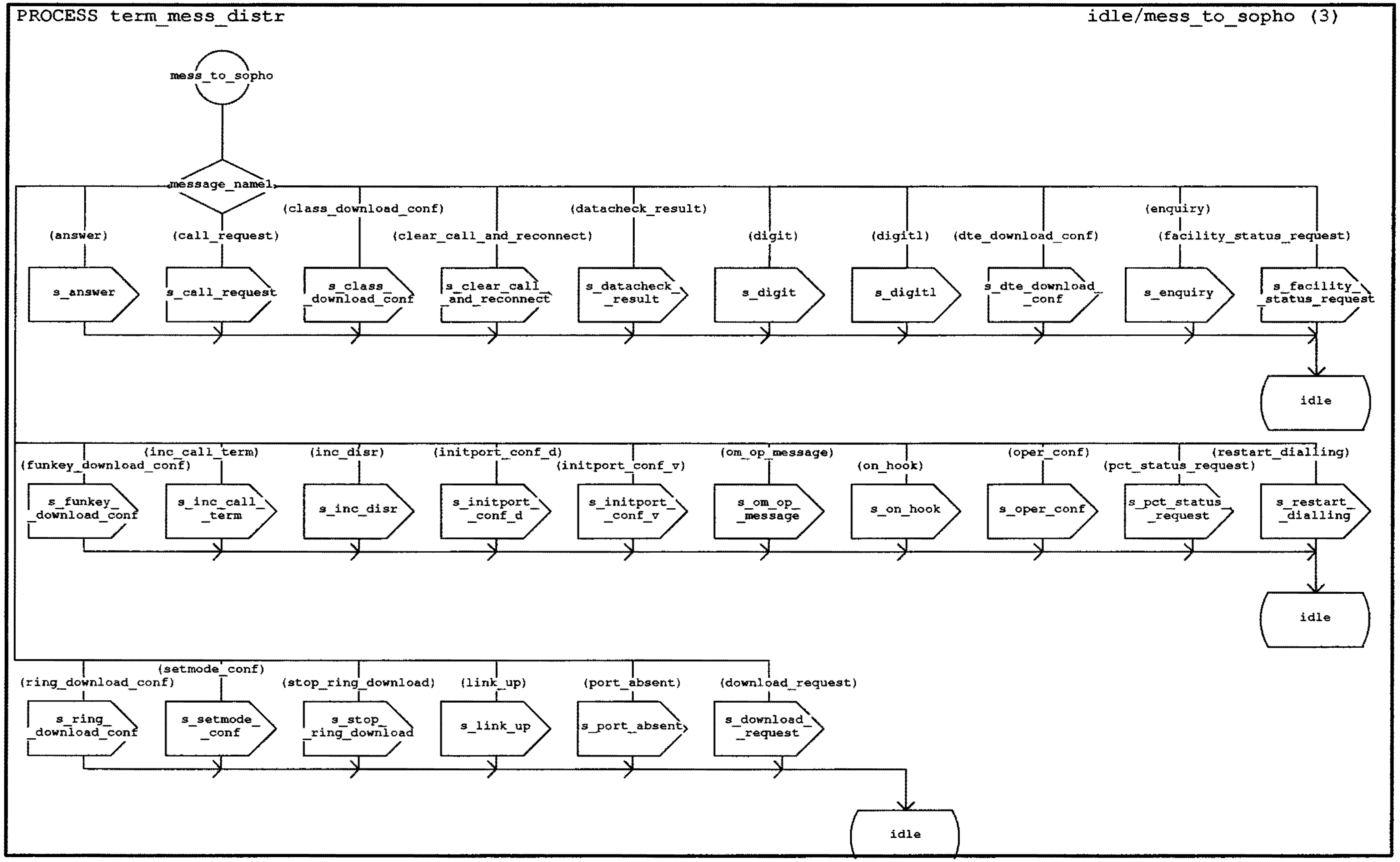


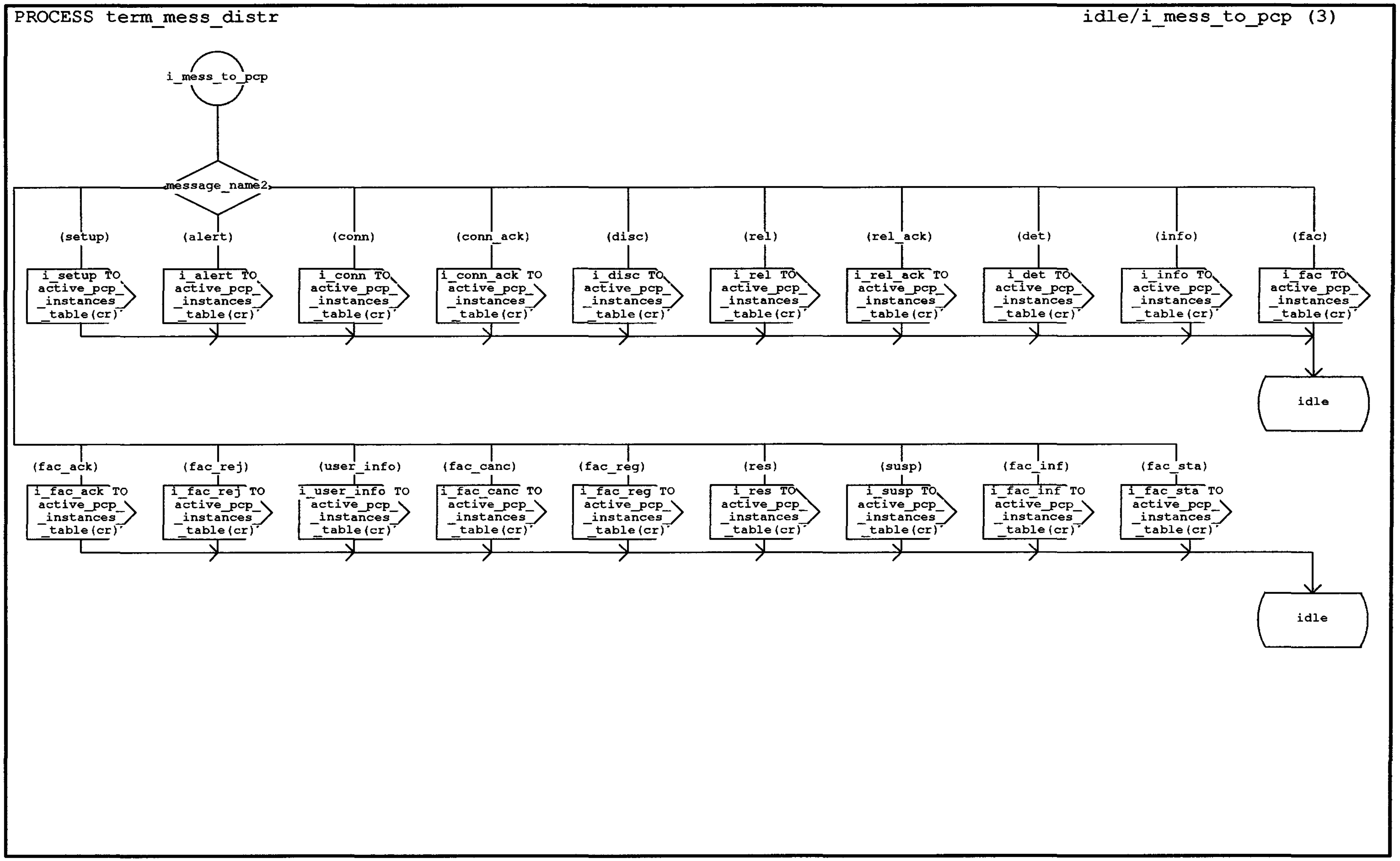

PROCESS sopho_mess_distr
mess_to_IS_term (7)
mess_to_IS_term
message_name
(answ_ack)
s_answ_ack
(clear_call)
s_clear_call
(datacheck)
s_datacheck
(date_time)
s_date_time
(download_class)
s_download_ _class
(download_dte_info)
s_download_ _dte_info
(download_funkey)
s_download_ _funkey
(download_ring_rhythm)
s_download_ _ring_rhythm
(download_ring_time)
s_download_ _ring_time
(enable_status_request)
s_enable_ _status_request
idle
(inc_call)
s_inc_call
(initport)
s_initport
(led_control)
s_led_control
(om_op_command)
s_om_op_ _command
(pos_cpc)
s_pos_cpc
(pts)
s_pts
(send_digits)
s_send_digits
(set_mode)
s_set_mode
(status_monitoring)
s_status_ _monitoring
(term_inc_call)
s_term_inc_ _call
idle
(term_init)
s_term_init
(meter_result)
s_meter_result
(displ_cw)
s_displ_cw
(cw_off)
s_cw_off
(cng_nbr)
s_cng_nbr
(displ_nbr)
s_displ_nbr
idle

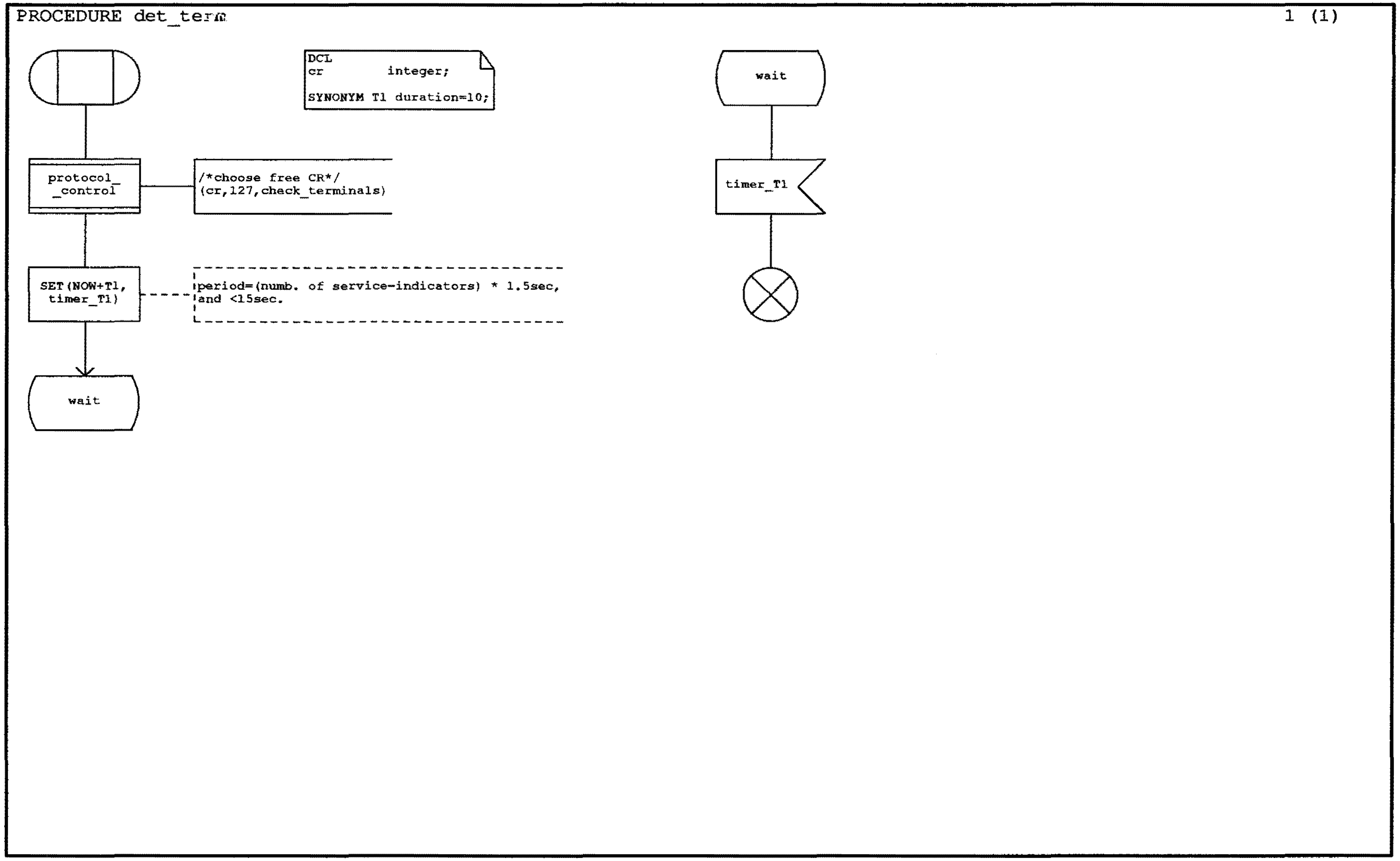

PROCESS sopho_mess_distr
mess_to_IDP (7)
mess_to_idp
message_name
(answ_ack)
s_answ_ack TO idp_id(ch)
(clear_call)
s_clear_call TO idp_id(ch)
(datacheck)
s_datacheck TO idp_id(ch)
(date_time)
s_date_time TO idp_id(ch)
(download_class)
s_download_class TO idp_id(ch)
(download_dte_info)
s_download_dte_info TO idp_id(ch)
(download_funkey)
s_download_funkey TO idp_id(ch)
(download_ring_rhythm)
s_download_ring_rhythm TO idp_id(ch)
(download_ring_time)
s_download_ring_time TO idp_id(ch)
(enable_status_request)
s_enable_status_request TO idp_id(ch)
idle
(inc_call)
s_inc_call TO idp_id(ch)
(initport)
s_initport TO idp_id(ch)
(led_control)
s_led_control TO idp_id(ch)
(om_op_command)
s_om_op_command TO idp_id(ch)
(pos_cpc)
s_pos_cpc TO idp_id(ch)
(pts)
s_pts TO idp_id(ch)
(send_digits)
s_send_digits TO idp_id(ch)
(set_mode)
s_set_mode TO idp_id(ch)
(status_monitoring)
s_status_monitoring TO idp_id(ch)
(term_inc_call)
s_term_inc_call TO idp_id(ch)
idle
(term_init)
s_term_init TO idp_id(ch)
(meter_result)
s_meter_result TO idp_id(ch)
(displ_cw)
s_displ_cw TO idp_id(ch)
(cw_off)
s_cw_off TO idp_id(ch)
(cng_nbr)
s_cng_nbr TO idp_id(ch)
(displ_nbr)
s_displ_nbr TO idp_id(ch)
idle


PROCESS sopho_mess_distr
s_mess_to_PCP (7)
s_mess_to_pcp
message_name
(answ_ack)
s_answ_ack TO pcp_id(ch)
(clear_call)
s_clear_call TO pcp_id(ch)
(datacheck)
s_datacheck TO pcp_id(ch)
(date_time)
s_date_time TO pcp_id(ch)
(download_class)
s_download_class TO pcp_id(ch)
(download_dte_info)
s_download_dte_info TO pcp_id(ch)
(download_funkey)
s_download_funkey TO pcp_id(ch)
(download_ring_rhythm)
s_download_ring_rhythm TO pcp_id(ch)
(download_ring_time)
s_download_ring_time TO pcp_id(ch)
(enable_status_request)
s_enable_status_request TO pcp_id(ch)
idle
(inc_call)
s_inc_call TO pcp_id(ch)
(initport)
s_initport TO pcp_id(ch)
(led_control)
s_led_control TO pcp_id(ch)
(om_op_command)
s_om_op_command TO pcp_id(ch)
(pos_cpc)
s_pos_cpc TO pcp_id(ch)
(pts)
s_pts TO pcp_id(ch)
(send_digits)
s_send_digits TO pcp_id(ch)
(set_mode)
s_set_mode TO pcp_id(ch)
(status_monitoring)
s_status_monitoring TO pcp_id(ch)
(term_inc_call)
s_term_inc_call TO pcp_id(ch)
idle
(term_init)
s_term_init TO pcp_id(ch)
(meter_result)
s_meter_result TO pcp_id(ch)
(displ_cw)
s_displ_cw TO pcp_id(ch)
(cw_off)
s_cw_off TO pcp_id(ch)
(cng_nbr)
s_cng_nbr TO pcp_id(ch)
(displ_nbr)
s_displ_nbr TO pcp_id(ch)
idle


PROCESS term_mess_distr
idle (3)
(1,1);
SIGNALSET
ISDN_mess_from_term,
TMP_mess_from_term;
NEWTYPE active_pcp_instances_table_type
ARRAY(call_reference,pid)
ENDNEWTYPE active_pcp_instances_table_type;
DCL
message_name1 name_type_2,
message_name2 name_type_3,
cr call_reference,
tei integer,
active_pcp_instances_table
active_pcp_instances_table_type;
idle
TMP_mess_from_term
(message_name1)
ISDN_mess_from_term
(message_name2,cr)
'examine active_
pcp instances_
table'
active_pcp_
_instances_table
(cr)=null
does message belong
to a existing pcp
process instance?
(false)
(true)
protocol_control
(cr,TEI,idle)
active_pcp_instances_
_table(cr):=
OFFSPRING
mess_to_sopho
i_mess_to_pcp


PROCESS term_mess_distr
idle/mess_to_sopho (3)
mess_to_sopho
message_name1
(answer)
s_answer
(call_request)
s_call_request
(class_download_conf)
s_class_download_conf
(clear_call_and_reconnect)
s_clear_call_and_reconnect
(datacheck_result)
s_datacheck_result
(digit)
s_digit
(digit1)
s_digit1
(dte_download_conf)
s_dte_download_conf
(enquiry)
s_enquiry
(facility_status_request)
s_facility_status_request
idle
(funkey_download_conf)
s_funkey_download_conf
(inc_call_term)
s_inc_call_term
(inc_disr)
s_inc_disr
(initport_conf_d)
s_initport_conf_d
(initport_conf_v)
s_initport_conf_v
(om_op_message)
s_om_op_message
(on_hook)
s_on_hook
(oper_conf)
s_oper_conf
(pct_status_request)
s_pct_status_request
(restart_dialling)
s_restart_dialling
idle
(ring_download_conf)
s_ring_download_conf
(setmode_conf)
s_setmode_conf
(stop_ring_download)
s_stop_ring_download
(link_up)
s_link_up
(port_absent)
s_port_absent
(download_request)
s_download_request
idle


PROCESS term_mess_distr
idle/i_mess_to_pcp (3)
i_mess_to_pcp
message_name2
(setup)
i_setup TO active_pcp_ _instances _table(cr)
(alert)
i_alert TO active_pcp_ _instances _table(cr)
(conn)
i_conn TO active_pcp_ _instances _table(cr)
(conn_ack)
i_conn_ack TO active_pcp_ _instances _table(cr)
(disc)
i_disc TO active_pcp_ _instances _table(cr)
(rel)
i_rel TO active_pcp_ _instances _table(cr)
(rel_ack)
i_rel_ack TO active_pcp_ _instances _table(cr)
(det)
i_det TO active_pcp_ _instances _table(cr)
(info)
i_info TO active_pcp_ _instances _table(cr)
(fac)
i_fac TO active_pcp_ _instances _table(cr)
idle
(fac_ack)
i_fac_ack TO active_pcp_ _instances _table(cr)
(fac_rej)
i_fac_rej TO active_pcp_ _instances _table(cr)
(user_info)
i_user_info TO active_pcp_ _instances _table(cr)
(fac_canc)
i_fac_canc TO active_pcp_ _instances _table(cr)
(fac_reg)
i_fac_reg TO active_pcp_ _instances _table(cr)
(res)
i_res TO active_pcp_ _instances _table(cr)
(susp)
i_susp TO active_pcp_ _instances _table(cr)
(fac_inf)
i_fac_inf TO active_pcp_ _instances _table(cr)
(fac_sta)
i_fac_sta TO active_pcp_ _instances _table(cr)
idle


PROCEDURE det_term
1 (1)
DCL
cr integer;
SYNONYM T1 duration=10;
protocol_
_control
/*choose free CR*/
(cr,127,check_terminals)
SET(NOW+T1,
timer_T1)
period=(numb. of service-indicators) * 1.5sec,
and <15sec.
wait
wait
timer_T1

APPENDIX B

SIMULATION OF A BASIC CALL

In this appendix the trace file of the simulation example of section 4.5.2 is printed. In the trace list the moments on which the monitor commands are given are inserted as comments written in this letter type. The trace level is set to 6 (the maximum). The simulator generator SDT SAC Version R4B is used.

```
*** TRANSITION START
*      PID     : TERM_MESS_DISTR:1
*      STATE   : START STATE
*      NOW     : 0.00
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID     : SOPHO_MESS_DISTR:1
*      STATE   : START STATE
*      NOW     : 0.00
*   TASK
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID     : SOPHO_MESS_DISTR:1
*      STATE   : IDLE
*      INPUT   : TMP_MESS_FROM_SOPHO
*      SENDER  : ENV:1
*      NOW     : 0.00
*      PARAMETER(S) : INITPORT, 0
*   TASK
*   DECISION
*   DECISION
*   TASK
*   DECISION
*   CREATE INIT_AND_DOWNL:1
*      PARAMETER(S) : 0
*   TASK
*   DECISION
*   OUTPUT OF S_INITPORT TO INIT_AND_DOWNL:1
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID     : INIT_AND_DOWNL:1
*      STATE   : START STATE
*      NOW     : 0.00
*      PARAMETER(S) : 0
*** NEXTSTATE POWERED_UP

*** TRANSITION START
*      PID     : INIT_AND_DOWNL:1
*      STATE   : POWERED_UP
*      INPUT   : S_INITPORT
*      SENDER  : SOPHO_MESS_DISTR:1
*      NOW    : 0.00
*   TASK
*   DECISION
*   TASK
*   OUTPUT OF S_INITPORT_CONF_V TO ENV:1
*** NEXTSTATE PRE_OPER
```

Here the situation Now=0.1 is reached!

```
*** TRANSITION START
*      PID    : TERM_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : ISDN_MESS_FROM_TERM
*      SENDER : ENV:1
*      NOW    : 0.10
*      PARAMETER(S) : SETUP, 0
*   TASK
*   DECISION
*   CREATE PROTOCOL_CONTROL:1
*      PARAMETER(S) : 0, 0, IDLE
*   TASK
*   DECISION
*   OUTPUT OF I_SETUP TO PROTOCOL_CONTROL:1
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:1
*      STATE  : START STATE
*      NOW    : 0.10
*      PARAMETER(S) : 0, 0, IDLE
*   TASK
*   DECISION
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:1
*      STATE  : IDLE
*      INPUT  : I_SETUP
*      SENDER : TERM_MESS_DISTR:1
*      NOW    : 0.10
*   DECISION
*   OUTPUT OF CHAN_REQUEST TO SOPHO_MESS_DISTR:1
*      PARAMETER(S) : 0
```

```
*** NEXTSTATE WAIT

*** TRANSITION START
*      PID    : SOPHO_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : CHAN_REQUEST
*      SENDER : PROTOCOL_CONTROL:1
*      NOW    : 0.10
*      PARAMETER(S) : 0
*   DECISION
*   DECISION
*   TASK
*   OUTPUT OF B_CHAN_NOT_ASSIGNED TO PROTOCOL_CONTROL:1
*      PARAMETER(S) : CHAN_OCCUPIED
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:1
*      STATE  : WAIT
*      INPUT  : B_CHAN_NOT_ASSIGNED
*      SENDER : SOPHO_MESS_DISTR:1
*      NOW    : 0.10
*      PARAMETER(S) : CHAN_OCCUPIED
*   DECISION
*   OUTPUT OF I_DISC TO ENV:1
*   SET ON TIMER DISC_TIMER AT 120.10
*** NEXTSTATE DISCONNECT_INDICATION

Here the situation Now=0.2 is reached!


*** TRANSITION START
*      PID    : SOPHO_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : TMP_MESS_FROM_SOPHO
*      SENDER : ENV:1
*      NOW    : 0.20
*      PARAMETER(S) : TERM_INIT, 0
*   TASK
*   DECISION
*   DECISION
*   DECISION
*   DECISION
*   OUTPUT OF S_TERM_INIT TO INIT_AND_DOWNL:1
```

```
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : INIT_AND_DOWNL:1
*      STATE  : PRE_OPER
*      INPUT  : S_TERM_INIT
*      SENDER : SOPHO_MESS_DISTR:1
*      NOW    : 0.20
*   OUTPUT OF S_OPER_CONF TO ENV:1
*   OUTPUT OF IDP_FINISHED TO SOPHO_MESS_DISTR:1
*      PARAMETER(S) : 0
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : SOPHO_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : IDP_FINISHED
*      SENDER : INIT_AND_DOWNL:1
*      NOW    : 0.20
*      PARAMETER(S) : 0
*   TASK
*** NEXTSTATE IDLE

Here the situation Now=0.3 is reached!


*** TRANSITION START
*      PID    : SOPHO_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : TMP_MESS_FROM_SOPHO
*      SENDER : ENV:1
*      NOW    : 0.30
*      PARAMETER(S) : INC_CALL, 0
*   TASK
*   DECISION
*   DECISION
*   DECISION
*   DECISION
*   CREATE PROTOCOL_CONTROL:2
*      PARAMETER(S) : 0, 127, IDLE
*   TASK
*   DECISION
*   OUTPUT OF S_INC_CALL TO PROTOCOL_CONTROL:2
*** NEXTSTATE IDLE
*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:2
*      STATE  : START STATE
*      NOW    : 0.30
*      PARAMETER(S) : 0, 127, IDLE
*   TASK
*   DECISION
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:2
*      STATE  : IDLE
*      INPUT  : S_INC_CALL
*      SENDER : SOPHO_MESS_DISTR:1
*      NOW    : 0.30
*   OUTPUT OF I_SETUP TO ENV:1
*   SET ON TIMER SETUP_TIMER AT 1.80
*   TASK
*** NEXTSTATE CALL_PRESENT


Here the situation Now=0.4 is reached!


*** TRANSITION START
*      PID    : TERM_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : ISDN_MESS_FROM_TERM
*      SENDER : ENV:1
*      NOW    : 0.40
*      PARAMETER(S) : ALERT, 1
*   TASK
*   DECISION
*   DECISION
*   OUTPUT OF I_ALERT TO PROTOCOL_CONTROL:2
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:2
*      STATE  : CALL_PRESENT
*      INPUT  : I_ALERT
*      SENDER : TERM_MESS_DISTR:1
*      NOW    : 0.40
*   DECISION
*   TASK
*** NEXTSTATE CALL_RECEIVED
```

Here the situation Now=0.5 is reached!

```
*** TRANSITION START
*      PID    : TERM_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : ISDN_MESS_FROM_TERM
*      SENDER : ENV:1
*      NOW    : 0.50
*      PARAMETER(S) : CONN, 1
*   TASK
*   DECISION
*   DECISION
*   OUTPUT OF I_CONN TO PROTOCOL_CONTROL:2
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:2
*      STATE  : CALL_RECEIVED
*      INPUT  : I_CONN
*      SENDER : TERM_MESS_DISTR:1
*      NOW    : 0.50
*   DECISION
*   DECISION
*   CREATE PROTOCOL_CONTROL:3
*      PARAMETER(S) : 0, 127, CONNECT_REQUESTED
*   OUTPUT OF CHAN_CHANGE TO SOPHO_MESS_DISTR:1
*      PARAMETER(S) : 0, PROTOCOL_CONTROL:3
*   DECISION
*   TASK
*   DECISION
*** STOP   (NO SIGNALS WERE DISCARDED)
*   TIMER OF TYPE SETUP_TIMER WAS REMOVED

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:3
*      STATE  : START STATE
*      NOW    : 0.50
*      PARAMETER(S) : 0, 127, CONNECT_REQUESTED
*   TASK
*   DECISION
*   OUTPUT OF S_ANSWER TO ENV:1
*** NEXTSTATE CONNECT_REQUESTED

*** TRANSITION START
*      PID    : SOPHO_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : CHAN_CHANGE
*      SENDER : PROTOCOL_CONTROL:2+
*      NOW    : 0.50
*      PARAMETER(S) : 0, PROTOCOL_CONTROL:3
*   DECISION
*   TASK
*** NEXTSTATE IDLE
```

Here the situation Now=0.6 is reached!

```
*** TRANSITION START
*      PID    : SOPHO_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : TMP_MESS_FROM_SOPHO
*      SENDER : ENV:1
*      NOW    : 0.60
*      PARAMETER(S) : ANSW_ACK, 0
*   TASK
*   DECISION
*   DECISION
*   DECISION
*   DECISION
*   DECISION
*   OUTPUT OF S_ANSW_ACK TO PROTOCOL_CONTROL:3
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:3
*      STATE  : CONNECT_REQUESTED
*      INPUT  : S_ANSW_ACK
*      SENDER : SOPHO_MESS_DISTR:1
*      NOW    : 0.60
*   OUTPUT OF I_CONN_ACK TO ENV:1
*** NEXTSTATE CONNECTED
```

Here the situation Now=0.7 is reached!

```
*** TRANSITION START
*      PID    : TERM_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : ISDN_MESS_FROM_TERM
*      SENDER : ENV:1
*      NOW    : 0.70
*      PARAMETER(S) : REL, 1
*   TASK
*   DECISION
*   DECISION
*   OUTPUT OF I_REL TO PROTOCOL_CONTROL:3
*** NEXTSTATE IDLE

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:3
*      STATE  : CONNECTED
*      INPUT  : I_REL
*      SENDER : TERM_MESS_DISTR:1
*      NOW    : 0.70
*   OUTPUT OF S_ON_HOOK TO ENV:1
*   OUTPUT OF REL_CHAN TO SOPHO_MESS_DISTR:1
*      PARAMETER(S) : 0
*   OUTPUT OF I_REL_ACK TO ENV:1
*   TASK
*   DECISION
*** STOP   (NO SIGNALS WERE DISCARDED)

*** TRANSITION START
*      PID    : SOPHO_MESS_DISTR:1
*      STATE  : IDLE
*      INPUT  : REL_CHAN
*      SENDER : PROTOCOL_CONTROL:3+
*      NOW    : 0.70
*      PARAMETER(S) : 0
*   DECISION
*   TASK
*** NEXTSTATE IDLE

*** TIMER SIGNAL WAS SENT
*   TIMER    : DISC_TIMER
*   RECEIVER : PROTOCOL_CONTROL:1
*** NOW      : 120.10

*** TRANSITION START
```

```
*      PID    : PROTOCOL_CONTROL:1
*      STATE  : DISCONNECT_INDICATION
*      INPUT  : DISC_TIMER
*      SENDER : PROTOCOL_CONTROL:1
*      NOW    : 120.10
*   OUTPUT OF I_REL TO ENV:1
*   SET ON TIMER REL_TIMER AT 124.10
*** NEXTSTATE RELEASE_REQUEST

*** TIMER SIGNAL WAS SENT
*   TIMER     : REL_TIMER
*   RECEIVER  : PROTOCOL_CONTROL:1
*** NOW       : 124.10

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:1
*      STATE  : RELEASE_REQUEST
*      INPUT  : REL_TIMER
*      SENDER : PROTOCOL_CONTROL:1
*      NOW    : 124.10
*   DECISION
*   TASK
*   OUTPUT OF I_REL TO ENV:1
*   SET ON TIMER REL_TIMER AT 128.10
*** NEXTSTATE RELEASE_REQUEST

*** TIMER SIGNAL WAS SENT
*   TIMER     : REL_TIMER
*   RECEIVER  : PROTOCOL_CONTROL:1
*** NOW       : 128.10

*** TRANSITION START
*      PID    : PROTOCOL_CONTROL:1
*      STATE  : RELEASE_REQUEST
*      INPUT  : REL_TIMER
*      SENDER : PROTOCOL_CONTROL:1
*      NOW    : 128.10
*   DECISION
*   DECISION
*   TASK
*   DECISION
*** STOP   (NO SIGNALS WERE DISCARDED)

 MONITOR SYSTEM   (NOW = 128.10)
```