TR 101 034 V1.1.1 (1997-05)

Technical Report

Methods for Testing and Specification (MTS); Reports on experiments in validation methodology; PISN Cordless Terminal Mobility Incoming Call Additional Network Feature (ANF-CTMI)

Reference
DTR/MTS-00030-2 (9ko00ics.PDF)

Keywords
SDL, validation, PISN, QSIG, CTM

ETSI Secretariat

Postal address
F-06921 Sophia Antipolis Cedex - FRANCE

Office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

X.400
c= fr; a=atlas; p=etsi; s=secretariat

Internet
secretariat@etsi.fr
http://www.etsi.fr

Contents

Intellectual Property Rights

ETSI has not been informed of the existence of any Intellectual Property Right (IPR) which could be, or could become essential to the present document. However, pursuant to the ETSI Interim IPR Policy, no investigation, including IPR searches, has been carried out. No guarantee can be given as to the existence of any IPRs which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

As part of the research necessary prior to the development of the SDL Validation Methodology Handbook (EG 201 015), experiments were undertaken to validate three quite different protocol standards. The results of these experiments are (will be) available in the following Technical Reports:

Part 1: Validation of the GSM CCBS supplementary service (MTS-TR 004);

Part 2: Validation of the PISN Cordless Terminal Incoming Call Additional Network Feature;

Part 3: Validation of the Core INAP Capability Set 2 (CS2) protocol (DTR/MTS-00030-3, expected by end 1998).

1 Scope

As part of the preparation for writing an SDL validation methodology handbook, PT65 carried out an experimental validation of the PISN Cordless Terminal Incoming Call Additional Network Feature (ETS 300 696 [1]) supplementary service. This document describes the validation process and results. The work was divided into two different stages:

- creating a validation model (formalize the SDL diagram); and
- validating this model.

The objectives of the experiment were as follows:

- to demonstrate that a validation model can be constructed for a supplementary service where the basic service is not specified in the original informal SDL;
- to produce formal SDL that can be validated by automated tools but which is also understandable by readers who are not experts in SDL.

2 References

References may be made to:

a) specific versions of publications (identified by date of publication, edition number, version number, etc.), in which case, subsequent revisions to the referenced document do not apply; or

b) all versions up to and including the identified version (identified by "up to and including" before the version identity); or

c) all versions subsequent to and including the identified version (identified by "onwards" following the version identity); or

d) publications without mention of a specific version, in which case the latest version applies.

A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] ETS 300 696 (1996): "Private Integrated Services Network (PISN); Inter-exchange signalling protocol; Cordless Terminal Incoming Call Additional Network Feature (ANF-CTMI)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following definitions apply:

validation: The process by which appropriate methods, procedures and tools are used to evaluate that a standard:

- satisfies the purpose expressed in the record of requirements on which the standard is based;
- can be fully implemented;
- when implemented is able to offer all the functionality and performance expressed in the record of requirements on which the standard is based;
- conforms to the established criteria for standards.

behaviour tree: A description of the behaviour of a system in the form of a tree. Nodes of the tree are different states of the system, with the root node representing the initial system state. Branches between nodes represent transitions between states caused by external or internal stimuli.

deadlock: A state of a system such that no progress is possible.

formal model: A system model that is expressed in a language such that its behaviour can be unambiguously interpreted.

semantics: The meaning of a specification that relates it to the real system it defines.

syntax: The form of a specification, the rules specifying the use of the elements of the specification language.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ANF	Additional Network Feature
ANF-CTMI	Cordless Terminal Incoming Call Additional Network Feature
ASN.1	Abstract Syntax Notation Number 1
CTM	Cordless Terminal Mobility
HDB	Home Data Base
MSC	Message Sequence Chart
PINX	Private Integrated Network eXchange
PISN	Private Integrated Services Network
SDL	Specification and Description Language
VDB	Visitor Data Base

4 The standard description

The ANF-CTMI standard describes the inter-exchange signalling (QSIG) to support the routing of an incoming call to a user of a cordless terminal. Although full details of the user will be maintained at a single Private Integrated Network eXchange (PINX), the Home PINX, it is possible that when the call arrives, the user may have roamed to the coverage area of a different PINX, the Visitor PINX. The incoming call must be detected as a call to a mobile user and then directed to the appropriate Visitor PINX using the most efficient route possible.

ETS 300 696 [1] uses normative text supplemented by informal SDL to describe the operation of ANF-CTMI. Message structures and operations are defined using Abstract Syntax Notation Number 1 (ASN.1).

5 The validation model

In order to validate the standard, it was necessary to build a validation model which expressed the requirements of the service using formal SDL. This has been done by interpreting the informal SDL, Message Sequence Chart (MSC), ASN.1 and text contained in ETS 300 696 [1].

5.1 Starting point

As with most ISDN and PISN supplementary service standards, the Stage 3 SDL for ANF-CTMI expresses tasks in a very informal way which is incomplete, is semantically incorrect and does not conform to the SDL syntax. As a result, the following issues needed to be resolved:

- At the system and block level:
  - there were no Block or System diagram;
  - there were no Channels or Signal routes defined;

- signals and their parameters were described using ASN.1;
- there were no Signallist definitions.

- At the process level:
  - processes did not begin with a "start" symbol;
  - the QSIG basic service is not modelled and so the process diagrams must be considered to be incomplete;
  - assignments, timer functions and conditions are expressed as informal text and are, consequently, syntactically incorrect.

5.2 Defining the validation model

5.2.1 The system model

The informal model in ETS 300 696 [1] assumes an external environment and four physical entities:

- the Rerouting PINX;
- the CTMI-detect PINX;
- the Home PINX;
- the Visitor PINX.

The originating caller and the cordless terminal itself are considered to exist in the external environment.

This overall architecture was maintained in the validation model (figure 1).

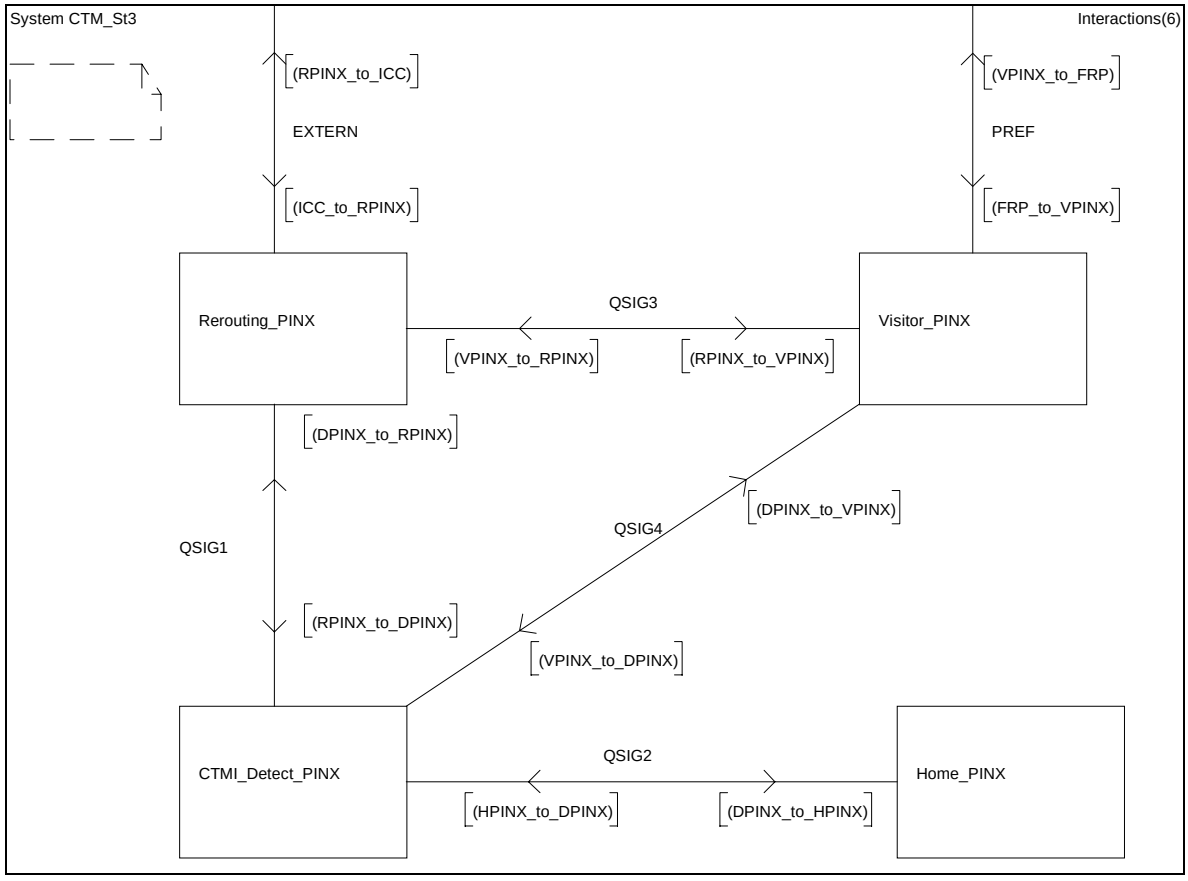

Figure 1: ANF-CTMI system

5.2.2 Specification of signals and data types

The ASN.1 specification of signals and data operations (figure 2) in ETS 300 696 [1] was used as the basis for the SDL signals, signallists and data type definitions (figure 3, figure 4 & figure 5).

```
CTM-Incoming-call-Operations { ccitt (0) identified-organization (3) etsi (0)
                qsig-ctm-incoming-call (696) ctmi-operations (0) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS         OPERATION, ERROR FROM Remote-Operation-Notation
                    { joint-iso-ccitt (2) remote-operations (4) notation (0) }
                Extension FROM Manufacturer-specific-service-extension-definition
                    { iso (1) standard (0)
                    pss1-generic-procedures (11582) msi-definition (0) }
                PSS1InformationElement FROM Generic-parameters-definition
                    { iso (1) standard (0)
                    pss1-generic-procedures (11582) pss1-generic-parameters (6) }
                Name FROM Name-Operations
                    { iso (1) standard (0)
                    pss1-name (13868) name-operations (0) }
                basicServiceNotProvided, notAvailable FROM General-Error-List
                    { ccitt (0) recommendation (0) q (17) 950 general-error-list (1) }
                Address, PartyNumber, PartySubaddress, PresentedNumberScreened FROM
                    Addressing-Data-Elements
                    { iso (1) standard (0) pss1-generic-procedures (11582)
                    addressing-data-elements (9) };

CtmiEnquiry     ::= OPERATION
                -- Sent from the CTMI-detect PINX to the Home PINX.
                    ARGUMENT    EnquiryArg
                    RESULT  EnquiryRes
                    ERRORS { invalidServedUserNr, locationNotKnown, notAvailable,
                        basicServiceNotProvided, unspecified }

CtmiDivert      ::= OPERATION
                -- Sent from the CTMI-detect PINX to the Rerouteing PINX.
                    ARGUMENT    DivertArg
                    RESULT  DummyRes
                    ERRORS { notAvailable, unspecified }

CtmiInform      ::= OPERATION
                -- Sent from the Rerouteing PINX to the Visitor PINX.
                    ARGUMENT    InformArg

EnquiryArg      ::= SEQUENCE { pisnNumber       PartyNumber,
                -- The PISN number of the CTM user
                        qSIGInfoElement     PSS1InformationElement,
                -- The basic call information elements Bearer capability, High layer
                -- compatibility, Low layer compatibility can be embedded in the
                -- qSIGInfoElement in accordance with subclause 6.5.2.1.
                        argExtension        CtmiExtension OPTIONAL }
```

Figure 2: ASN.1 operations from ETS 300 696

```
DivertArg          ::= SEQUENCE { visitPINX          PartyNumber,
                   -- The PISN number of the Visitor PINX,
                   -- always a Complete Number.
                         callingNumber          PresentedNumberScreened,
                         pisnNumber         PartyNumber,
                   -- The PISN number of the CTM user,
                   -- always a Complete Number.
                         qSIGInfoElement      PSS1InformationElement,
                   -- The basic call information elements Bearer capability,
                   -- High layer compatibility, Low layer compatibility, Progress indicator and
                   -- Party category can be embedded in the qSIGInfoElement in accordance
                   -- with subclause 6.5.2.1.
                         callingUserSub         [1] PartySubaddress OPTIONAL,
                         callingUserName [2] Name OPTIONAL,
                         ctmUserSub         [3] PartySubaddress OPTIONAL,
                         argExtension       CtmiExtension OPTIONAL }

InformArg          ::= SEQUENCE { pisnNumber      PartyNumber,
                   -- The PISN number of the CTM user,
                      -- always a Complete Number.
                         argExtension       CtmiExtension OPTIONAL }

EnquiryRes         ::= CHOICE  { currLocation     [1] CurrLocation,
                         cfuActivated  [2] CfuActivated }

CurrLocation       ::= SEQUENCE { visitPINX          PartyNumber,
                   -- The PISN number of the Visitor PINX,
                   -- always a Complete Number.
                         pisnNumber         PartyNumber,
                   -- The PISN number of the CTM user,
                   -- always a complete number.
                         argExtension       CtmiExtension OPTIONAL }

CfuActivated       ::= SEQUENCE { divToAddress    Address,
                         divOptions          SubscriptionOption,
                         ctmName       [1] Name OPTIONAL,
                         argExtension       CtmiExtension OPTIONAL }

SubscriptionOption  ::= ENUMERATED          { noNotification (0),
                               notificationWithoutDivertedToNr (1),
                               notificationWithDivertedToNr (2) }
DummyRes           ::= CHOICE  { null       NULL,
                         extension  [1] IMPLICIT Extension,
                         sequOfExtn    [2] IMPLICIT SEQUENCE OF Extension }

CtmiExtension      ::= CHOICE  { none      NULL,
                         extension  [4] IMPLICIT Extension,
                         sequOfExtn    [5] IMPLICIT SEQUENCE OF Extension }

ctmiEnquiry              CtmiEnquiry  ::= 54
ctmiDivert               CtmiDivert    ::= 55
ctmiInform               CtmiInform    ::= 56
notAuthorized            ERROR    ::= 1007
locationNotKnown            ERROR    ::= 1015
unspecified              Unspecified   ::= 1008
Unspecified        ::=      ERROR    PARAMETER Extension

END                -- of CTM-Incoming-call-Operations
```

Figure 2 (concluded): ASN.1 operations from ETS 300 696

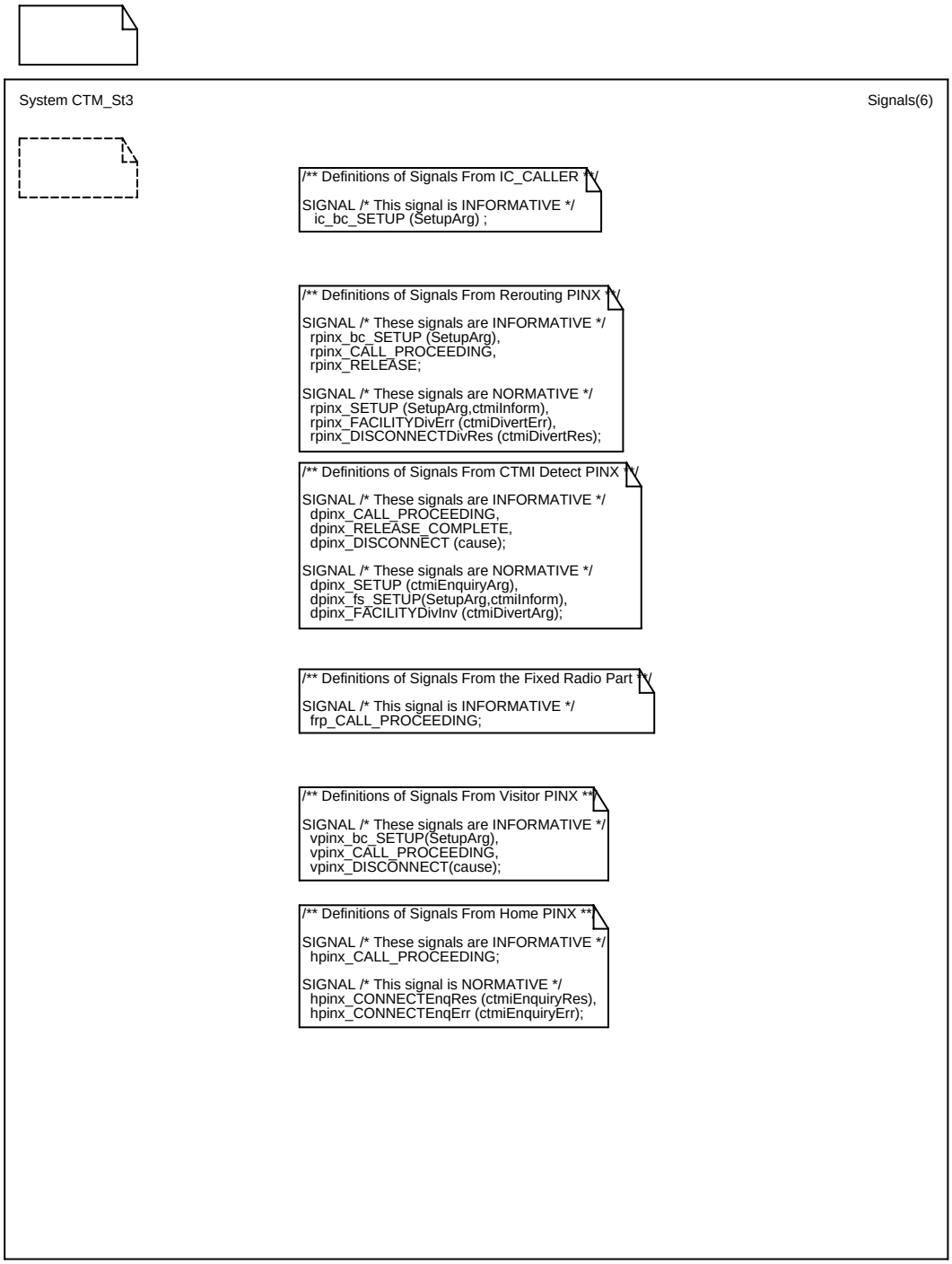

Figure 3: SDL signal definitions

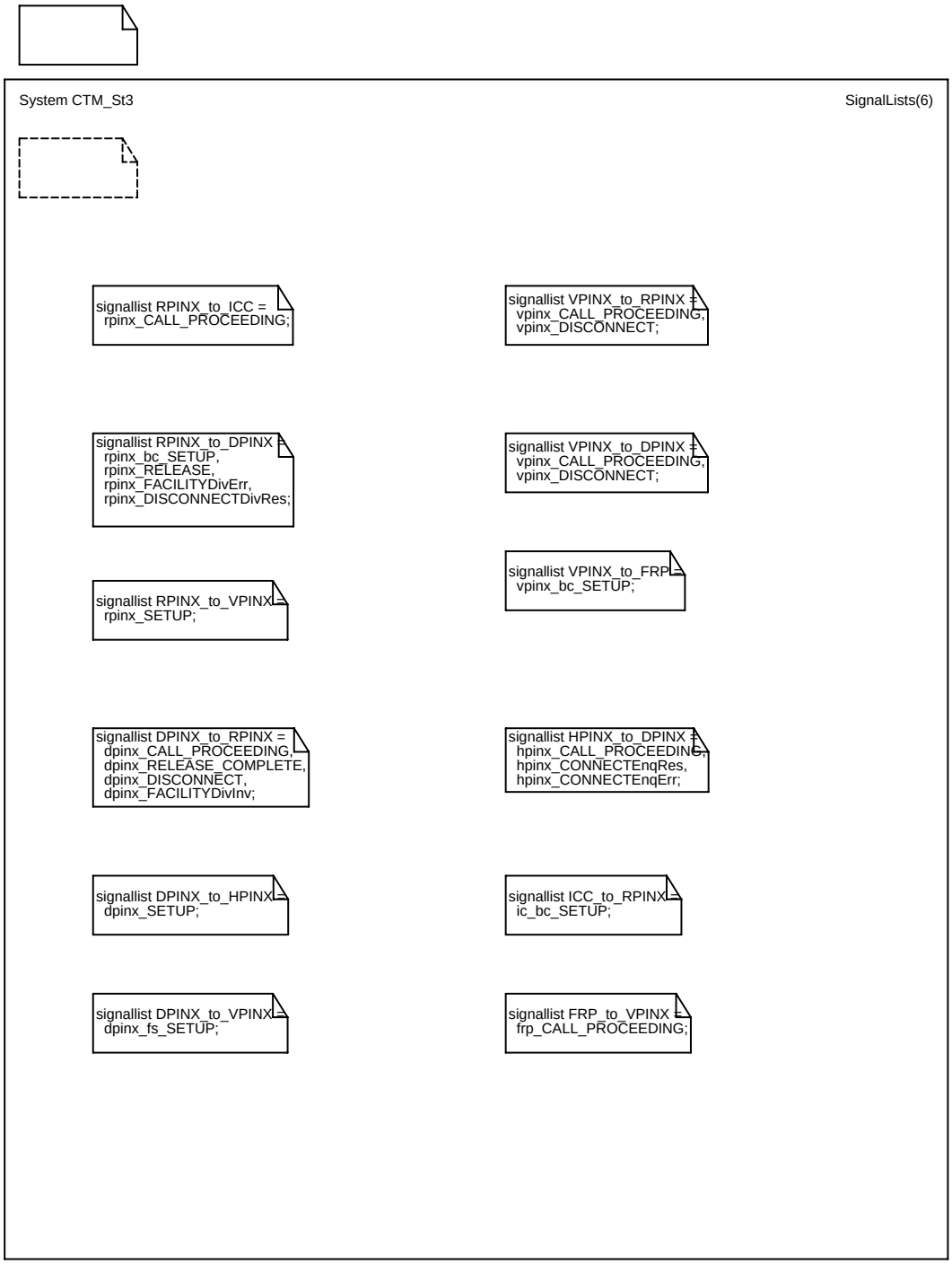

Figure 4: SDL signallist definitions

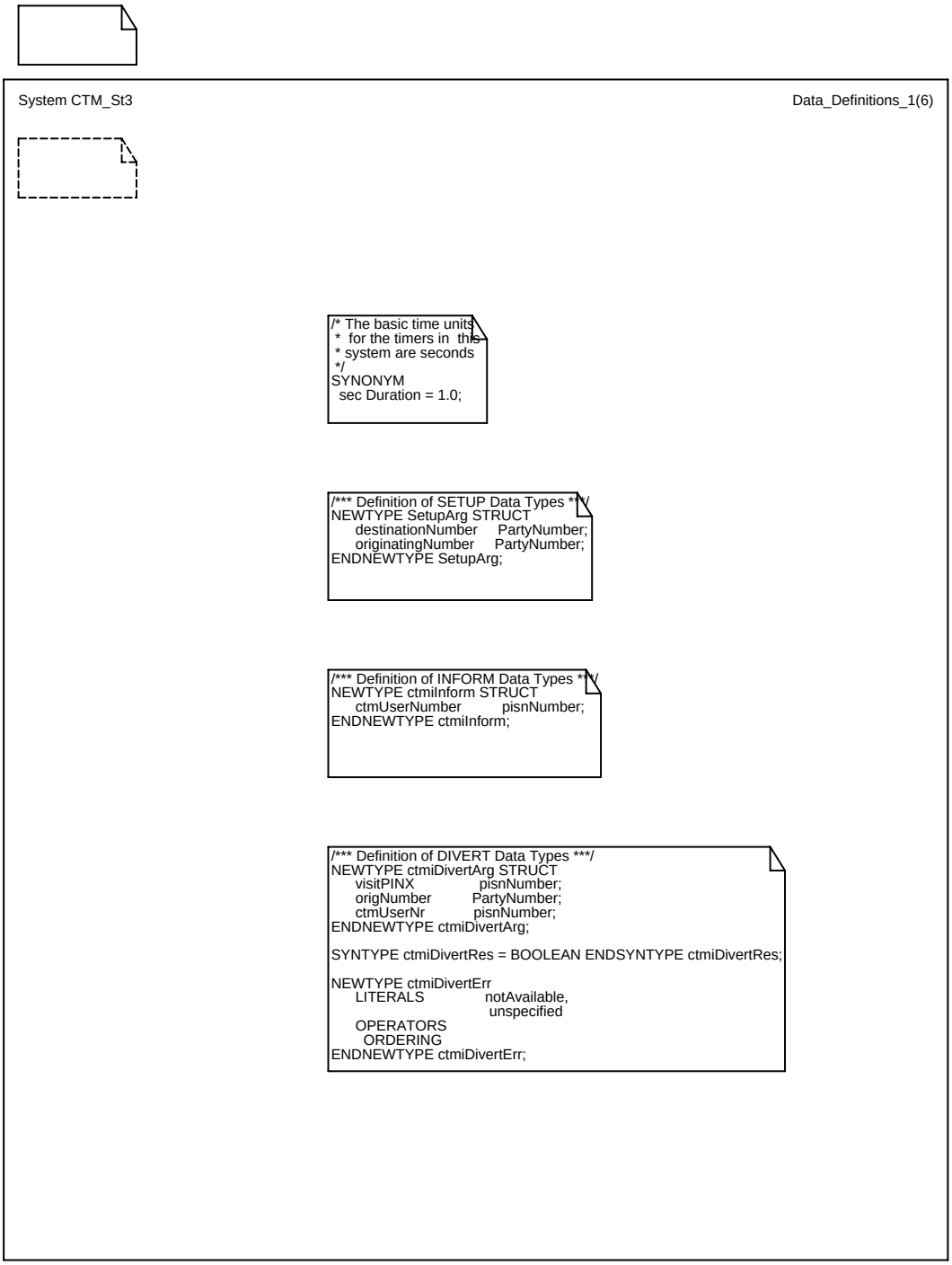

Figure 5: SDL data types

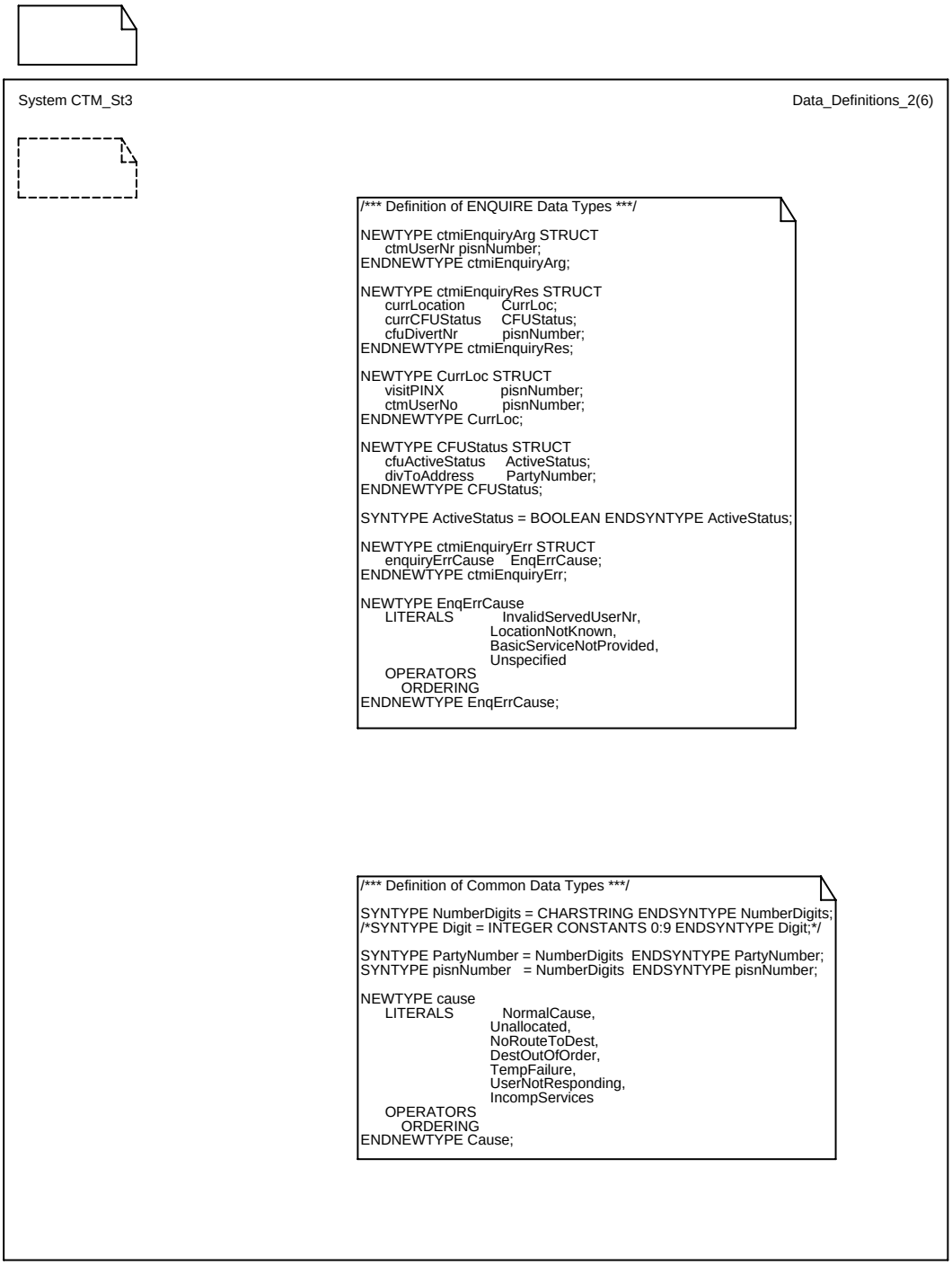

Figure 5 (continued): SDL data types

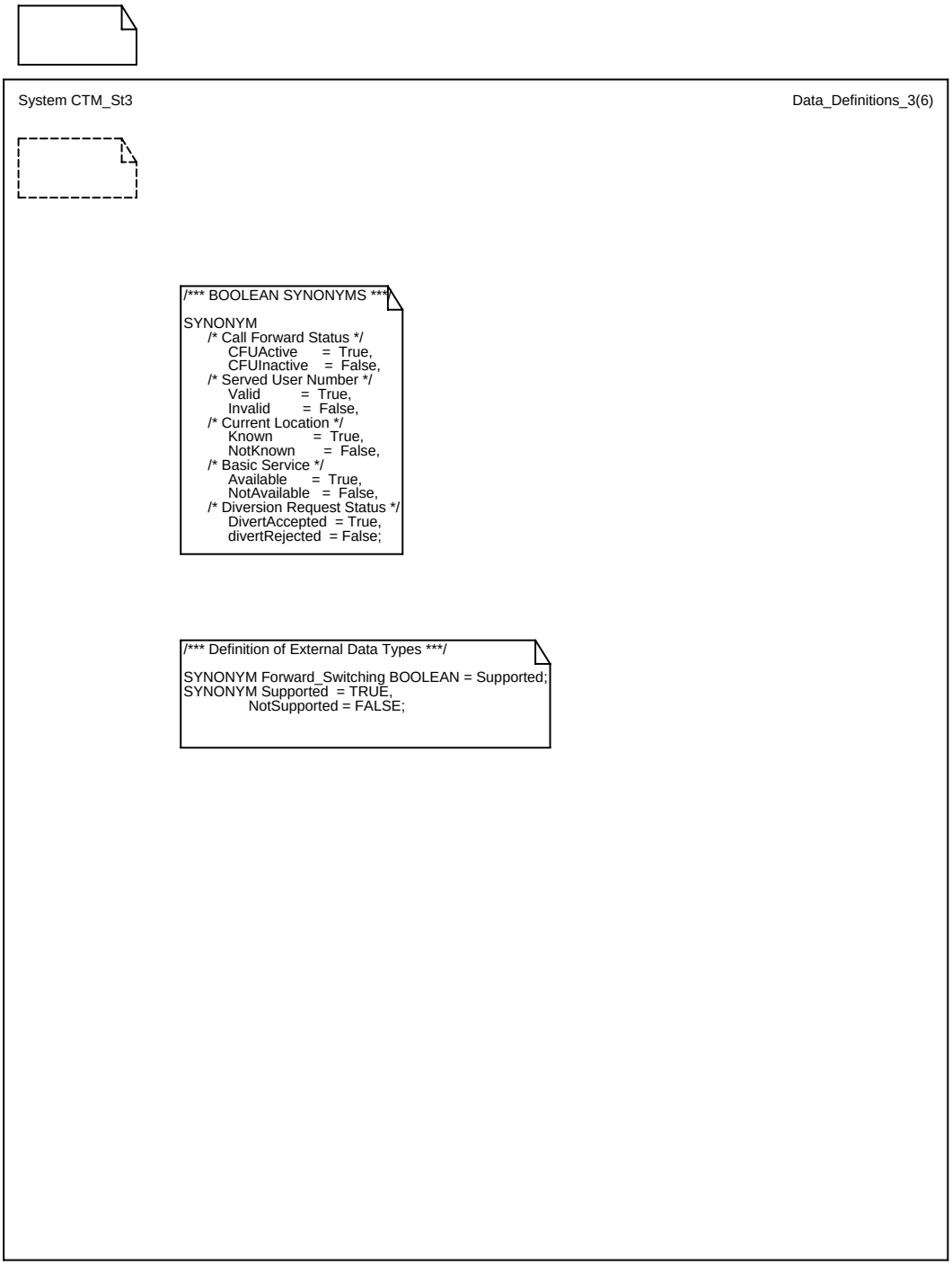

Figure 5 (concluded): SDL data types

5.3 Processes formalizing

As far as possible, the basic structure of the original informal SDL was kept in the formal model. The main changes to the SDL were:

- a start symbol was added to each process (see figure 7);
- local declaration of signals and parameters was included in a DCL statement in each process (see figure 7);
- conditional statements were re-expressed using the "ANY" construct or genuine BOOLEAN expressions (see figure 7);
- timer expressions were corrected as follows (see figure 7):
  - all timers were pre-set using a "TIMER T*x* := *nn**sec" statement;
  - "Start T*x*" ➔ SET (T*x*);
  - "Stop T*x*" ➔ RESET (T*x*);
  - "T*x* Expiry" ➔ T*x* /*Expiry*/;
- simplified QSIG basic service protocol was added to the model to replace expressions such as "Call cleared" (see figure 7);
- a block diagram was drawn for each process with names assigned to the signal routes. An example is shown in figure 6 All export statements were then written to include the "via" construct and a key to the route names was added as text to each page of each diagram (see figure 7).

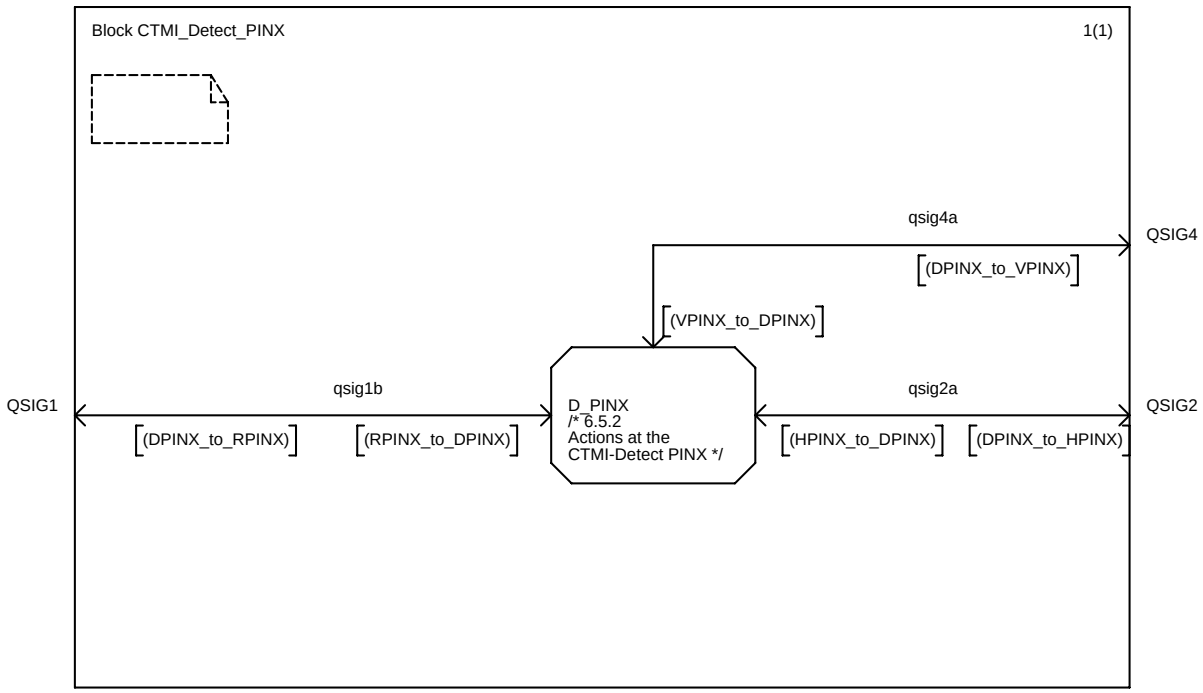

Figure 6: An example of a block (CTMI_Detect PINX)

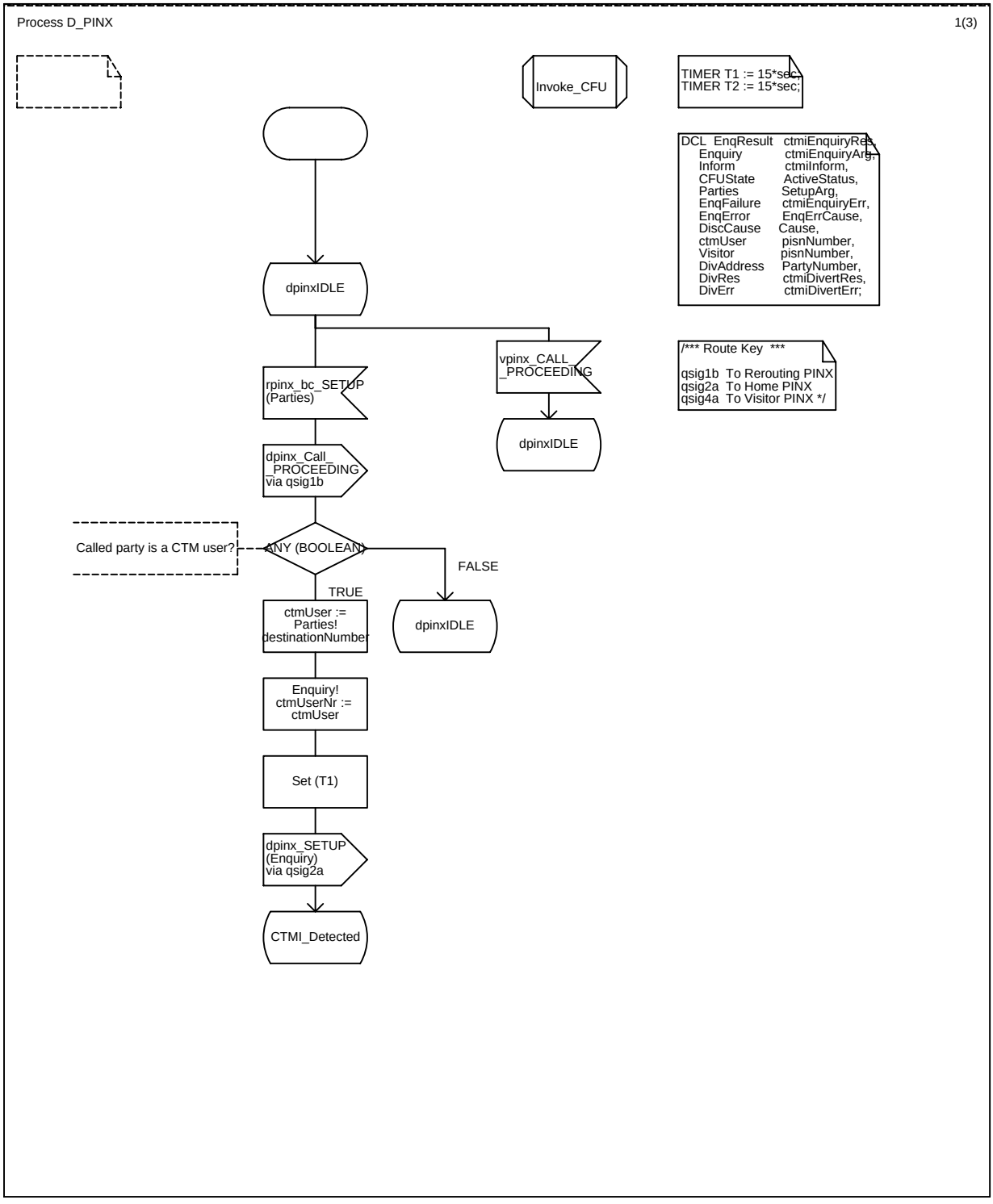

Figure 7: Example SDL showing use of start symbol, timers, DCL, route names and QSIG basic service protocol

5.4 Simplifying and improving the validation model

A number of conventions were implemented in order to make the formal validation model easier to read. These were mainly of an editorial nature:

- the word "Expiry" was added as a comment after the timer identifier in each time-out input symbol (see subclause 5.3);
- each export signal was prefixed with the identifier of the sending PINX so that it would be possible to identify the source of a message when reading only the SDL for the receiving process;
- a text box was added to each page of each process to act as a key to messages routes. This key identifies the name of each route and the physical entity which is reached by accessing the route (see subclause 5.3 and figure 1);
- functions such as the access of the HDB and VDB which were not essential to the control of the CTMI protocol were extracted to procedures so that they could be modelled at whatever level of complexity was deemed necessary.

6 The validation procedure

6.1 The validation method

Although the ANF-CTMI service is not very complex, there were significant benefits to taking a step-by-step approach to its validation. The method was as follows:

1. Recode the ASN.1 data operations into SDL signals, data types and synonyms.
2. Collect together associated signals into SDL signallists.
3. Draw the system diagram and block diagrams based on the architecture of the informal model.
4. Recode the procedures using the conventions described in subclause 5.4.
5. Use the syntax analyser on the SDL editing tool to check that the SDL syntax has been adhered to.
6. Use the semantic analyser on the SDL tool to check that the static semantics of the model are valid.

NOTE: Steps 1 to 6 were carried out using a PC variant of the SDL tool.

7. On the validation tool, set the priorities of time-outs and signals from the environment to be lower than signals internal to the model. Run the validator and correct any design errors found.
8. Repeat Step 1 with time-outs raised to the same priority level as internal signals.
9. Repeat Step 1 with all events raised to the same priority level.

Further steps to check manually generated Message Sequence Charts (MSCs) against the validation model were envisaged but could not be completed through lack of time.

7 The validation results

Having followed the stepwise method shown in subclause 6.1, there were no validation errors found in the model at Step 7 with external and timer events set to low priority. This configuration probably represents normal use of the service where time-outs are unlikely to occur and signals from the environment arrive in an orderly fashion. However, the Coverage Report showed that less than 50 % of the model was being exercised in this way.

Unnecessary validation errors (queue overflow) were generated in the early stages of testing because the default length of the message queue was set to 3 by the validation tool. In this particular study, increasing the queue length to 10 removed these problems.

Increasing the priority of the timer events improved the coverage to approximately 90 % and revealed a number of validation errors which were all of the "Implicit Signal Consumption" type. Further similar errors were discovered by the validation tool when the priority of external events was raised and coverage reached 95 %. The errors pointed to a smaller number of basic design faults which had to be corrected:

- The synchronization of states between the four physical entities (processes) was poor for the following reasons:
  - The basic service was modelled in a very simple way within the CTMI model. This meant that the "SETUP/CALL-PROCEEDING" and "DISCONNECT/RELEASE/RELEASE-COMPLETE" sequences were not used in all instances.
  - The Home PINX and Visitor PINX were, necessarily, modelled as linear processes which ran from the Idle state through some processing with two or three signal exports and then back to Idle. As there are no valid intermediate states, it was impossible to terminate these processes in the event of an unexpected time-out or call clearance. The model did not provide the necessary transitions in the Rerouting PINX and the CTMI Detect PINX to deal with such situations.
- The model did not take into consideration the fact that more than one similar events could occur at the same time. As an example, Timer T2 expires at the same time that the originating caller releases (i.e., before the service has been completed). This means that a DISCONNECT message from the Rerouting PINX to the CTMI Detect PINX crosses with a DISCONNECT message from the CTMI Detect PINX to the Rerouting PINX. Each process assumed that its DISCONNECT would initiate the necessary processing to prevent the other from being sent.
- Symbol coverage was initially reduced because the "X ::= ANY(BOOLEAN)" construct was used to randomly assign values to flags which would subsequently be tested to control the flow of control through the system. Although this is correct SDL, the validation tool interprets such an assignment by causing a value (always the same value) to be given to the boolean variable at compile-time rather than randomly at run-time. When these assignments were replaced with a general procedure (T_or_F) using a conditional statement based on "ANY", the coverage was improved.

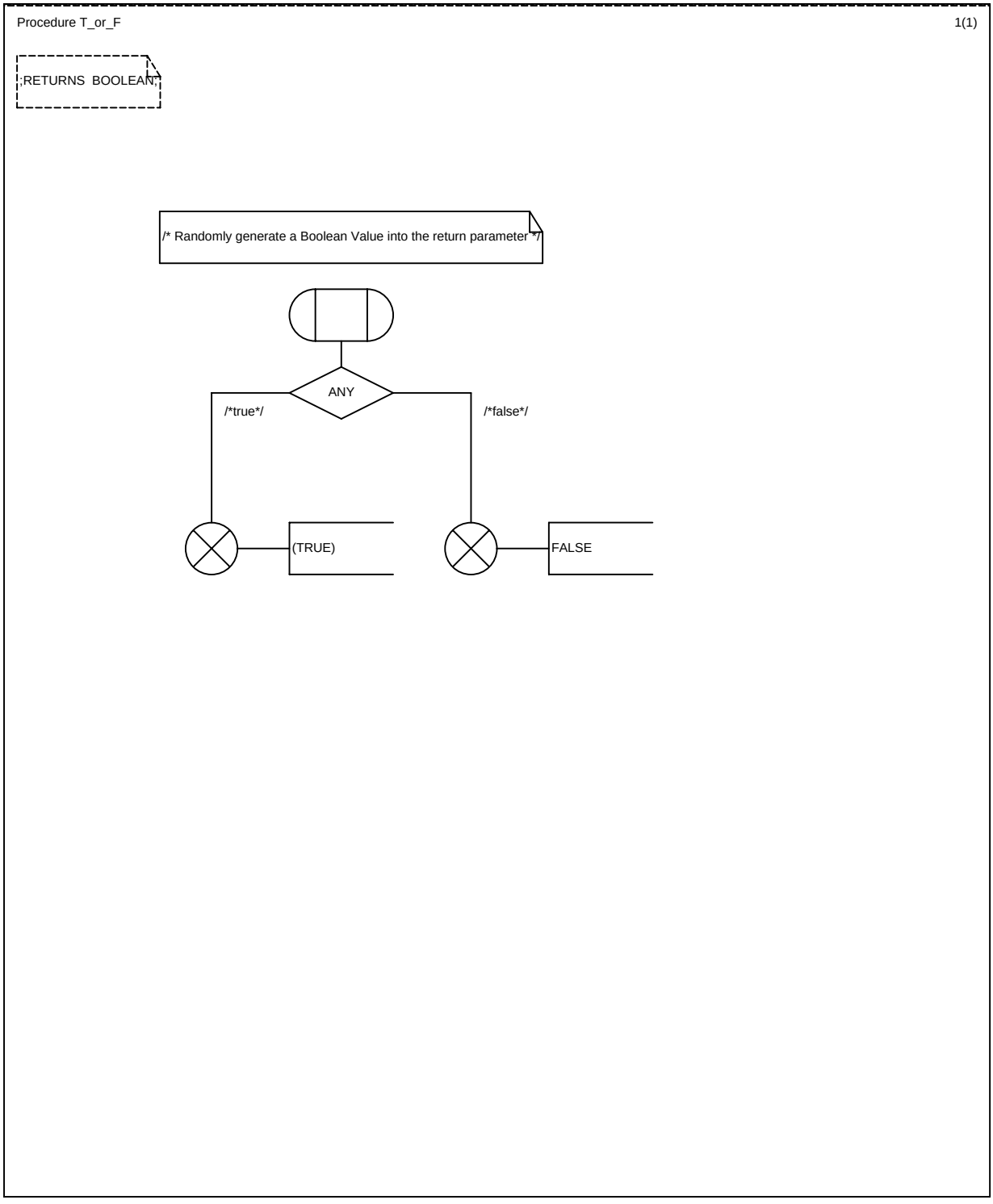

Figure 8: Procedure "T_or_F" to randomly assign boolean values to a variable

The model as it was originally produced from the informal SDL in ETS 300 696 [1] can be seen in annex B.

By progressively introducing corrections to the model, the number of implicit signal consumptions was reduced to three. These occur only after a complex series of abnormal events and could only be resolved by fully modelling the QSIG basic service. The symbol coverage was increased to 97,7 %. The final model can be seen in annex C.

8 Conclusions

It is not easy to determine the proportion of errors that existed in the original informal model as opposed to those that were introduced as part of the formalization process. The informal model uses simple statements in places where quite complex actions are necessary. As an example, the inclusion of Forward-Switching from the CTMI Detect PINX to the Visitor PINX when rerouting is not available, means that the call to the CTM user could be routed either through the Rerouting PINX or the CTMI Detect PINX. Further call handling then has to be take account of which PINX processed the call and this affects the SDL in both of these PINXs as well as the Visitor PINX. However, in the informal model, Forward-Switching is dealt with in a single task symbol containing the text, "*Do forward switching to the Visitor PINX*".

It is certainly true that the informal model did not take into account the synchronization of processes and the consumption of return messages that could or would arrive after a timer expiry or an unexpected release of the incoming call. In fact, as the informal SDL was expressed in a simple way in order to make it easy to read, it was not possible for it to contain very many basic design faults. One of the objectives of this experiment was to produce SDL that was easy to read and which could also be validated using automated tools and every attempt was made to carry forward the simplicity of the informal model into the validation model by the following means:

- avoiding the use of object orientation and other complex structures;
- keeping the division of processes into the Rerouting PINX, the CTMI Detect PINX, the Home PINX and the Visitor PINX;
- using signal names and data types that are recognizable from the ASN.1.

However, the validation model is not as easy to read and interpret as the informal SDL. The reduction in readability comes mainly from the additional complexity of function that had to be introduced to handle the basic service and the synchronization of processes.

8.1 Lessons to learn

The following aspects should be considered in the development of a methodology for the validation of SDL specifications:

1. Before beginning the formalization of a supplementary service it is necessary to reach agreement on the method to be used to incorporate the basic service into the validation model. There are a number of alternatives for this and it is not certain that the method chosen for ANF-CTMI, i.e., the integration of simplified basic service messages directly into the processes of the validation model, is the best approach. Further study is needed in order to determine which are the best alternatives for the implementation of basic service.
2. When preparing the validation model, move all activities that do not place requirements on implementors into procedures. Examples in the ANF-CTMI experiment are the HDB and VDB access procedures.
3. Special care should be taken to avoid process synchronization problems e.g. all parts of the protocol should be in "Idle" state before a new request is handled from the environment.
4. Once the formal model is coded, a step-by-step method should be followed for the validation of the model. In simple terms, the steps should be based on the following:
    i Analyse and correct the syntax of the SDL.
    ii Analyse and correct the static semantics of the model.
    iii Validate the model giving a high priority to internal signals. Correct any reported errors.
    iv Validate the model giving a high priority to internal signals and timer events. Correct any reported errors.
    v Validate the model giving a high priority to internal signals, timer events and signals from the environment. Correct any reported errors and assess the symbol coverage to determine whether it could be improved by changes to the model.
4. When evaluating errors reported by the validation tool, consider all of the errors together before making changes to the model. Taking one error at a time can lead to unnecessary coding and subsequent recoding of the SDL. However, it is not advisable to allow the validation to continue until large numbers of errors have been detected.

5. Although validation tools report errors such as implicit signal consumptions and produce MSCs to indicate where in a sequence of messages the error occurred, it should be remembered that the reported error is usually only a symptom of a design problem in a transition which may have occurred much earlier in the sequence;
6. Much information can be found by looking at the Coverage. Symbols not executed at all are probably not needed or cannot be executed in the configuration chosen for that Validation.

Annex A:
ANF-CTMI informal SDL taken from ETS 300 696

The following SDL is the informal stage-3 model of ANF-CTMI exactly as it appears in ETS 300 696 [1]. It consists of a single process, QSIG_CTMI, with a page for each of the physical entities, as follows:

- Rerouting_PINX;
- CTMI_detect_PINX;
- Home_PINX;
- Visitor_PINX.

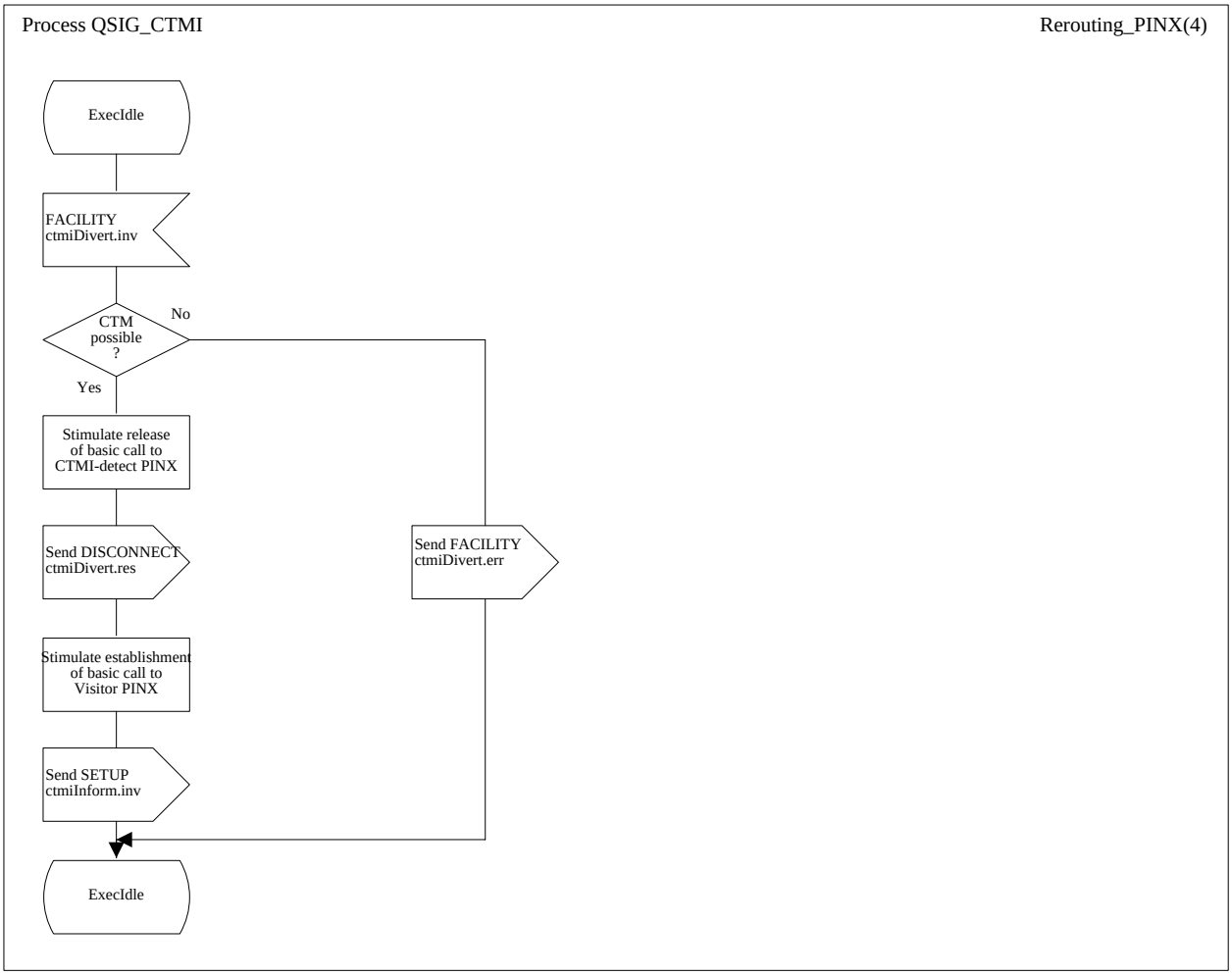

Figure A.1: Process QSIG_CTMI; page 1 of 4; Rerouting_PINX

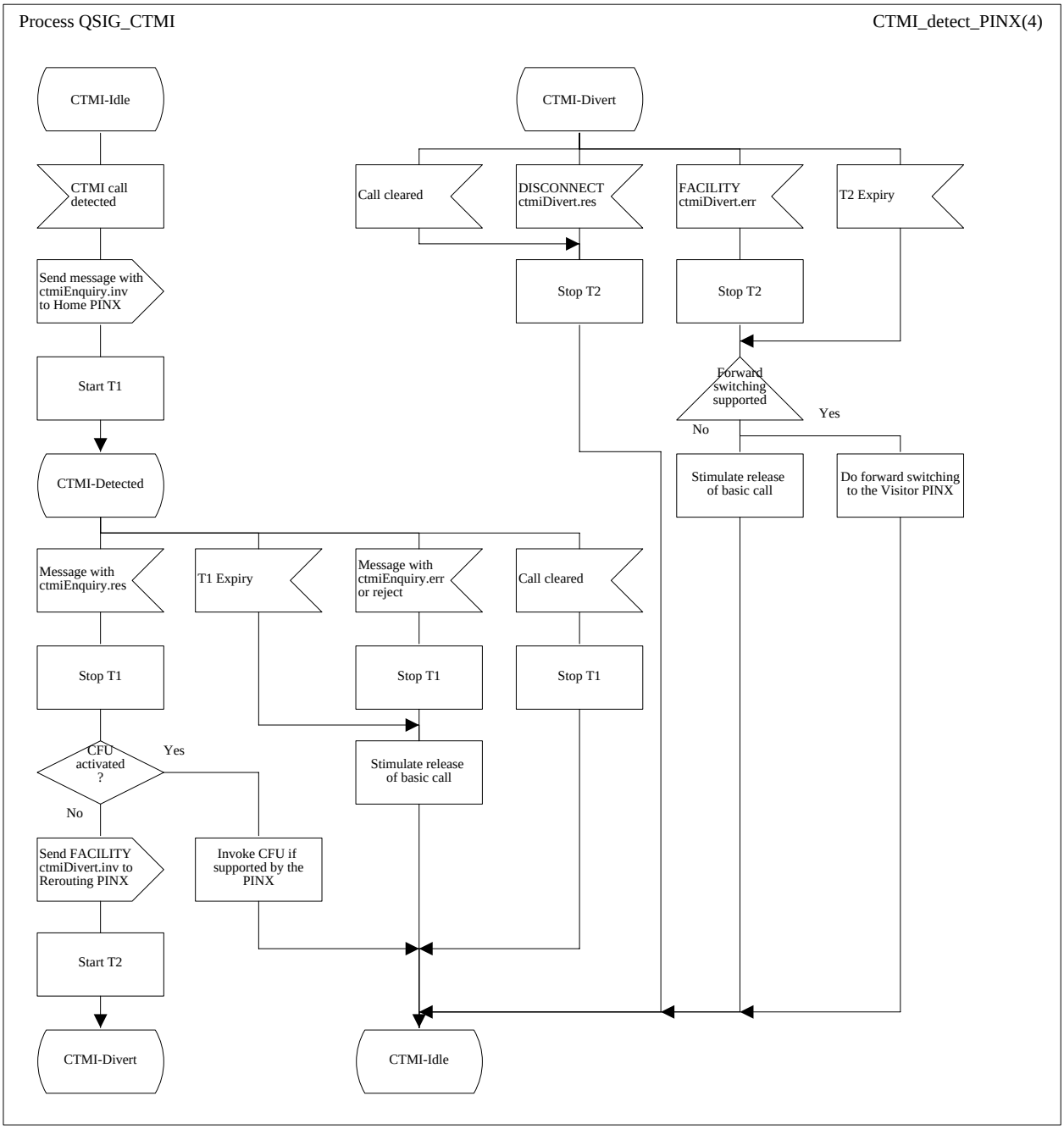

Figure A.2: Process QSIG_CTMI; page 2 of 4; CTMI_detect_PINX

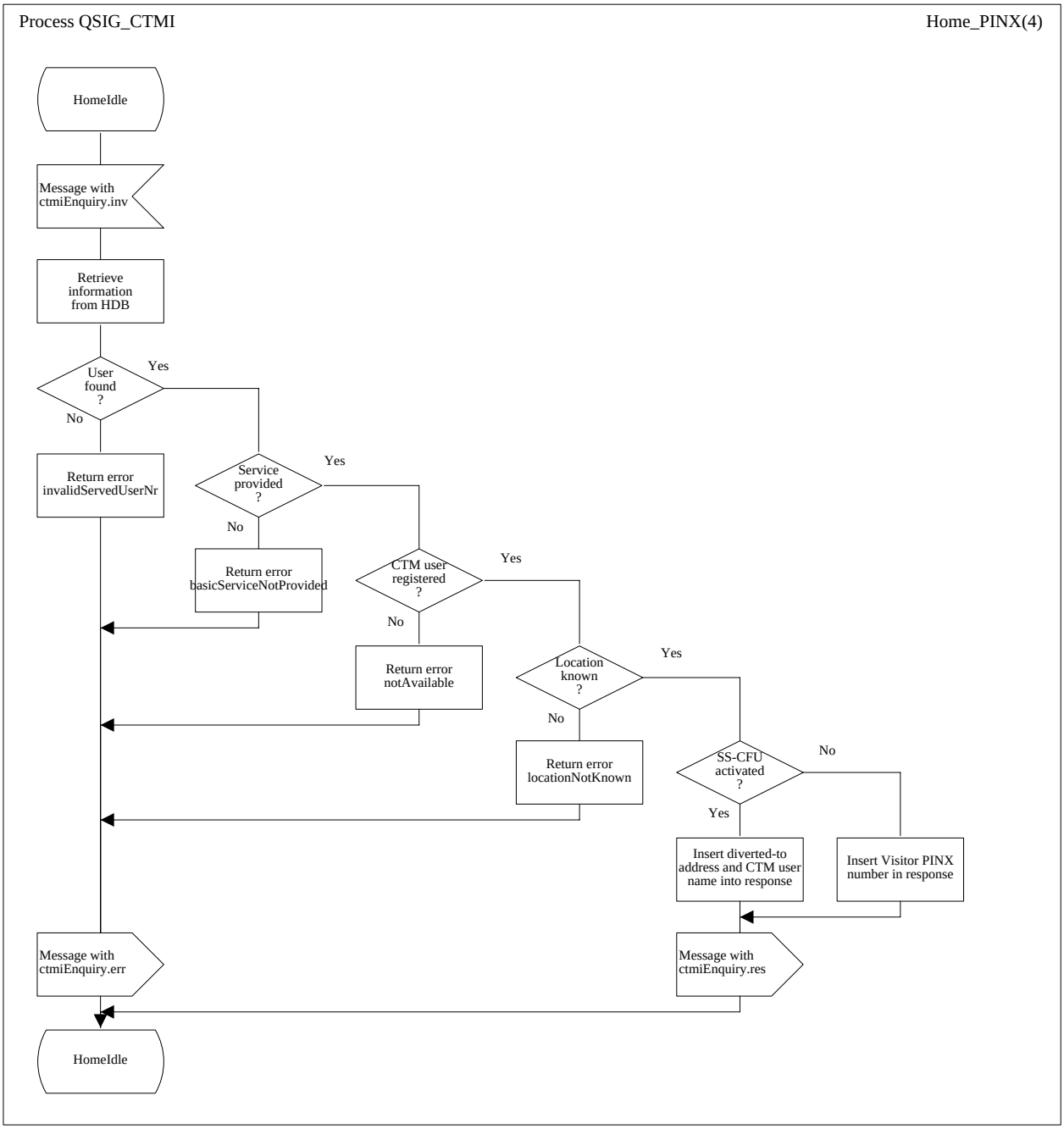

Figure A.3: Process QSIG_CTMI; page 3 of 4; Home_PINX

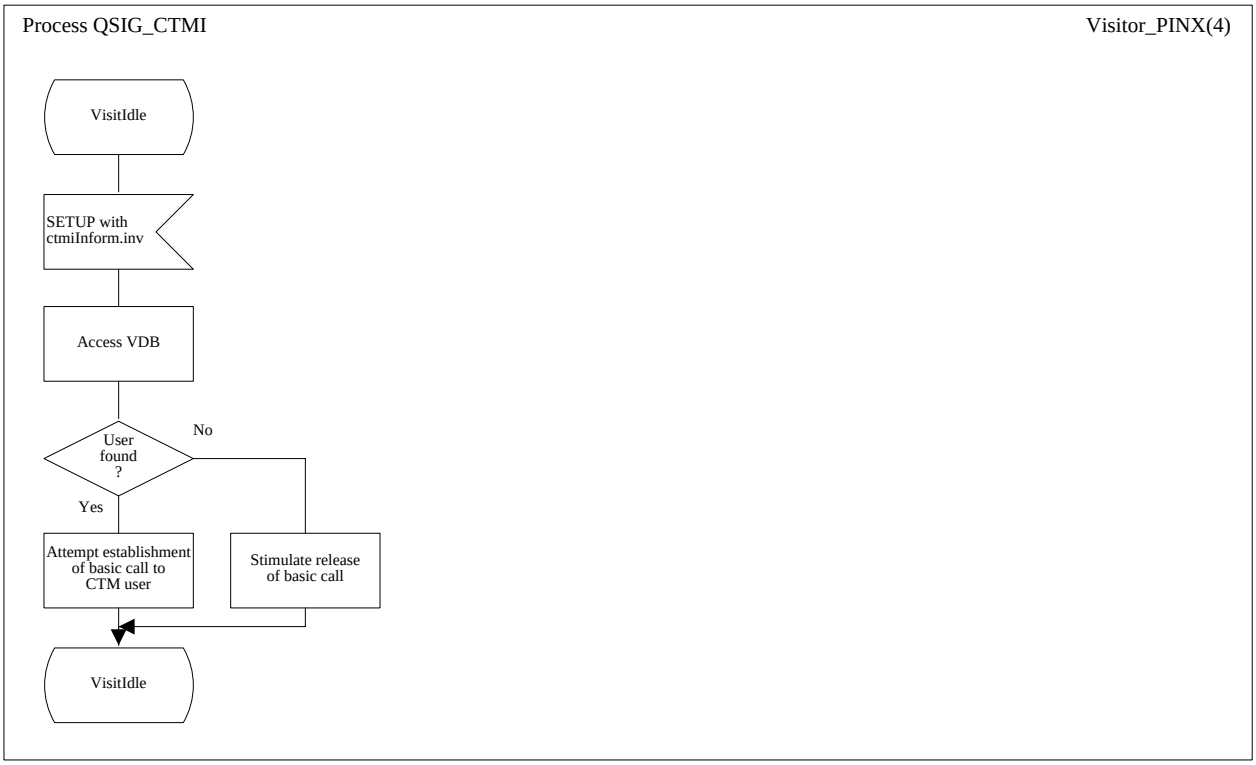

Figure A.4: Process QSIG_CTMI; page 4 of 4; Visitor_PINX

Annex B:
First formal model of ANF-CTMI

The following SDL is the formal model of ANF-CTMI on completion of successful syntax and static semantic analysis but before the start of validation. Each of the four physical entities is modelled as a separate process as follows:

- Rerouting PINX Process *R_PINX;*
- CTMI detect PINX Process *D_PINX;*
- Home PINX Process *H_PINX;*
- Visitor PINX Process *V_PINX*.

In addition, there are simple procedures included for the following functions:

- Invoke SS-CFU Procedure Invoke_CFU;
- Retrieve data from HDB Procedure AccessHDB;
- Retrieve data from VDB Procedure *AccessVDB*.

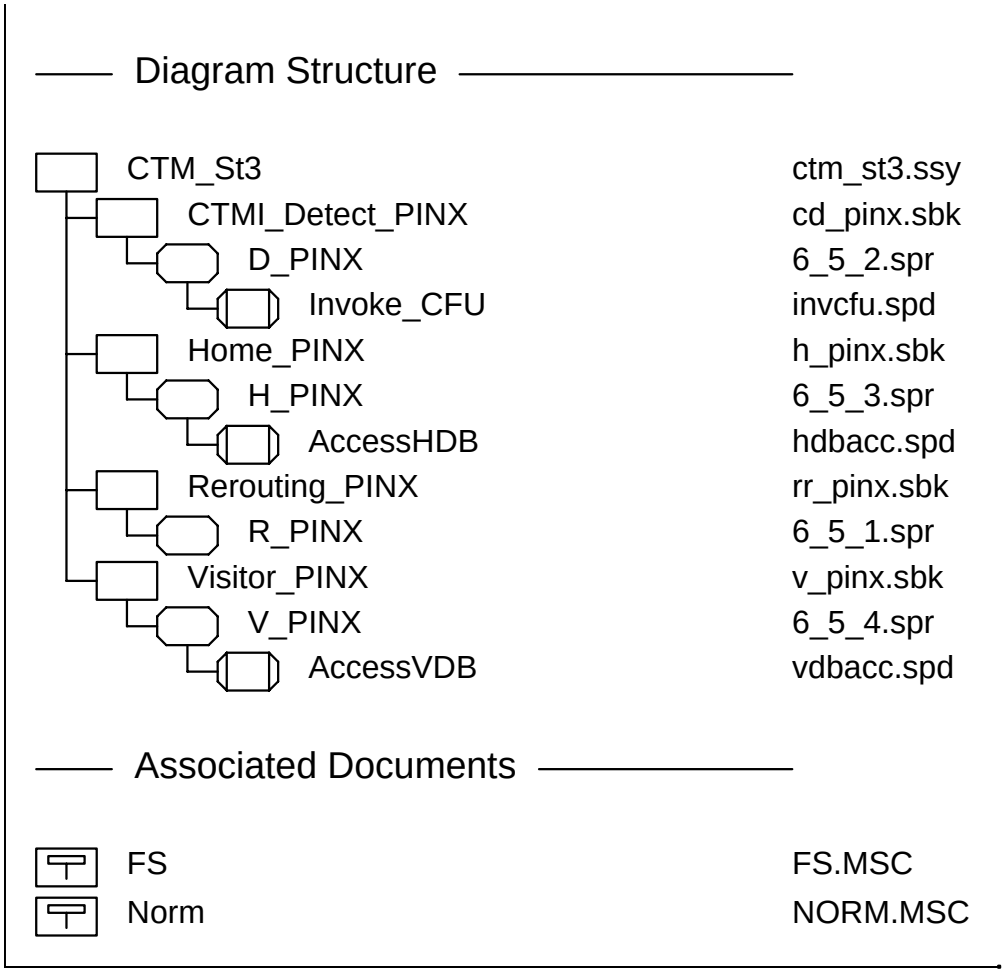

Figure B.1: ANF-CTMI SDL Model Organizer View

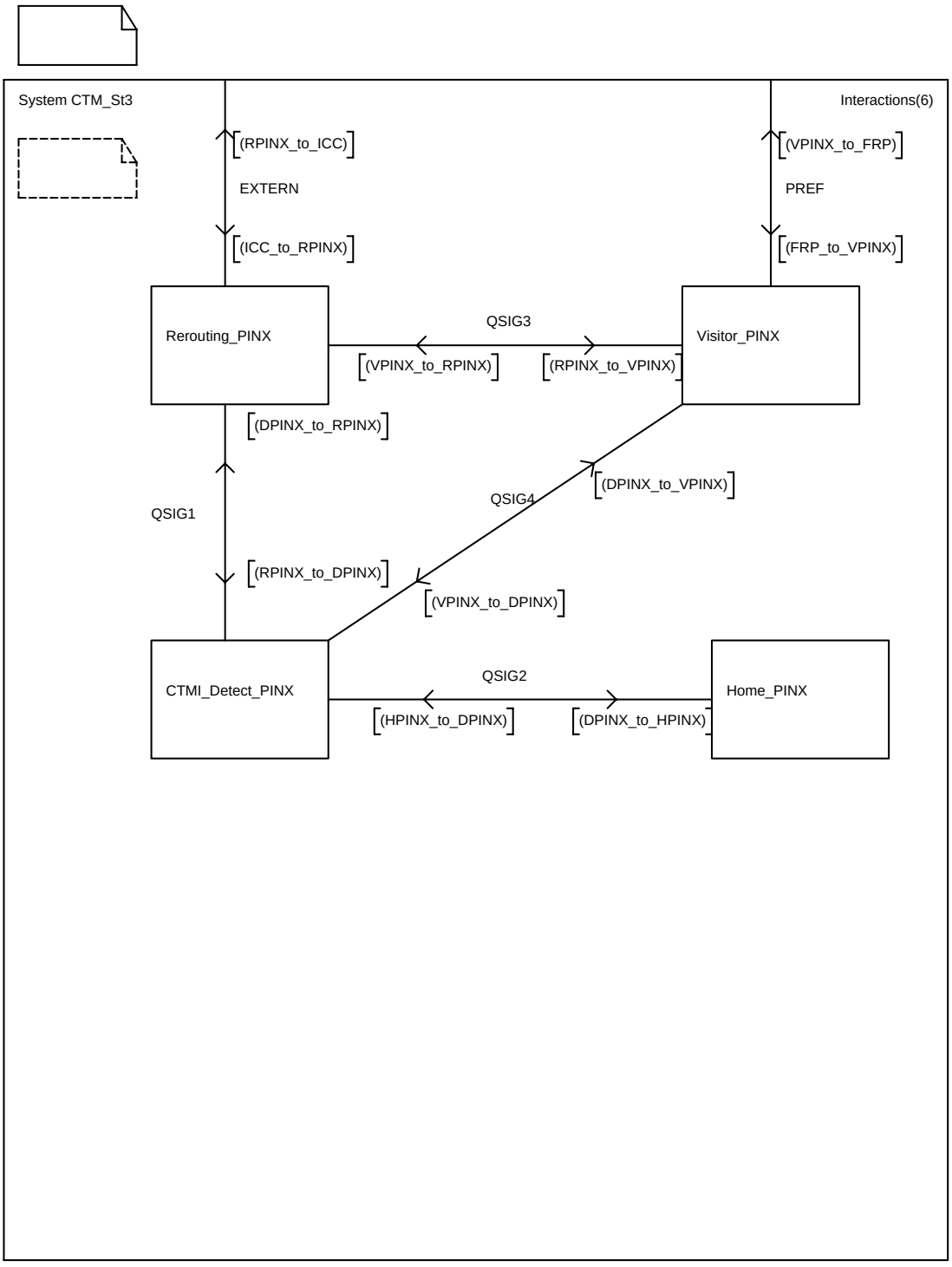

Figure B.2: ANF-CTMI SDL System

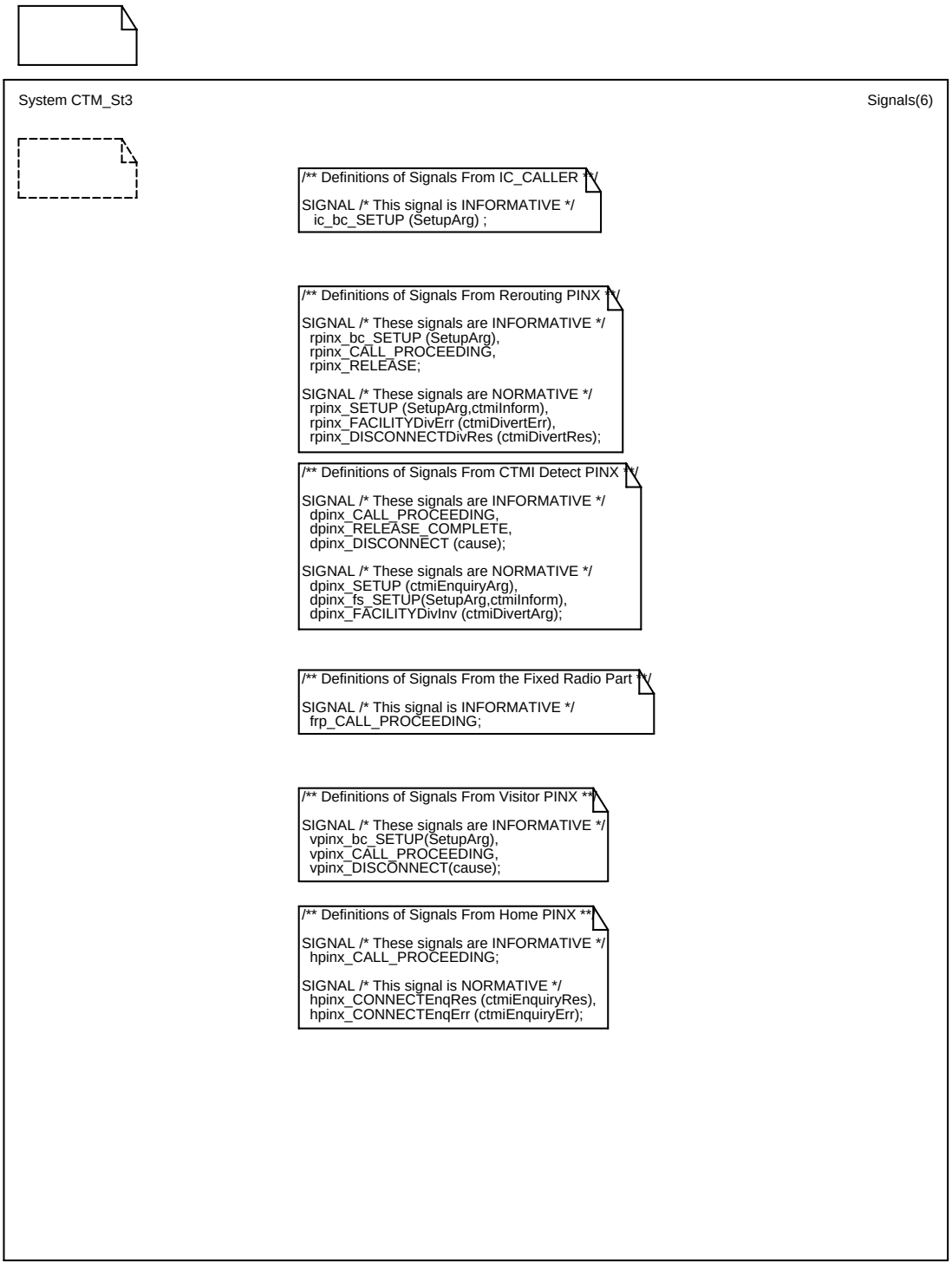

Figure B.3: ANF-CTMI Signal definitions

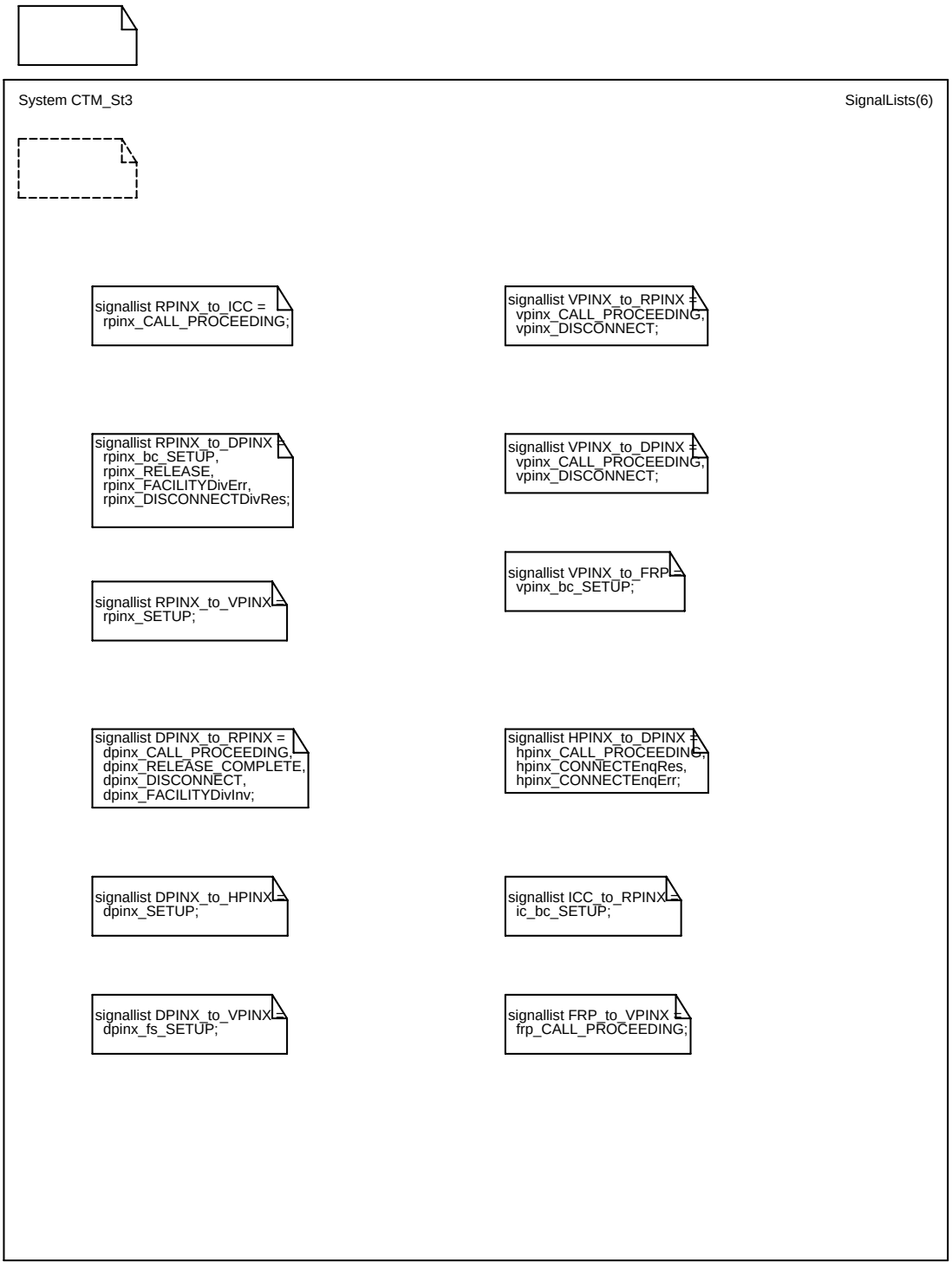

Figure B.4: ANF-CTMI signallist definitions

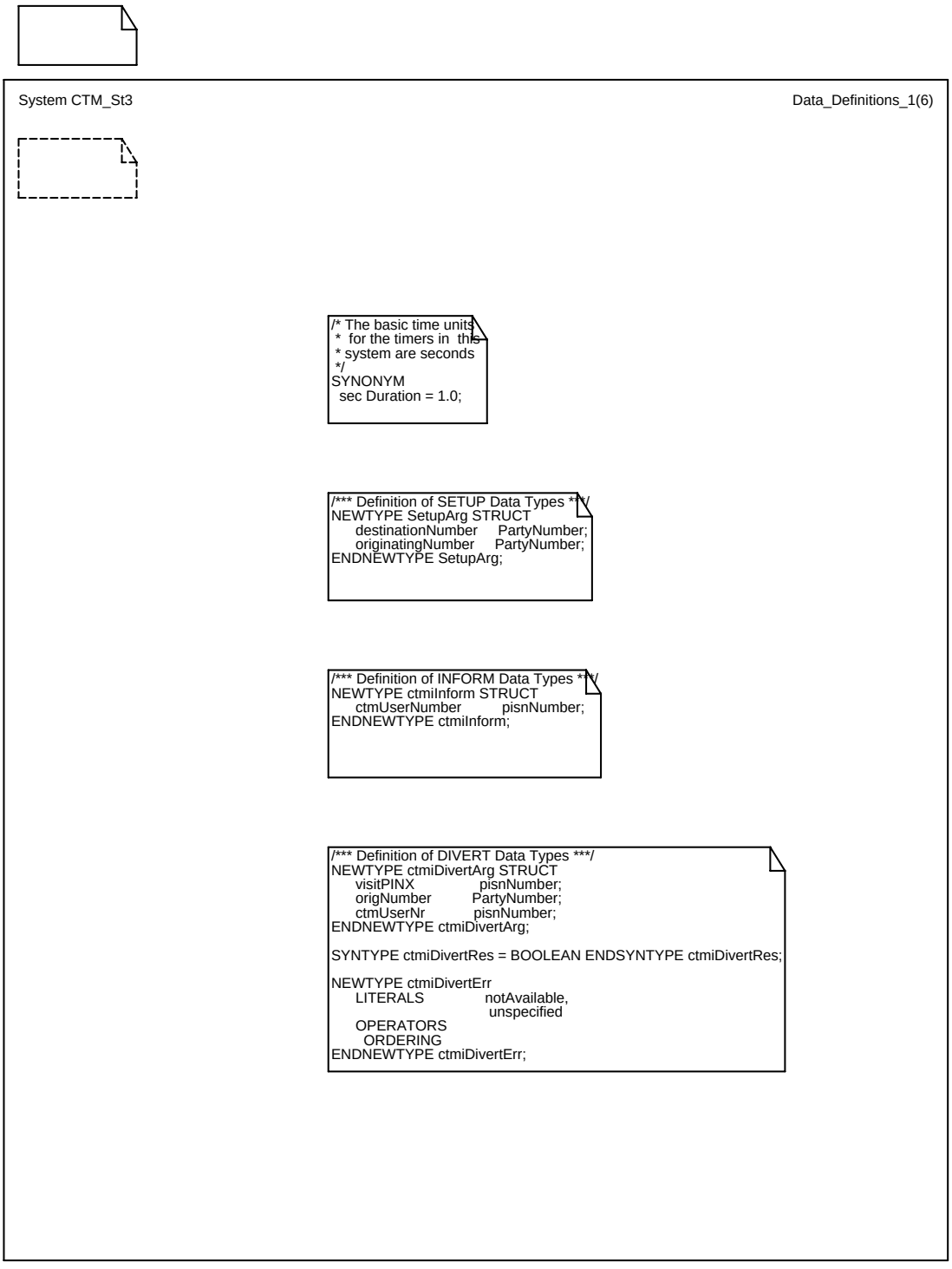

Figure B.5: ANF-CTMI Data types - page 1 of 3

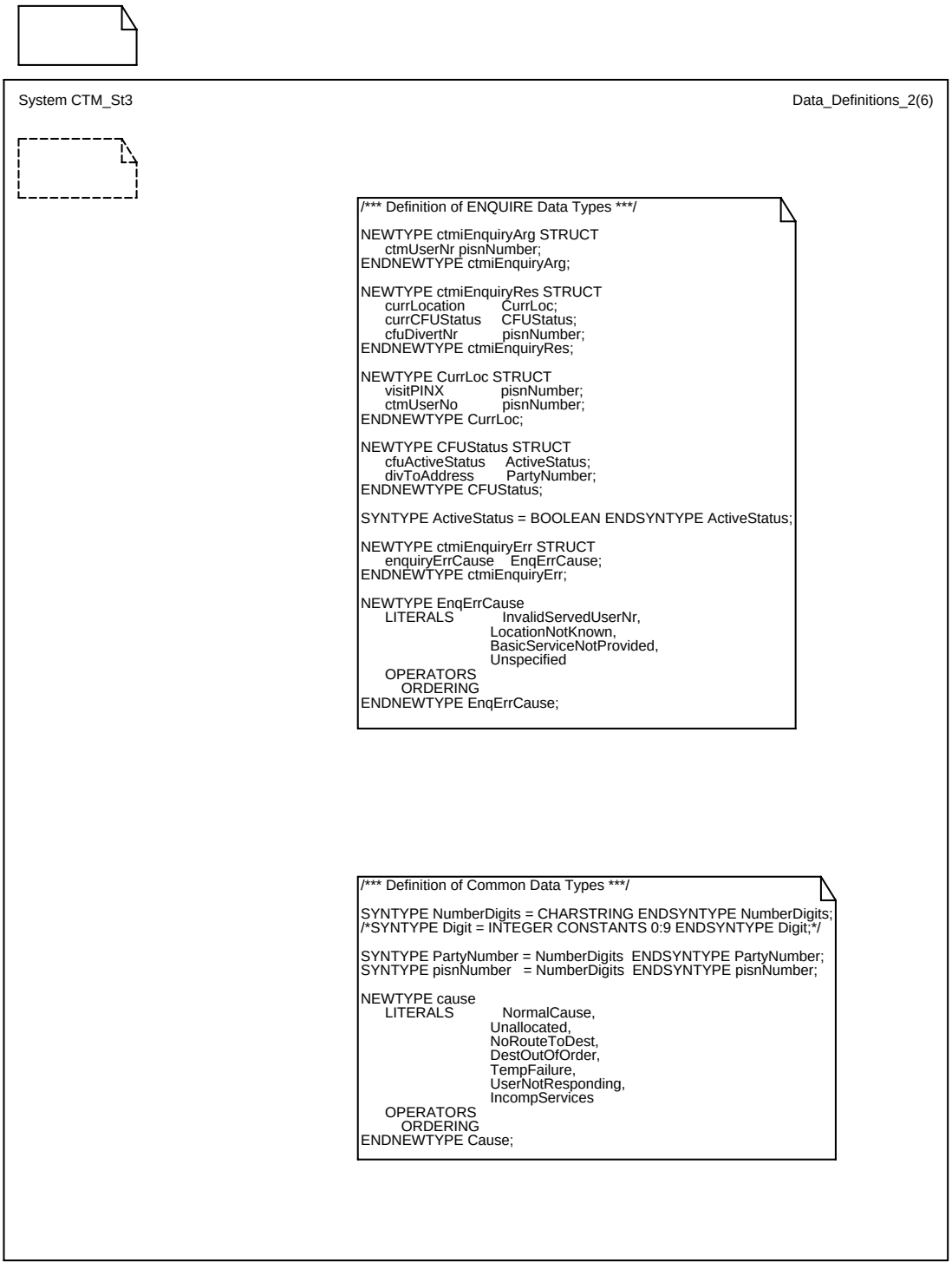

Figure B.6: ANF-CTMI Data type - page 2 of 3

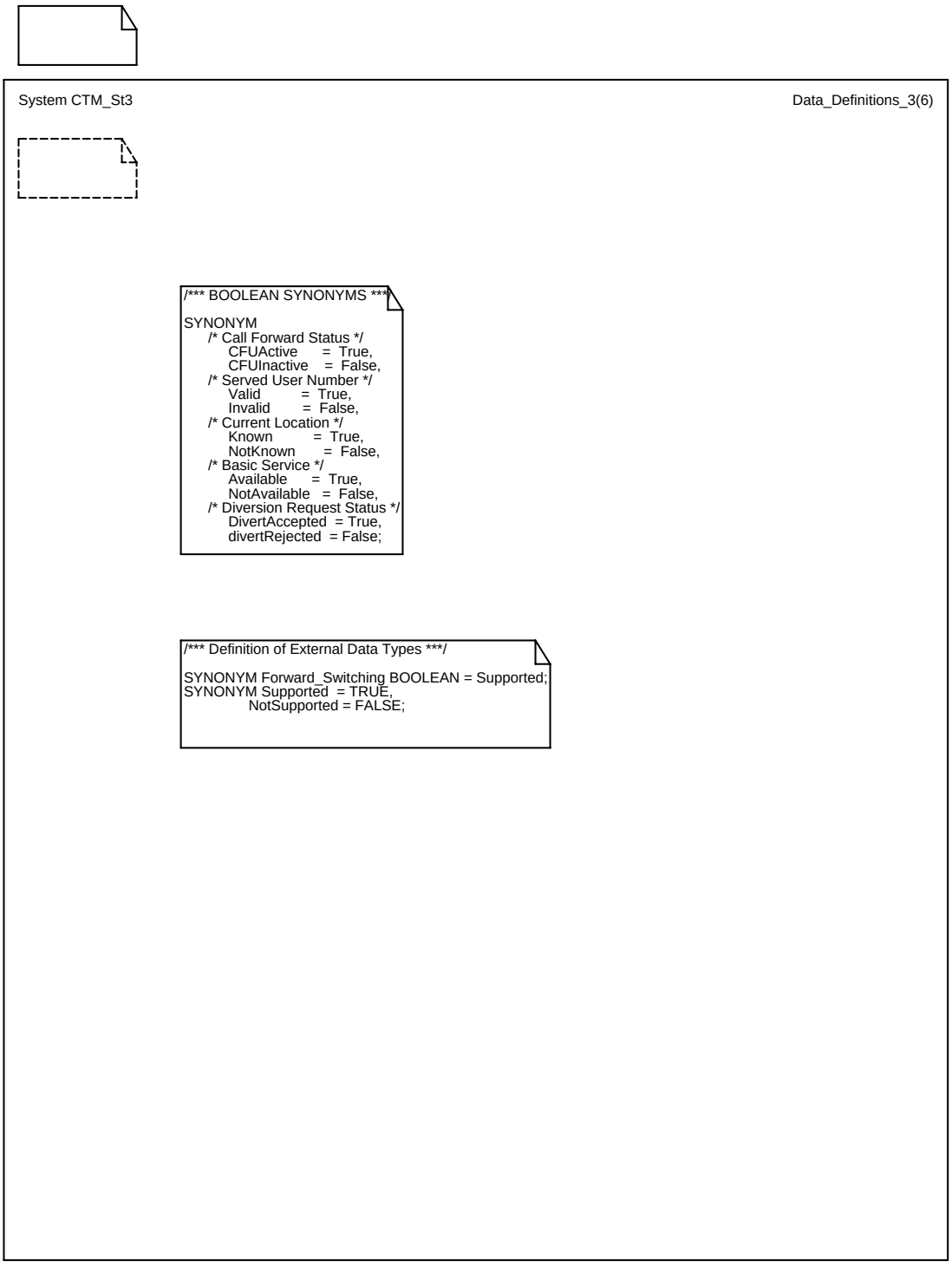

Figure B.7: ANF-CTMI Data type - page 3 of 3

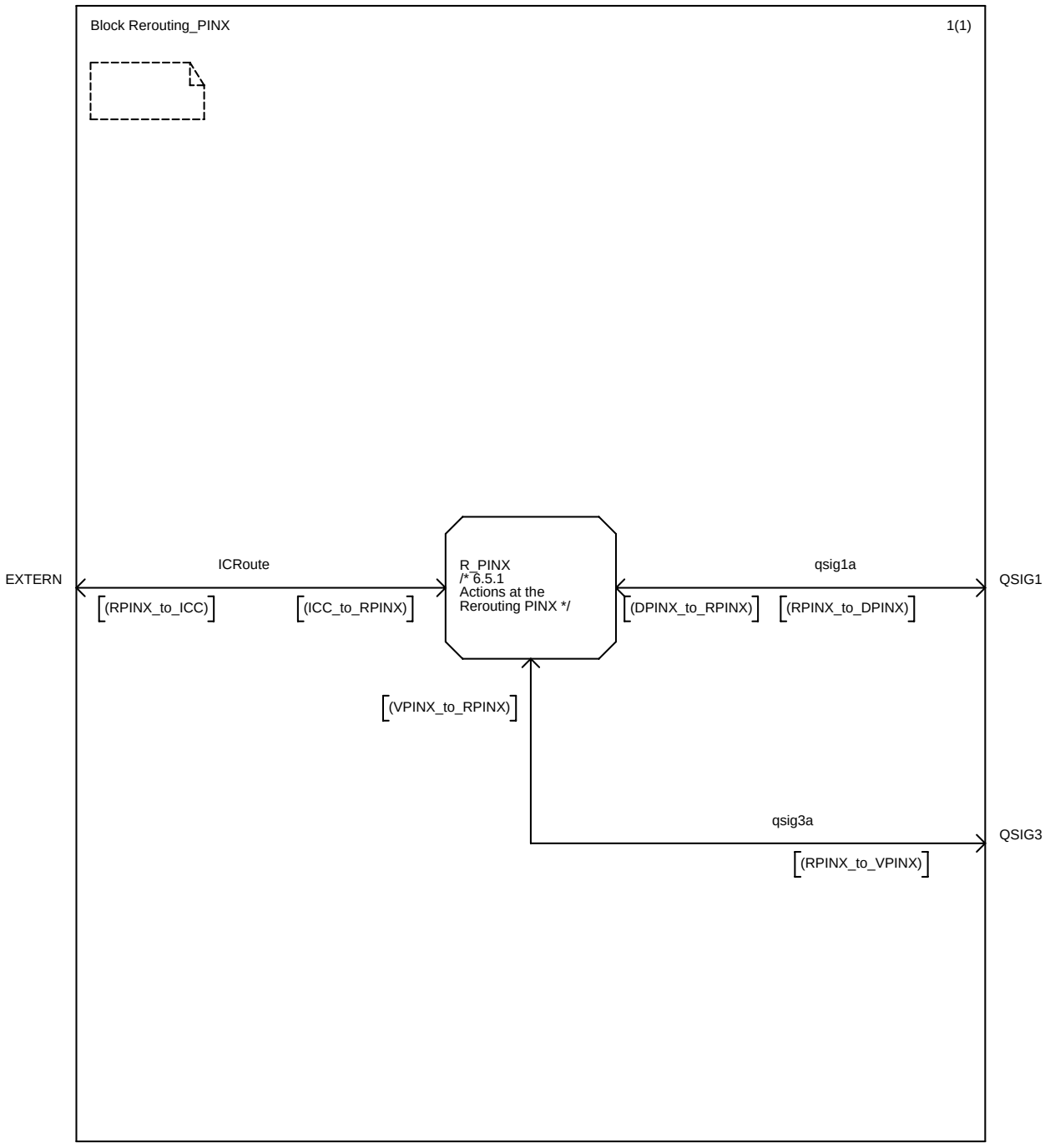

Figure B.8: ANF-CTMI Rerouting PINX Block

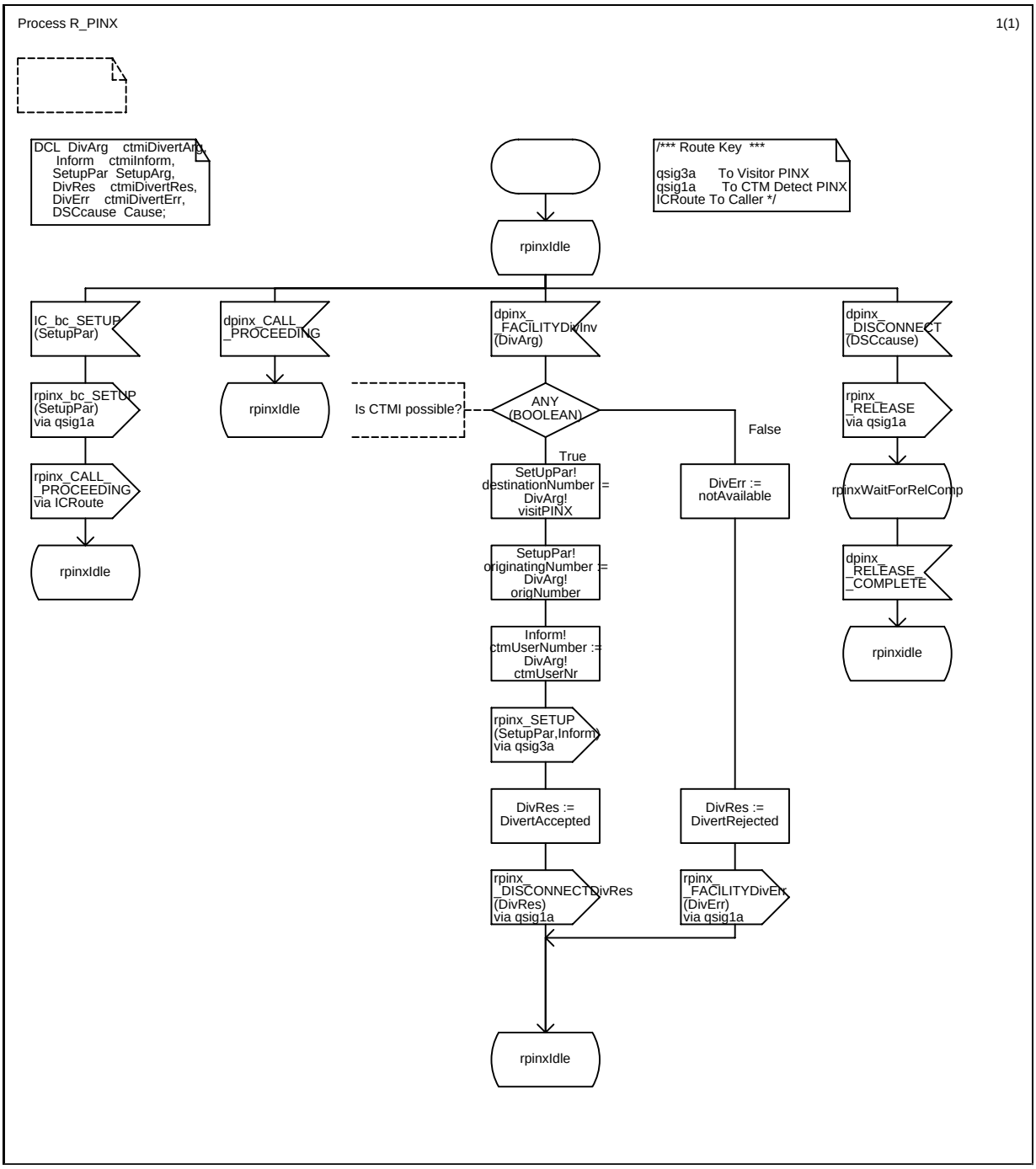

Figure B.9: Process R_PINX - page 1 of 1

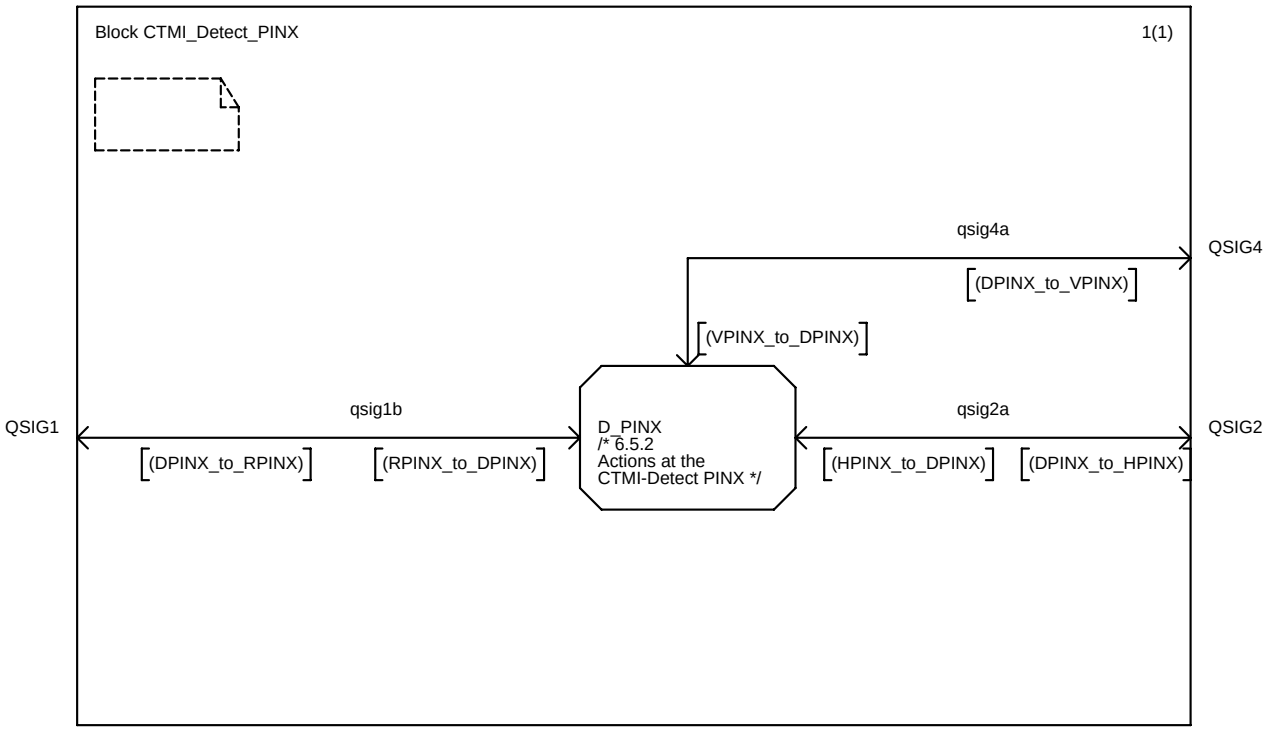

Figure B.10: ANF-CTMI CTMI Detect PINX Block

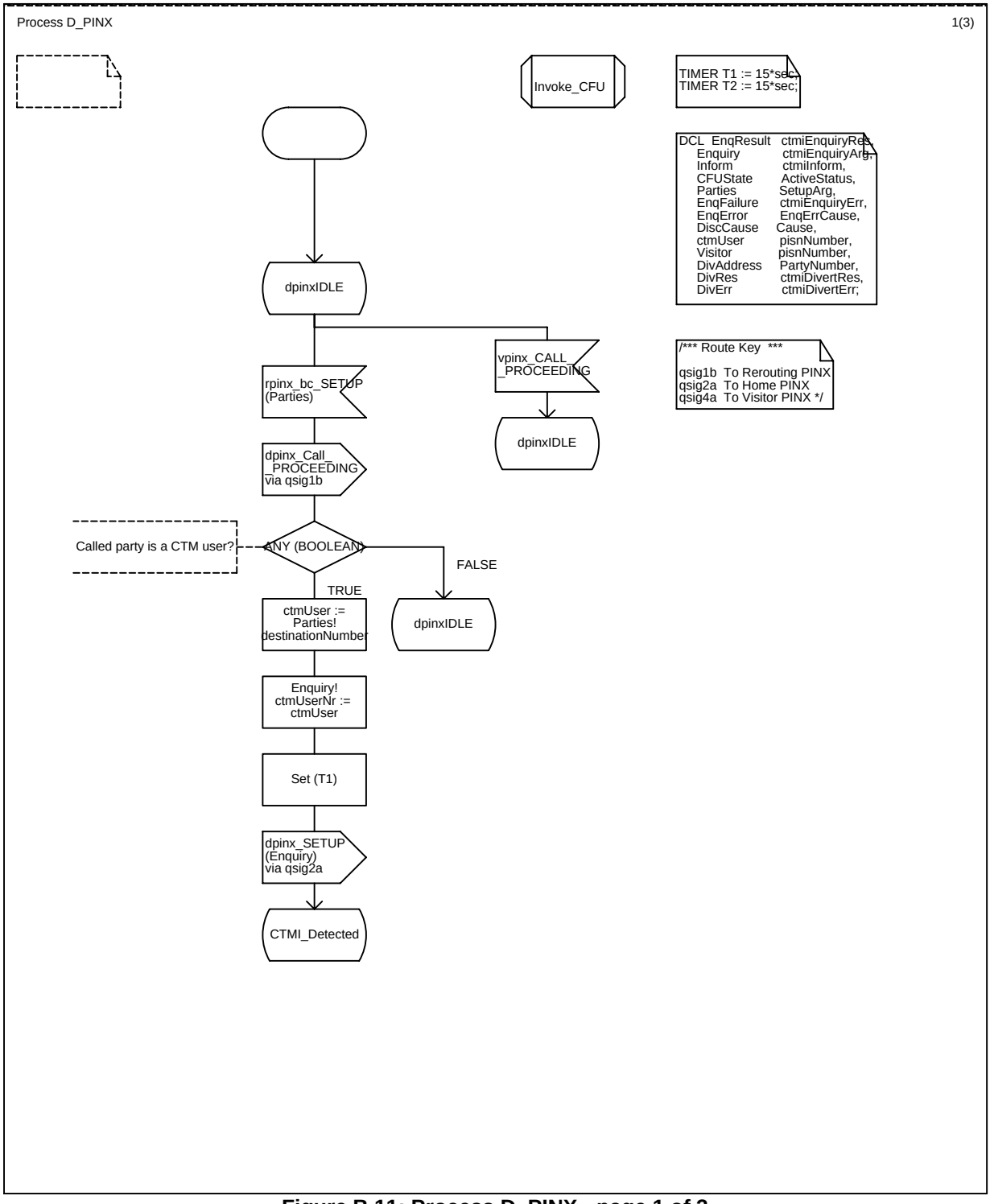

Figure B.11: Process D_PINX - page 1 of 3

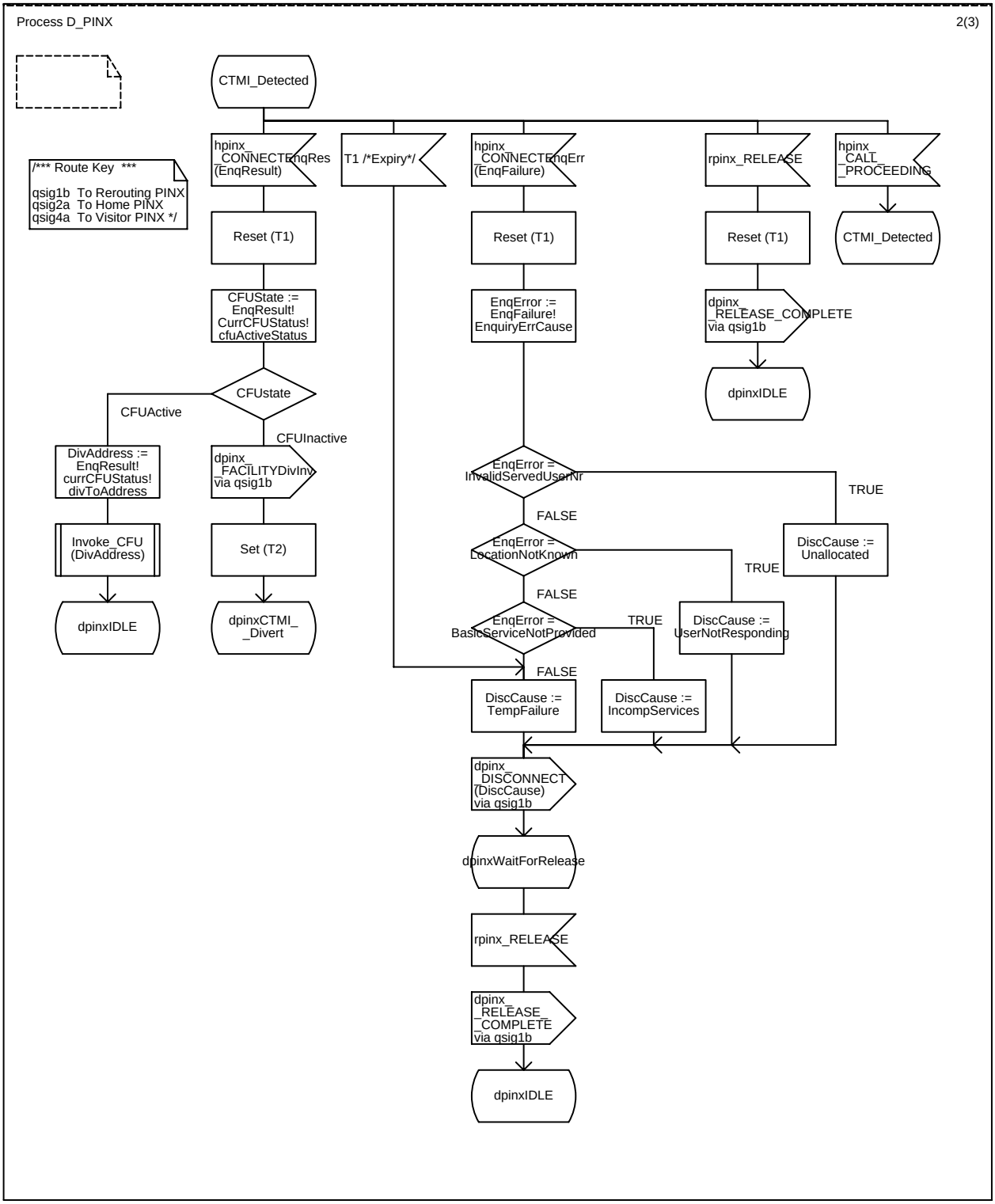

Figure B.12: Process D_PINX - page 2 of 3

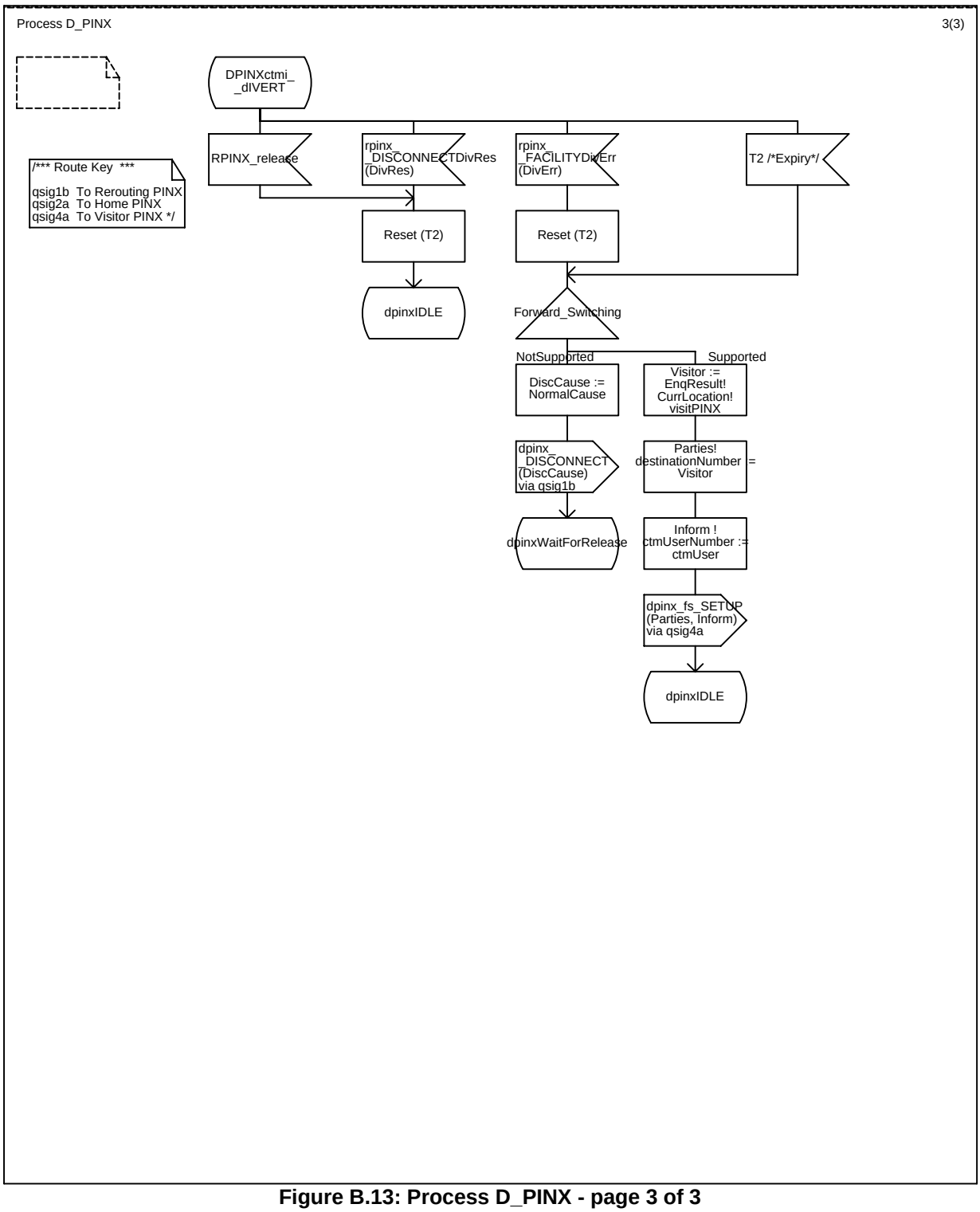

Figure B.13: Process D_PINX - page 3 of 3

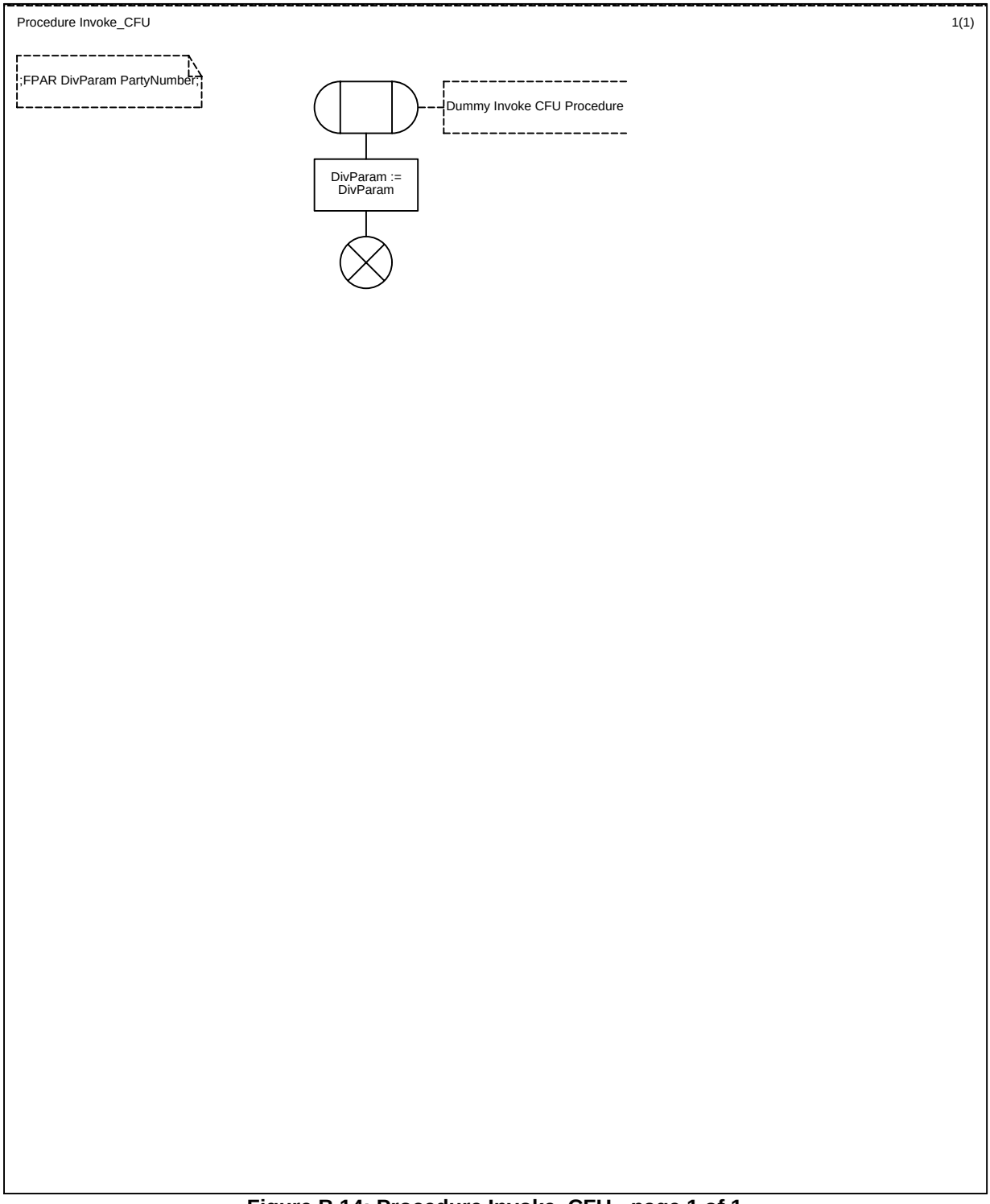

Figure B.14: Procedure Invoke_CFU - page 1 of 1

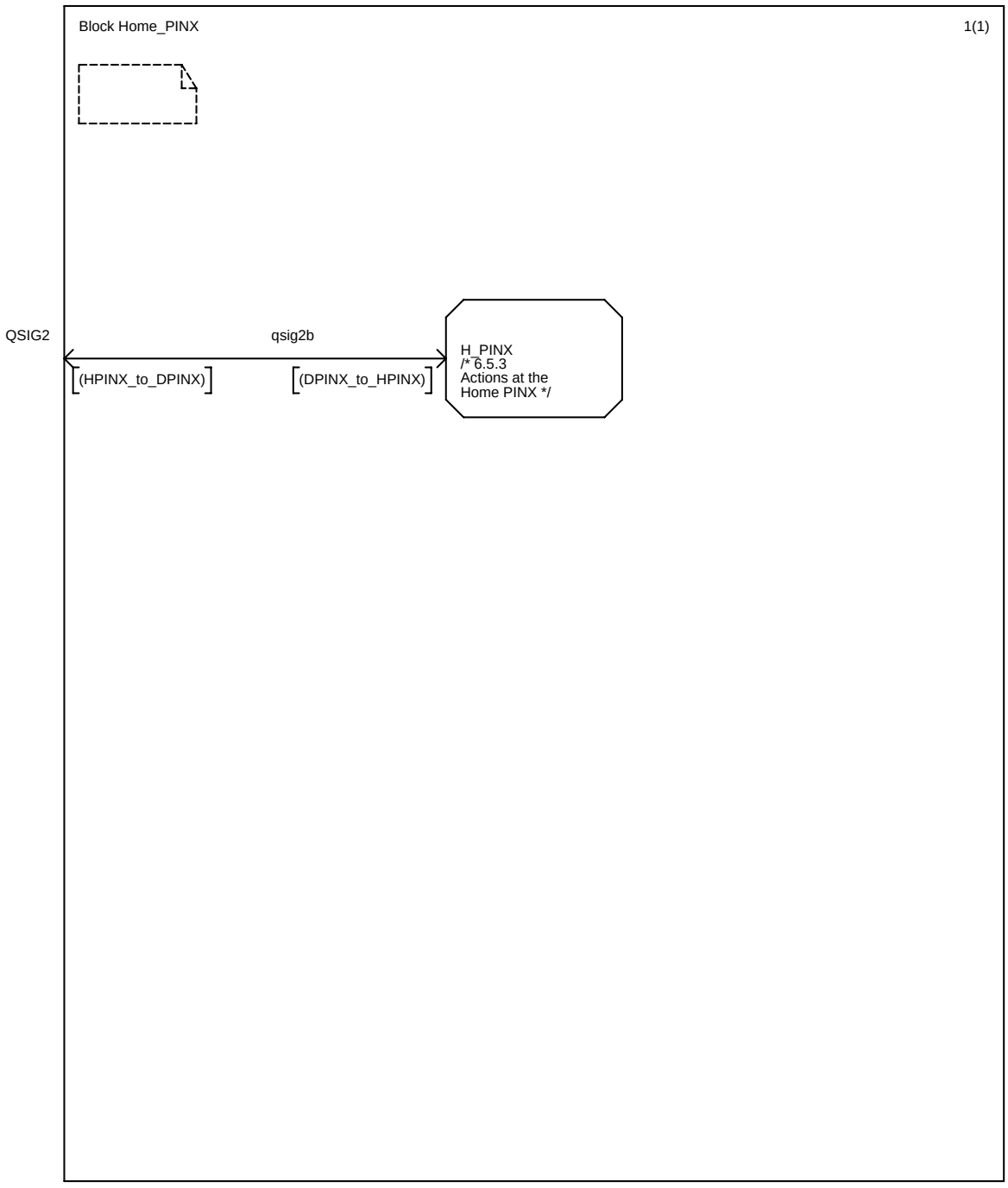

Figure B.15: ANF-CTMI Home PINX Block

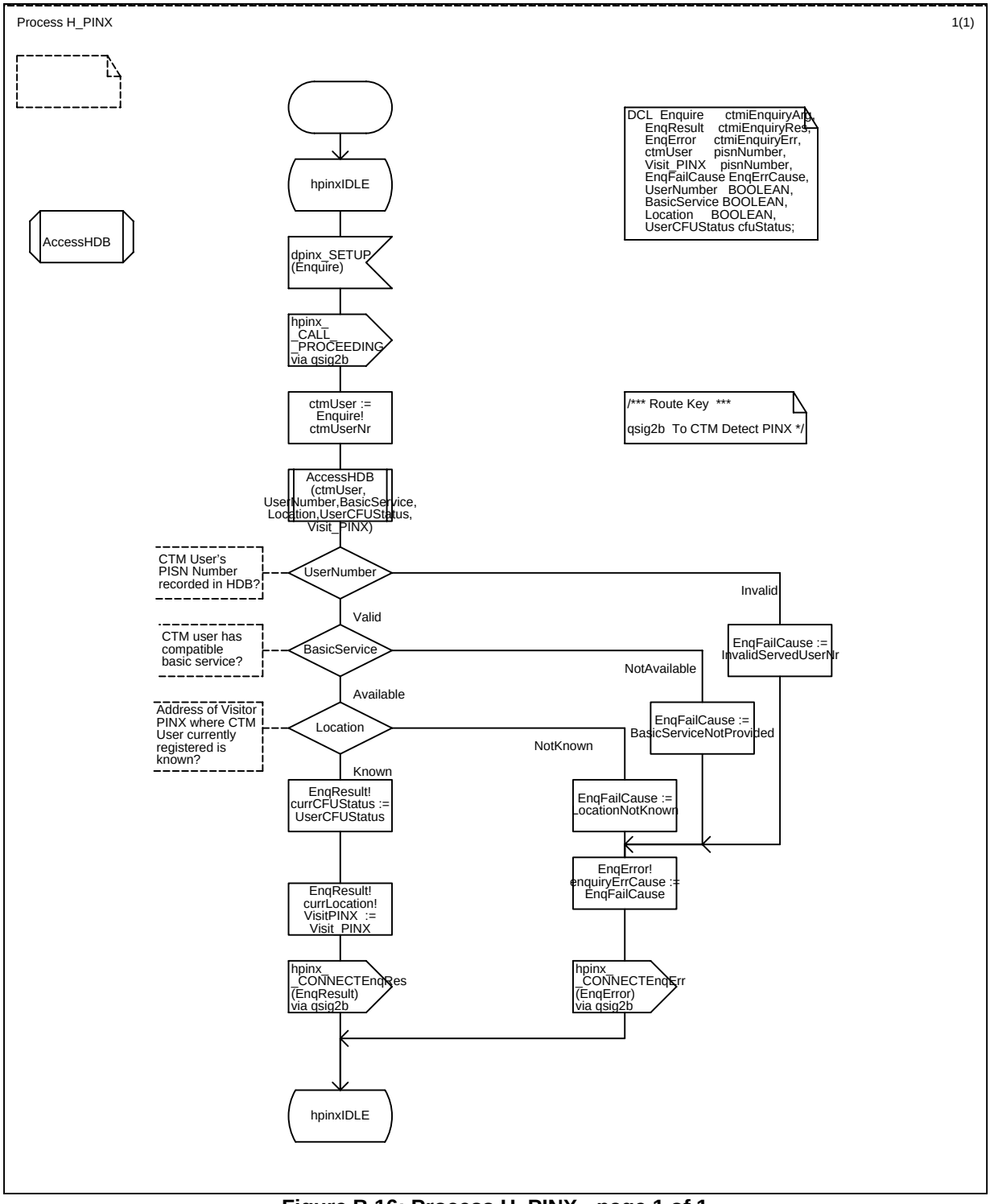

Figure B.16: Process H_PINX - page 1 of 1

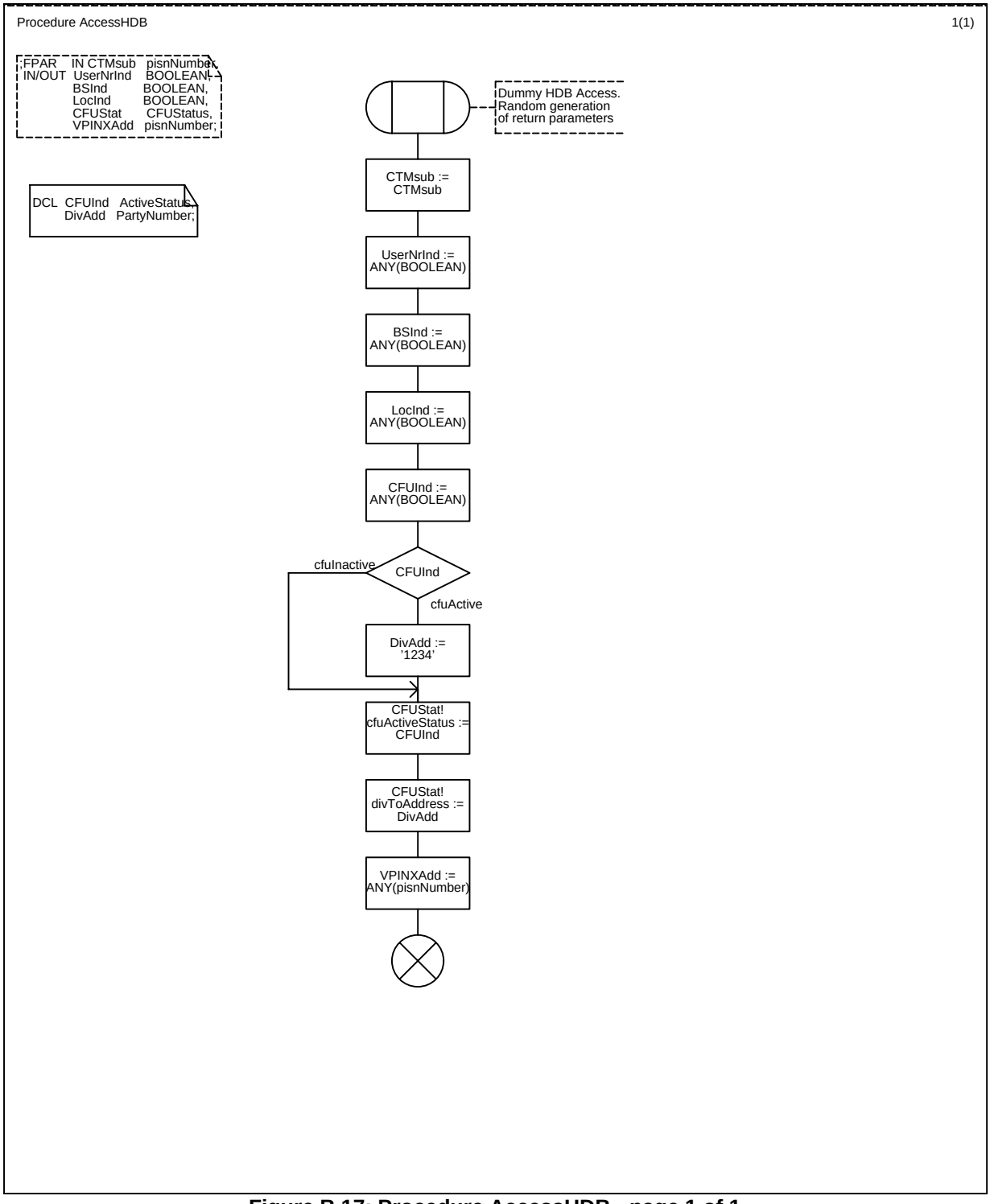

Figure B.17: Procedure AccessHDB - page 1 of 1

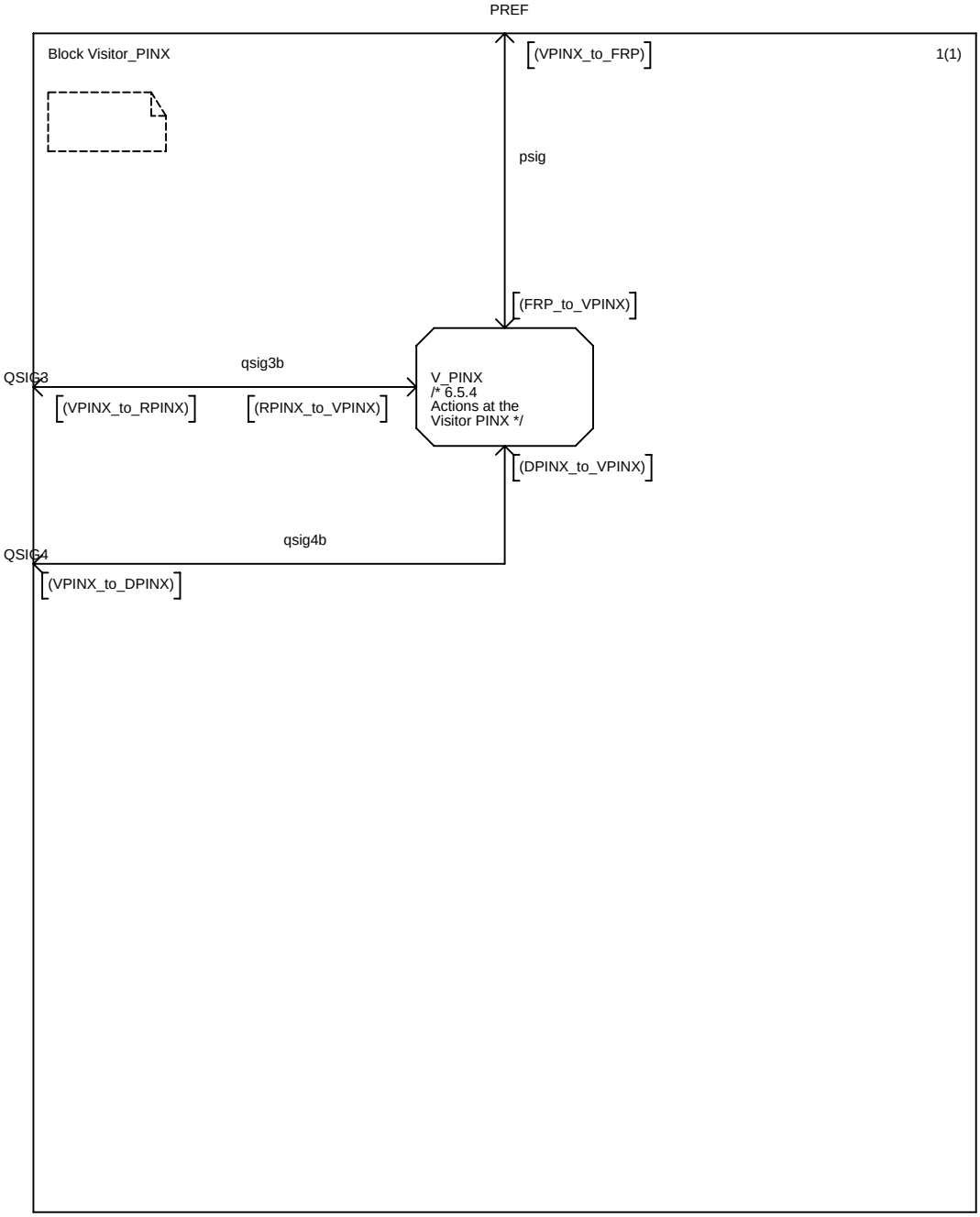

Figure B.18: ANF-CTMI Visitor PINX Block

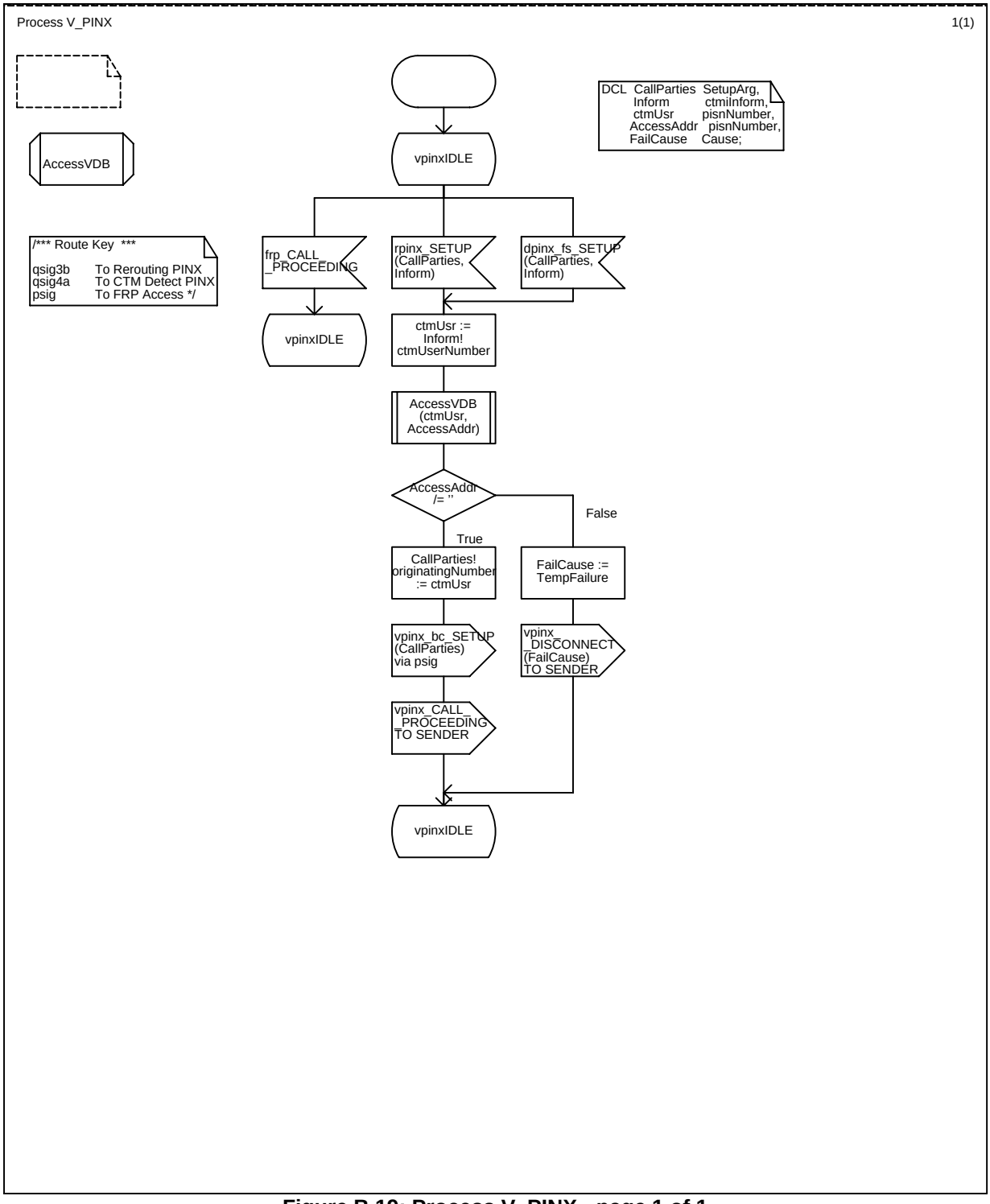

Figure B.19: Process V_PINX - page 1 of 1

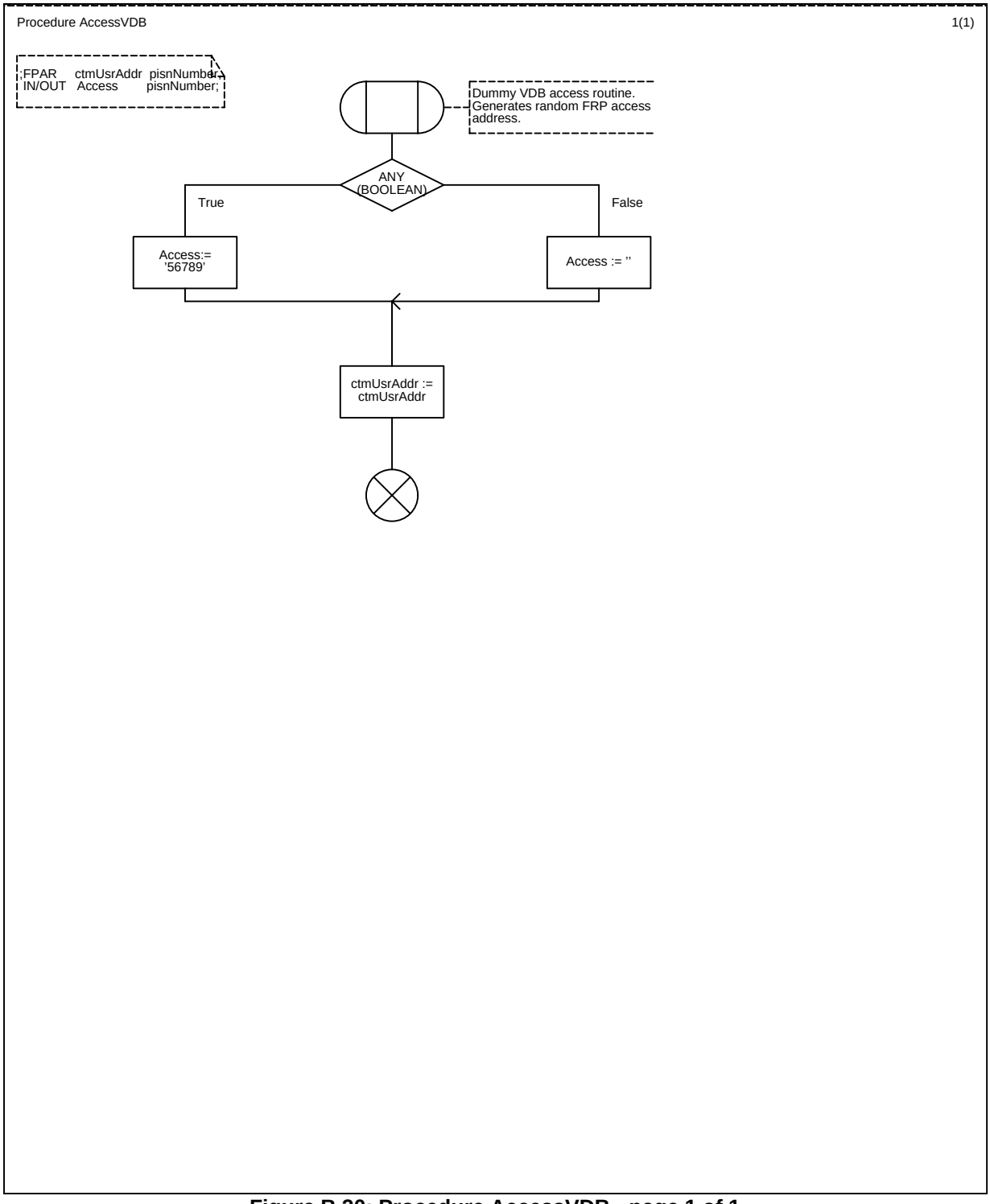

Figure B.20: Procedure AccessVDB - page 1 of 1

Annex C:
The formal model of ANF-CTMI after validation

The following SDL is the formal model of ANF-CTMI on completion of validation. As in the first model (annex B), each of the four physical entities is modelled as a separate process as follows:

- Rerouting PINX — Process *R_PINX;*
- CTMI detect PINX — Process *D_PINX;*
- Home PINX — Process *H_PINX;*
- Visitor PINX — Process *V_PINX*.

The following simple procedures are also included:

- Invoke SS-CFU — Procedure *Invoke_CFU*;
- Retrieve data from HDB — Procedure *AccessHDB*;
- randomly generate boolean values — Procedure *T_or_F;*
- Retrieve data from VDB — Procedure *AccessVDB*.

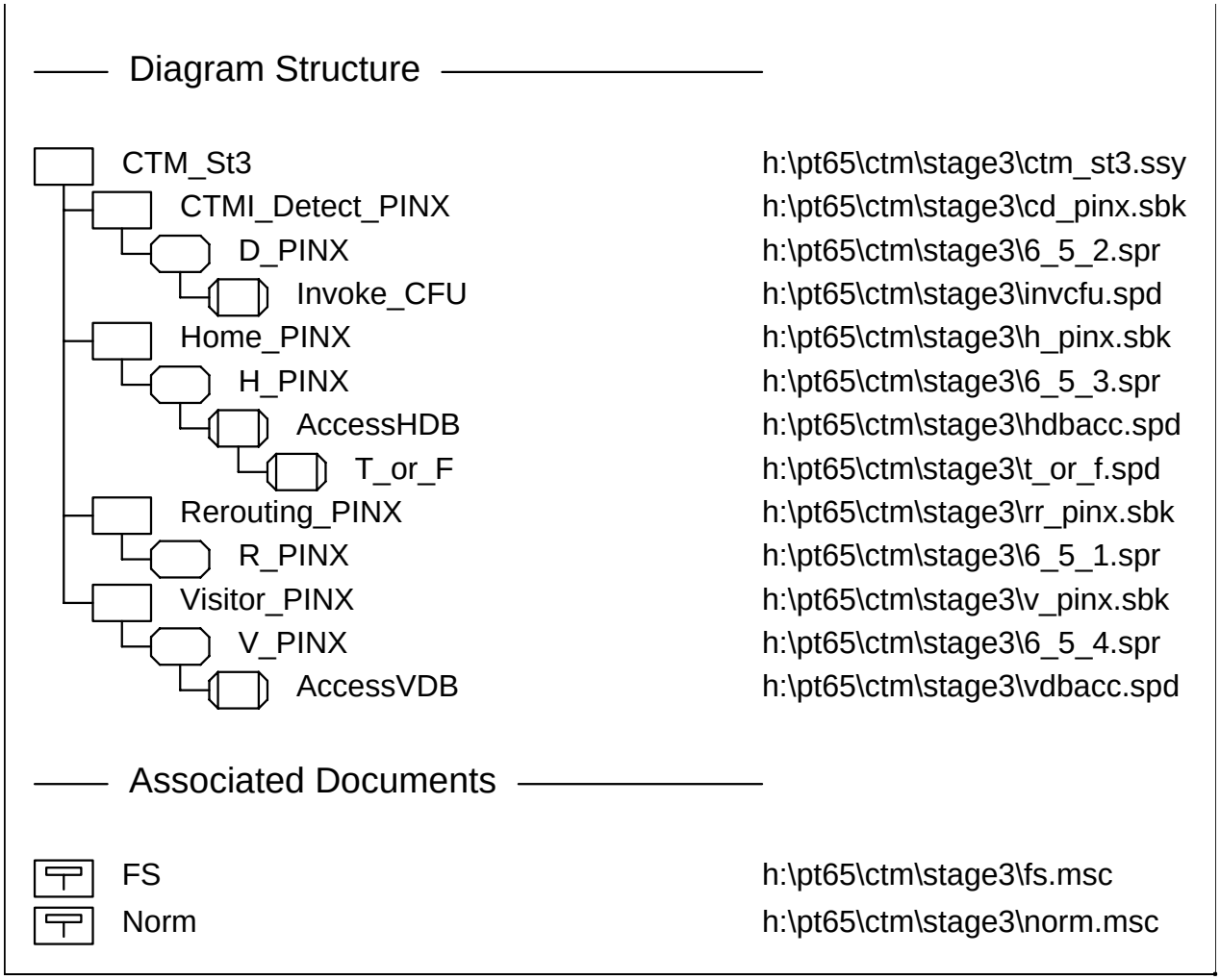

Figure C.1: ANF-CTMI SDL model organizer view

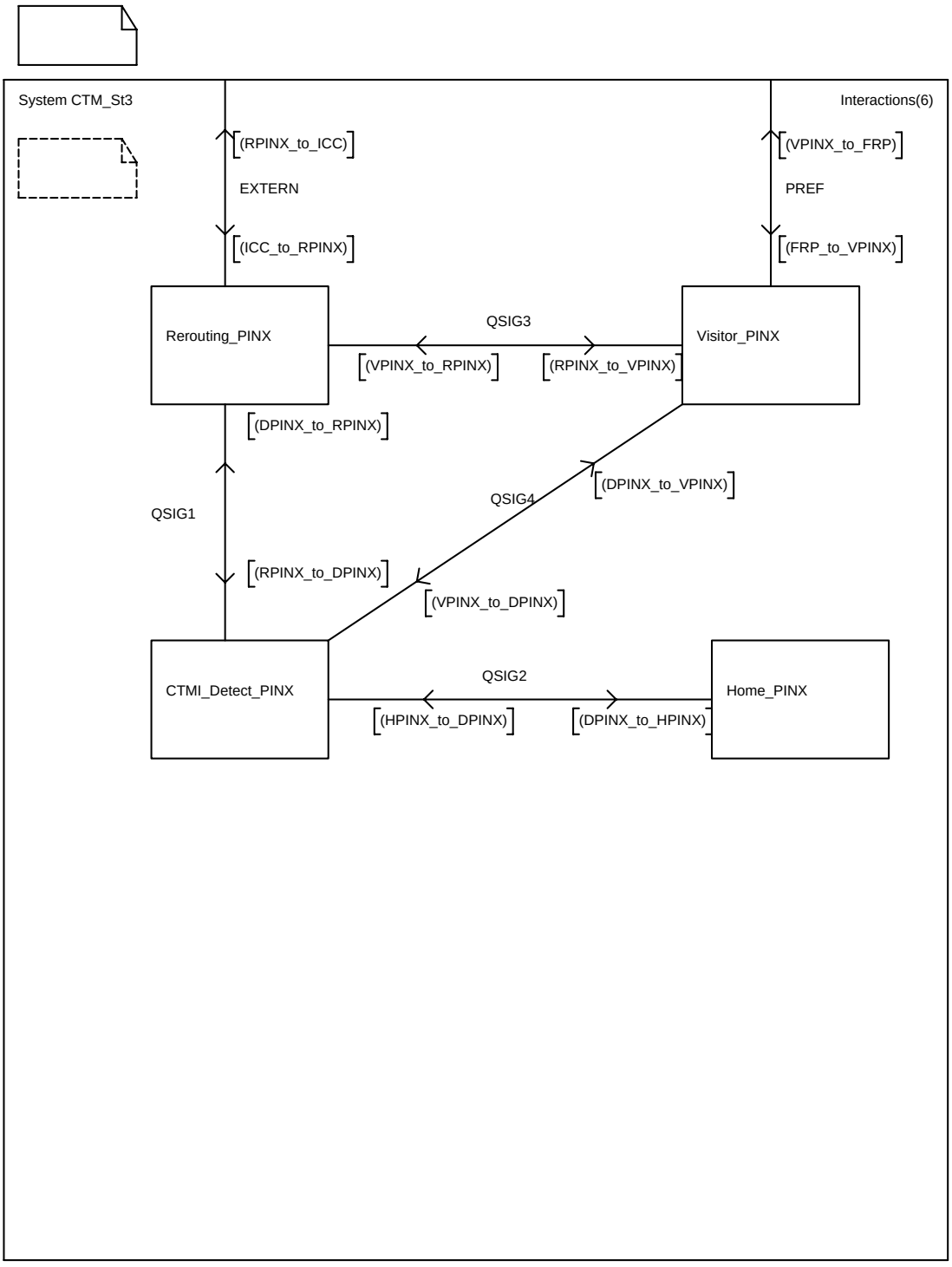

Figure C.2: ANF-CTMI SDL system

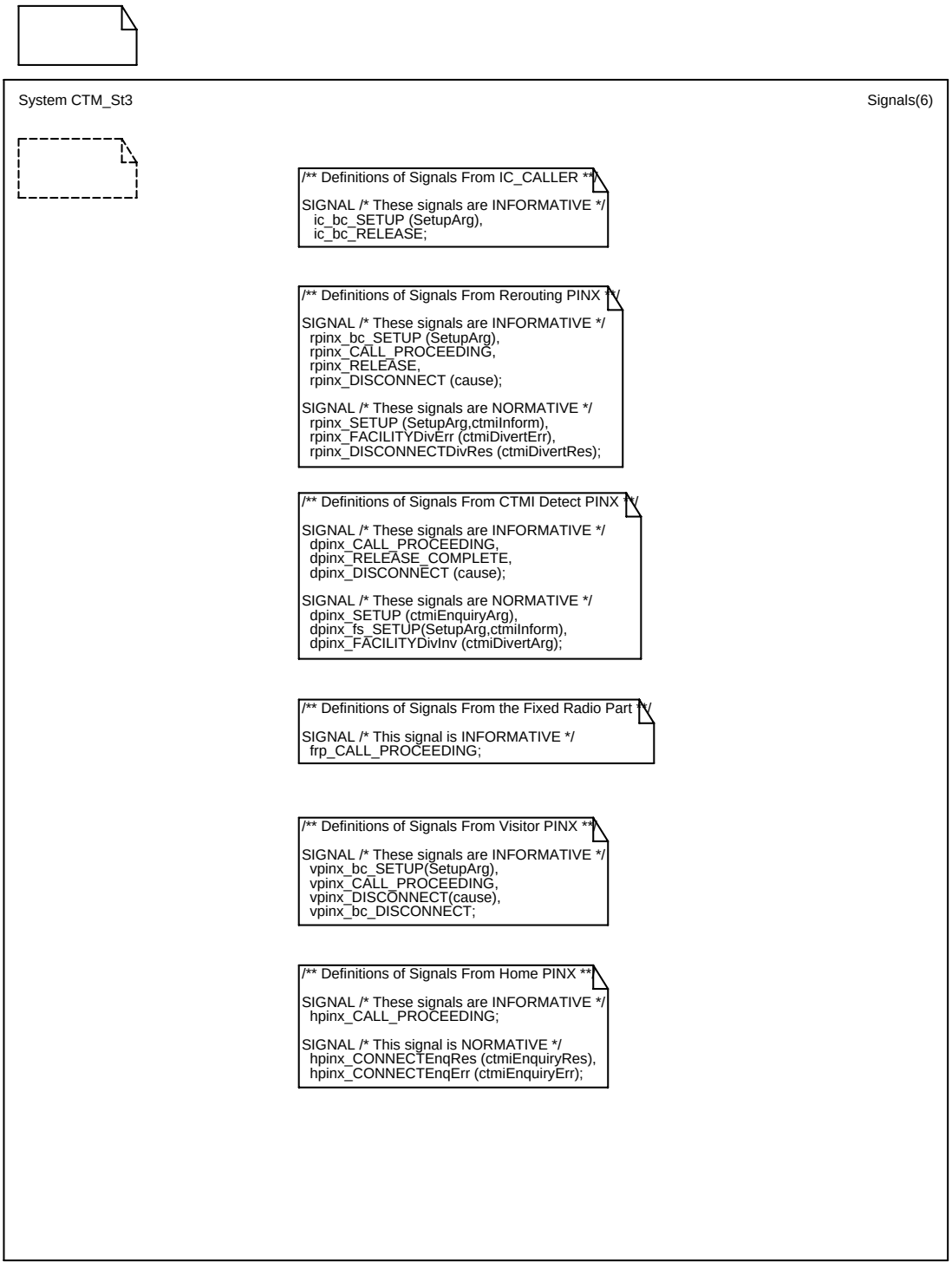

Figure C.3: ANF-CTMI signal definitions

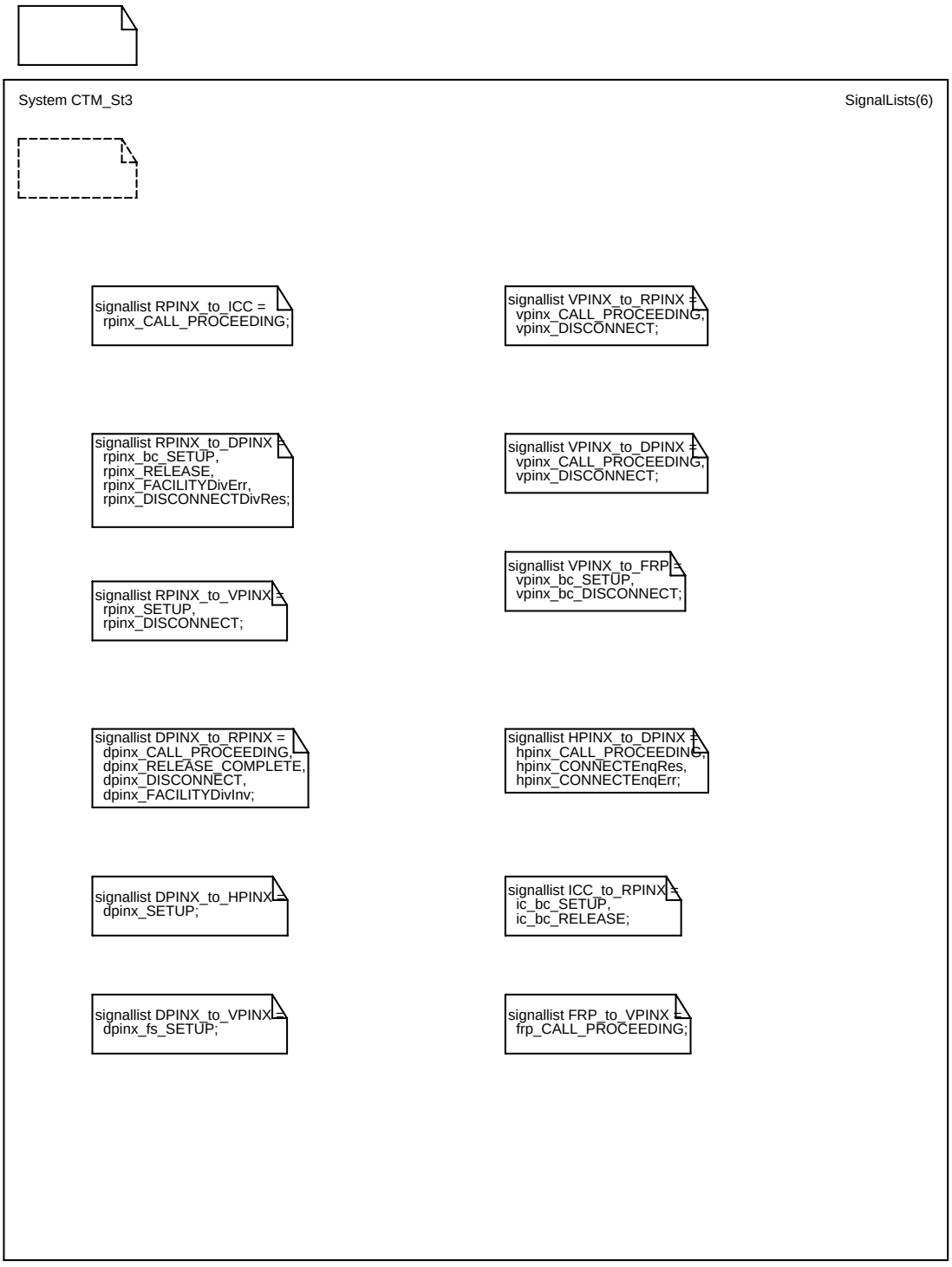

Figure C.4: ANF-CTMI signallist definitions

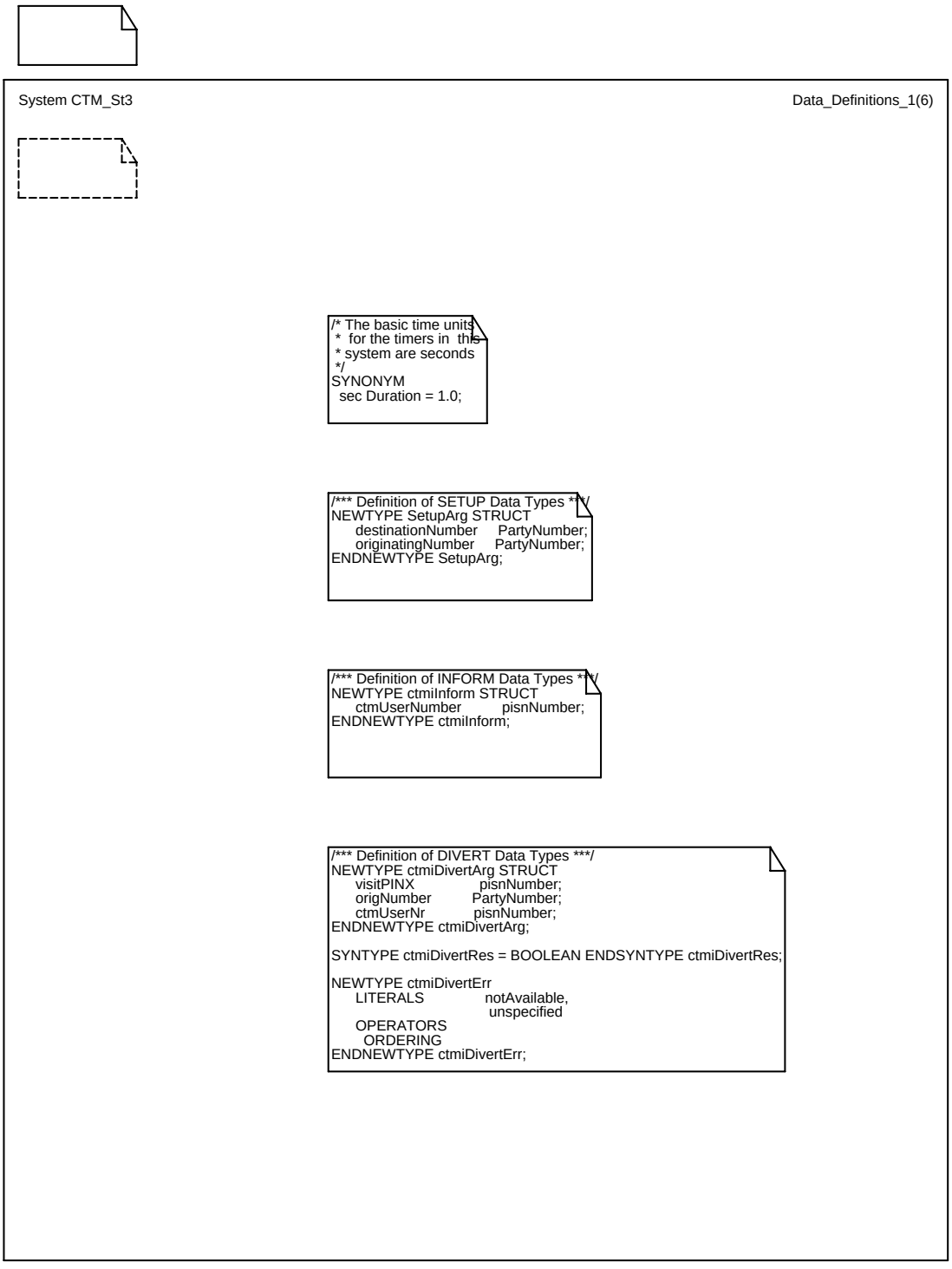

Figure C.5: ANF-CTMI Data types - page 1 of 3

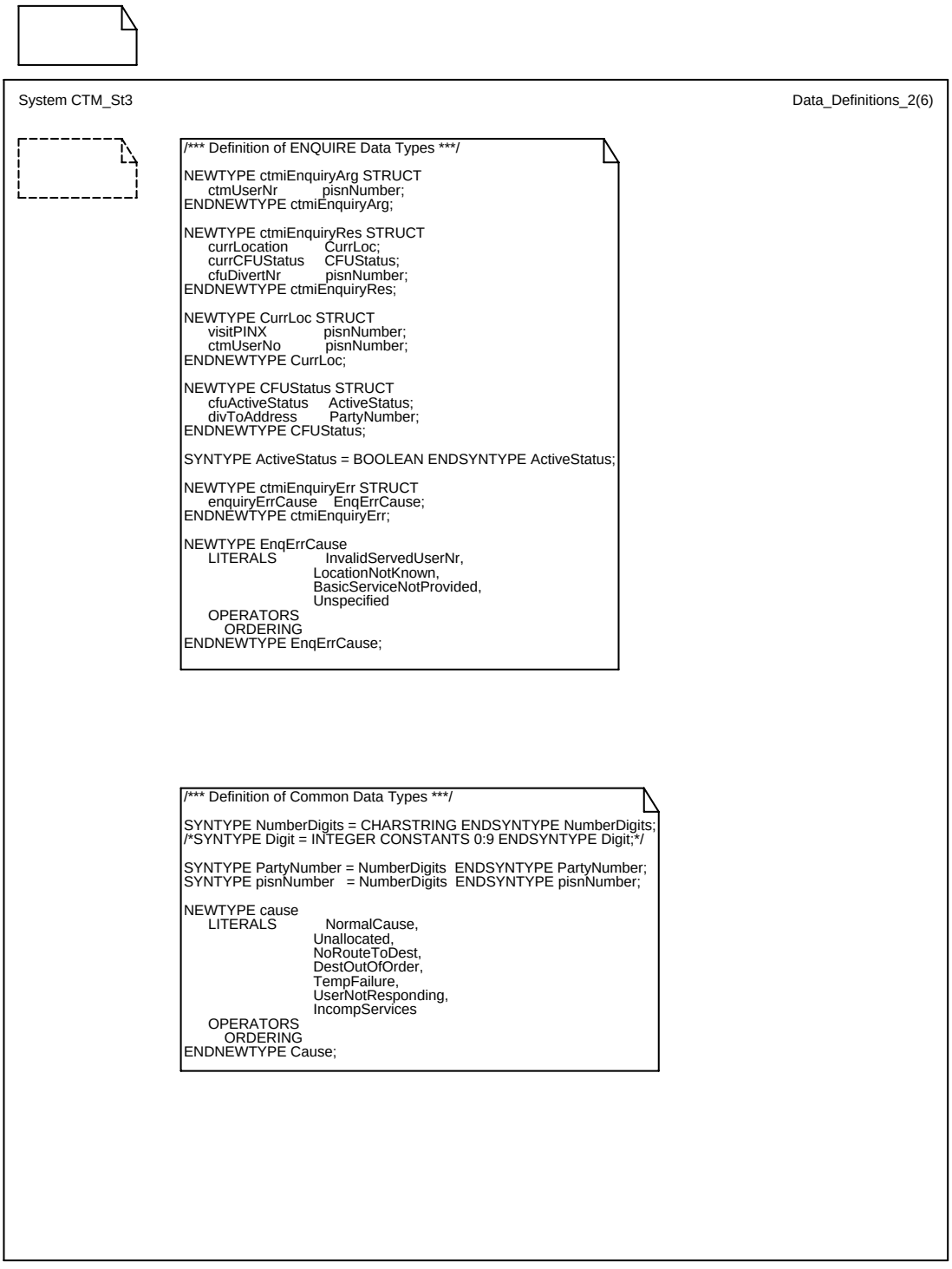

Figure C.6: ANF-CTMI Data type - page 2 of 3

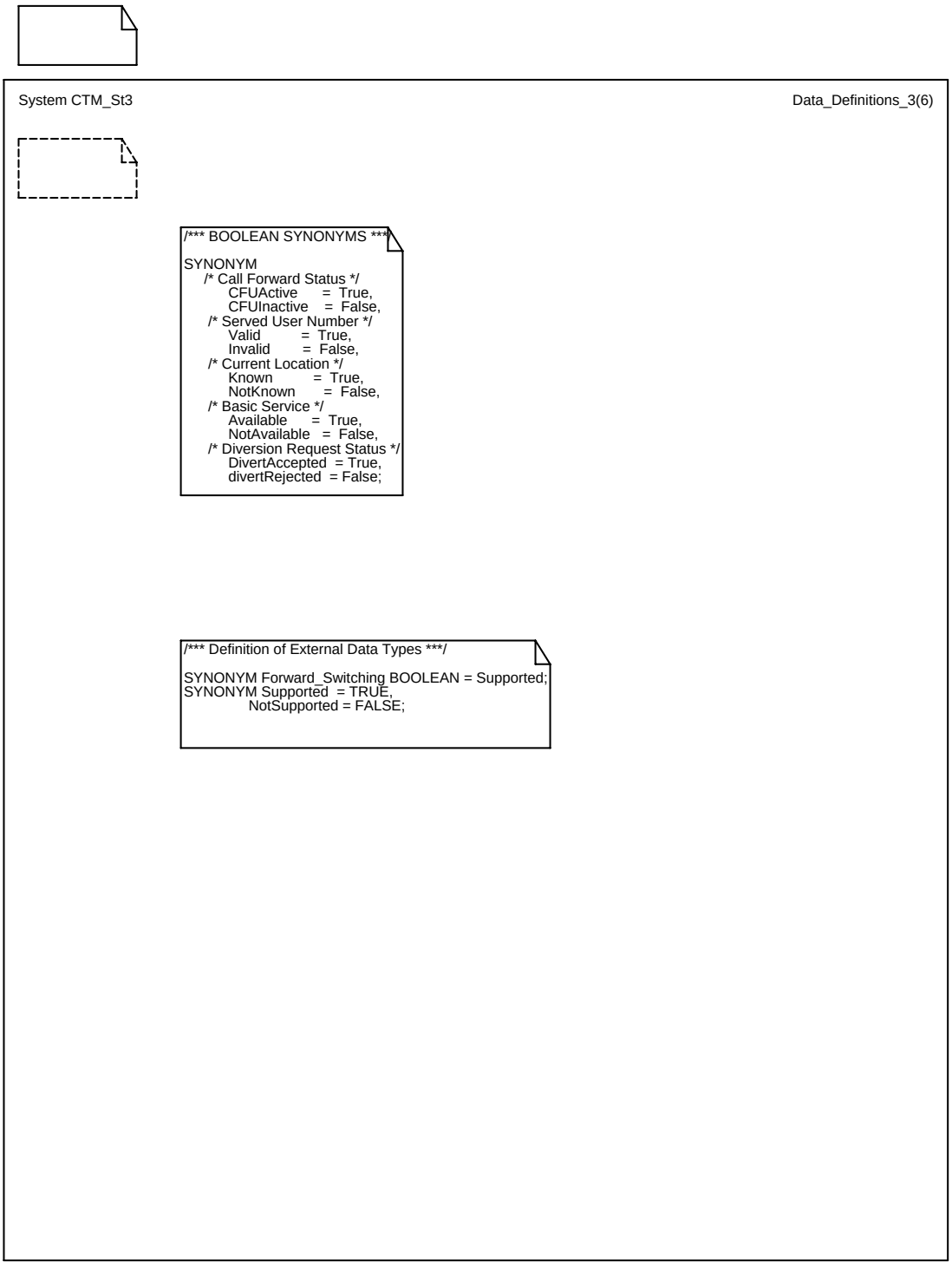

Figure C.7: ANF-CTMI Data type - page 3 of 3

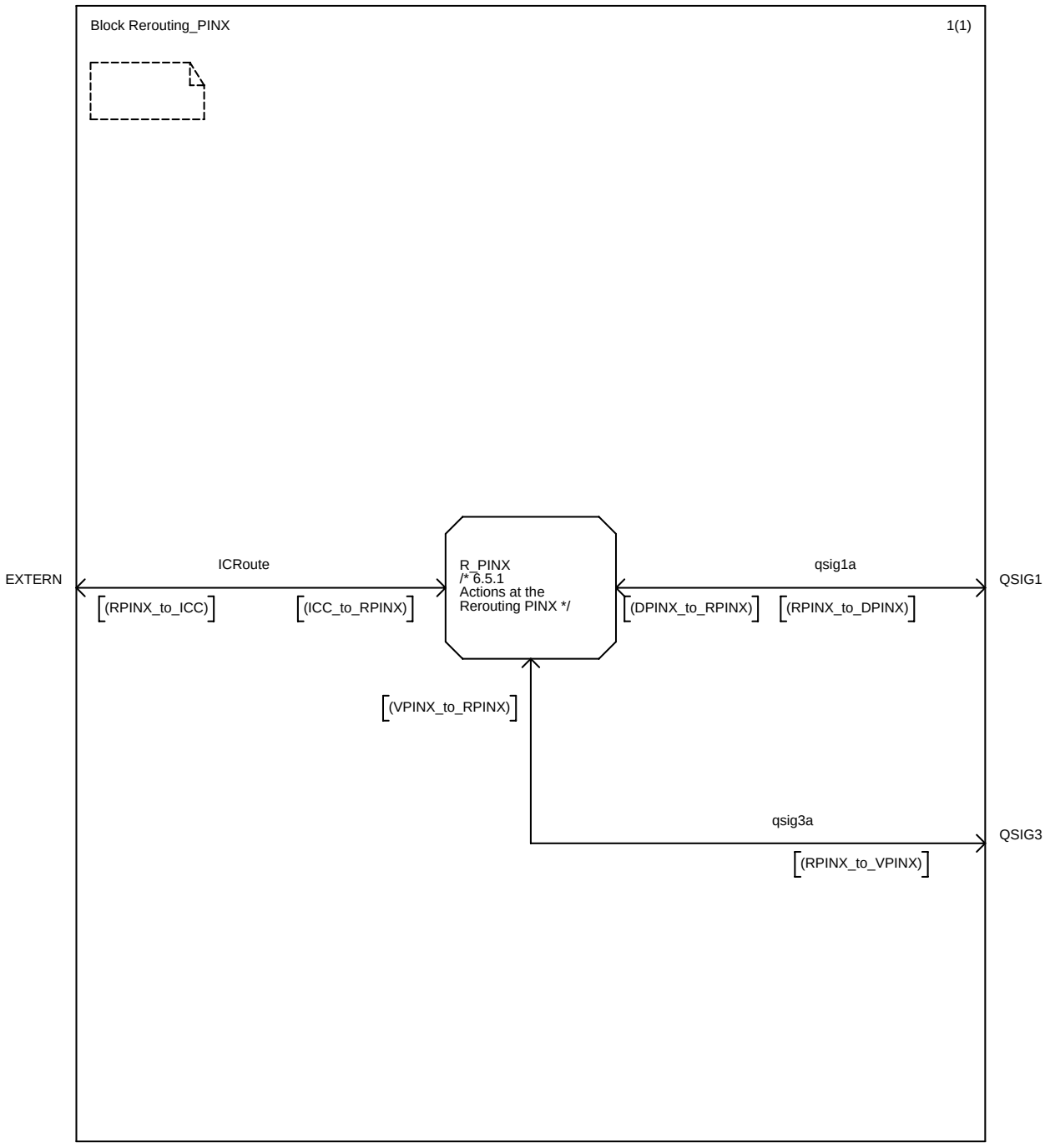

Figure C.8: ANF-CTMI Rerouting PINX Block

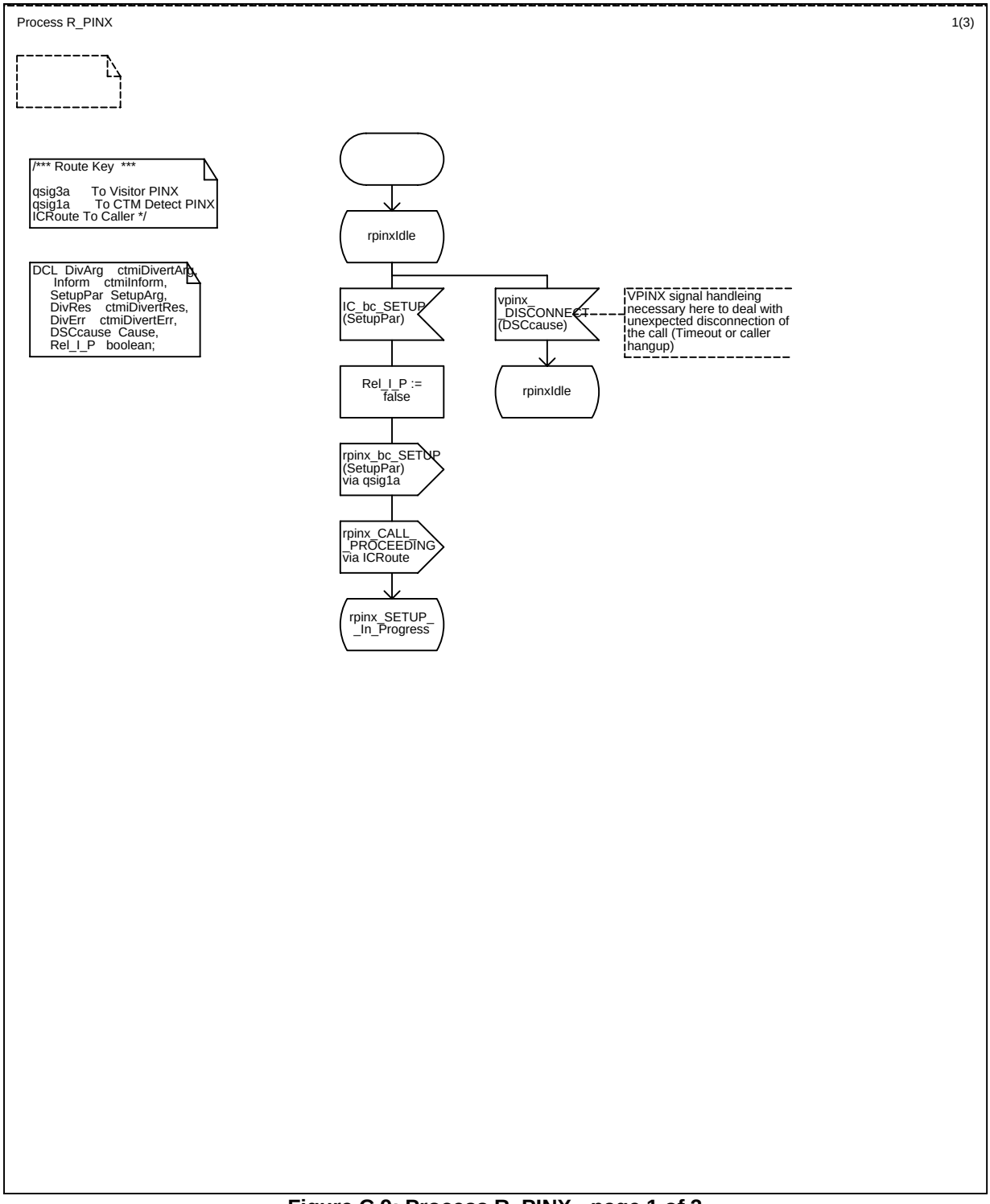

Figure C.9: Process R_PINX - page 1 of 3

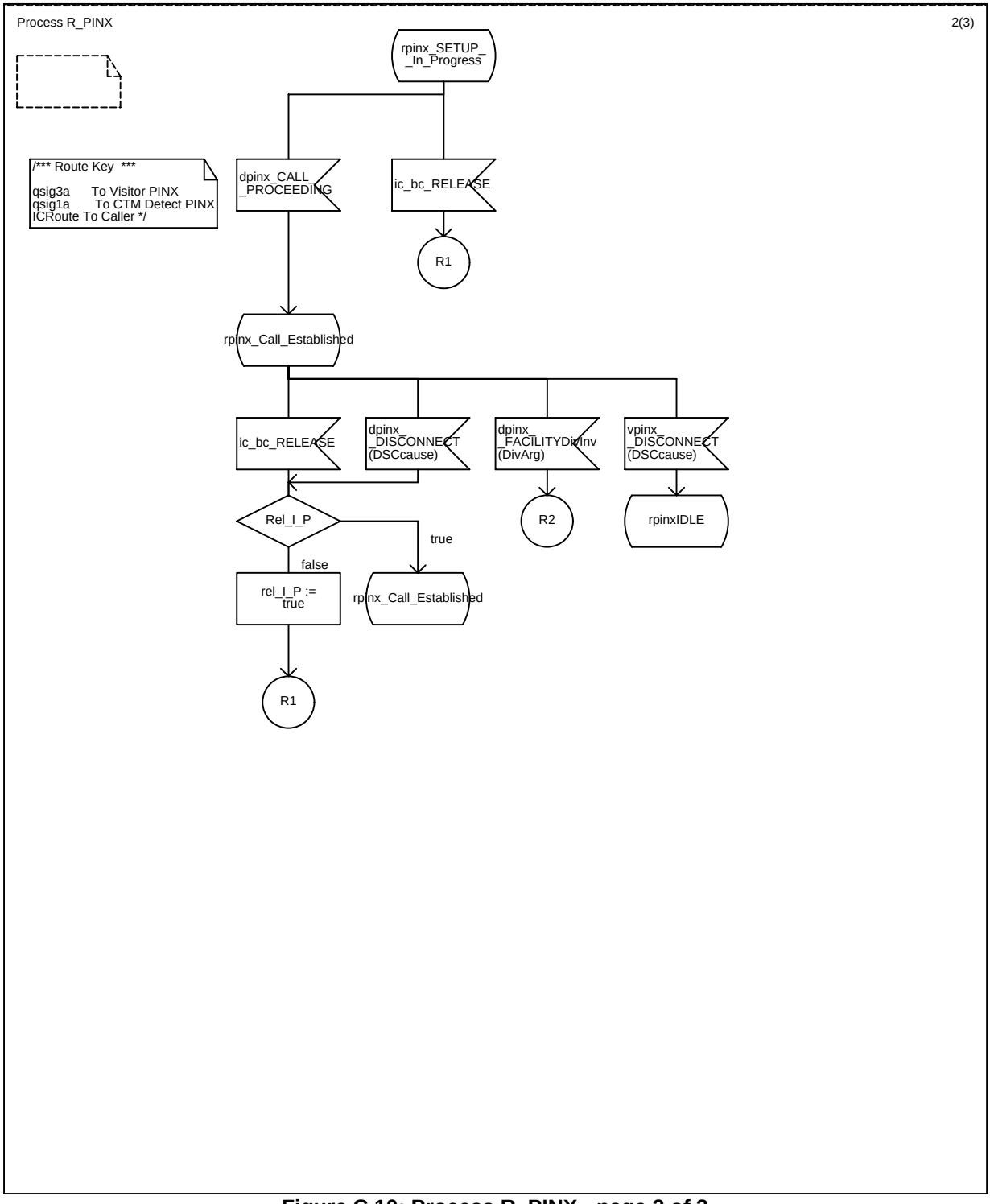

Figure C.10: Process R_PINX - page 2 of 3

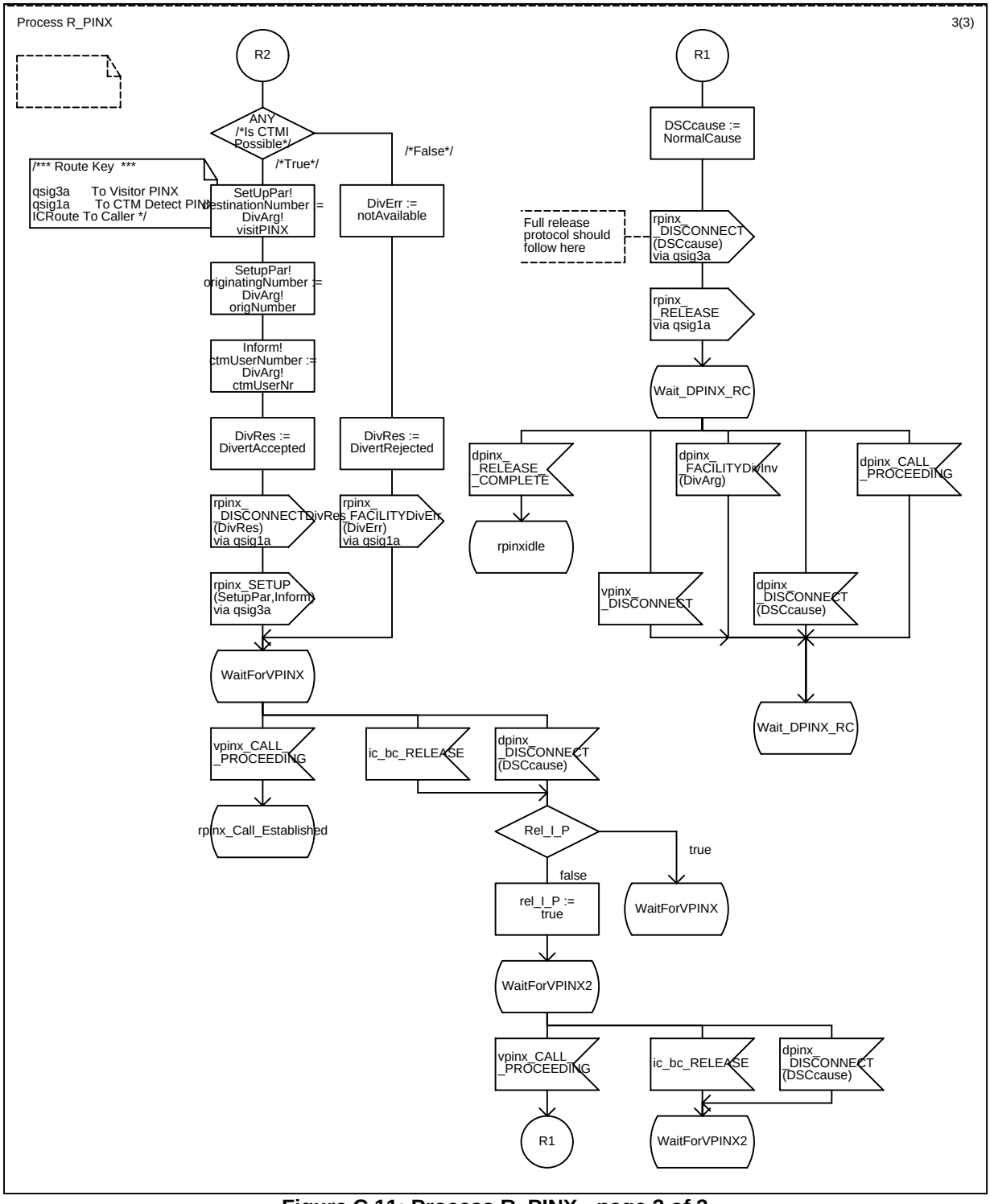

Figure C.11: Process R_PINX - page 3 of 3

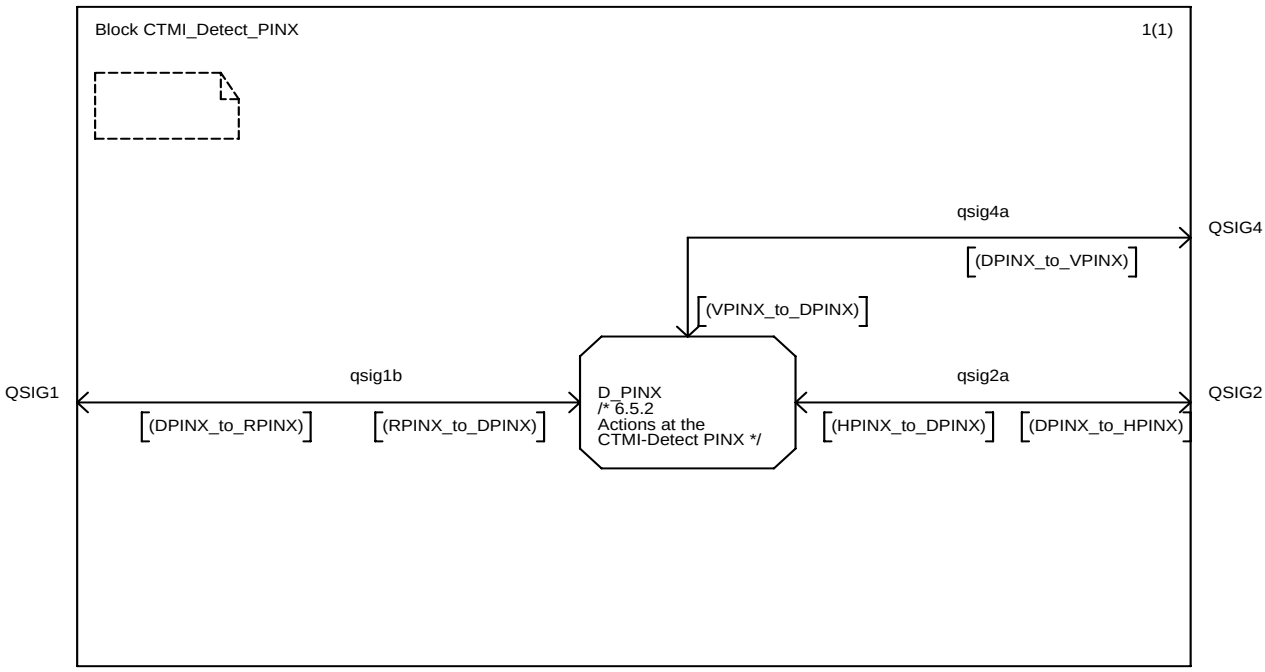

Figure C.12: ANF-CTMI CTMI Detect PINX Block

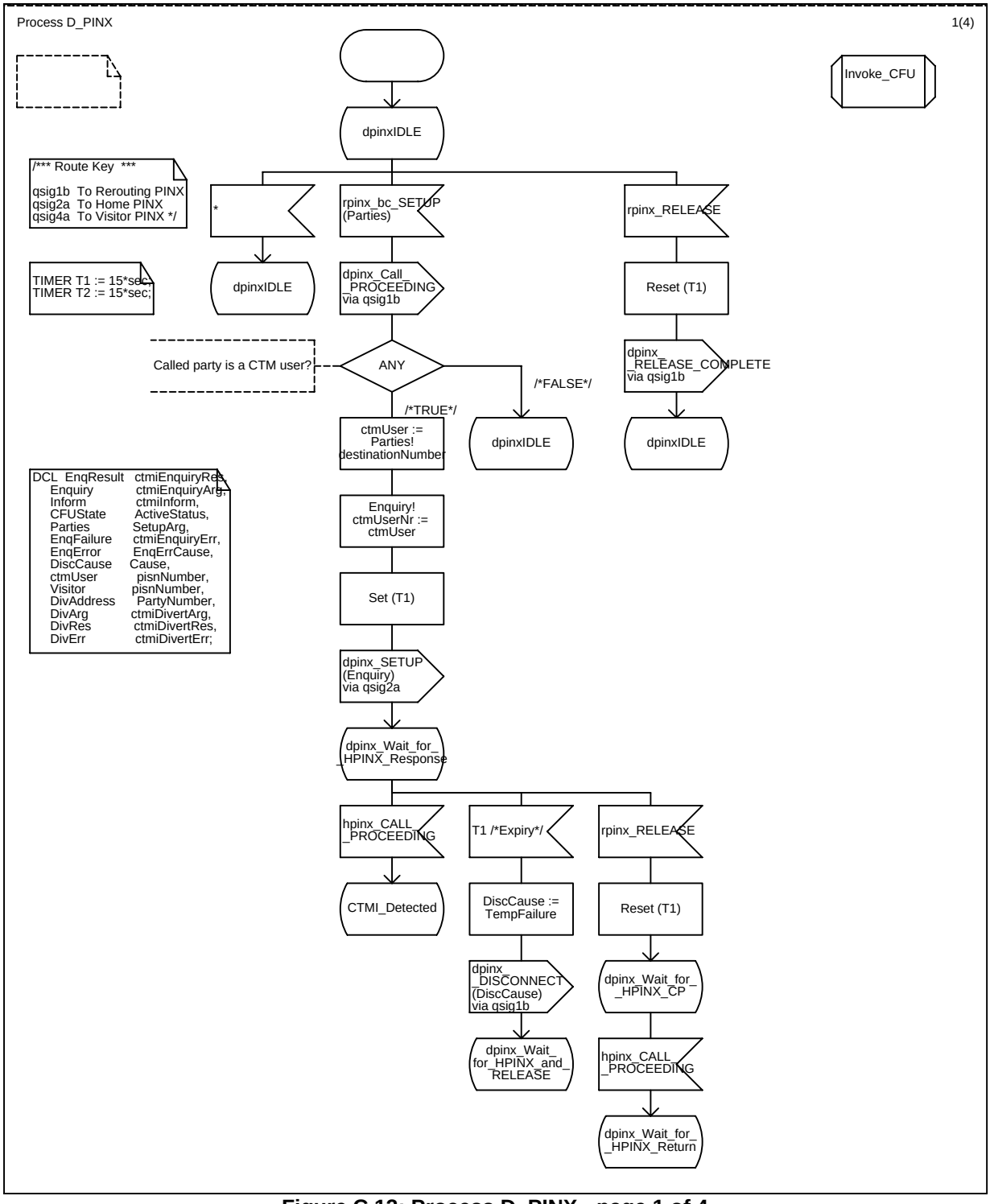

Figure C.13: Process D_PINX - page 1 of 4

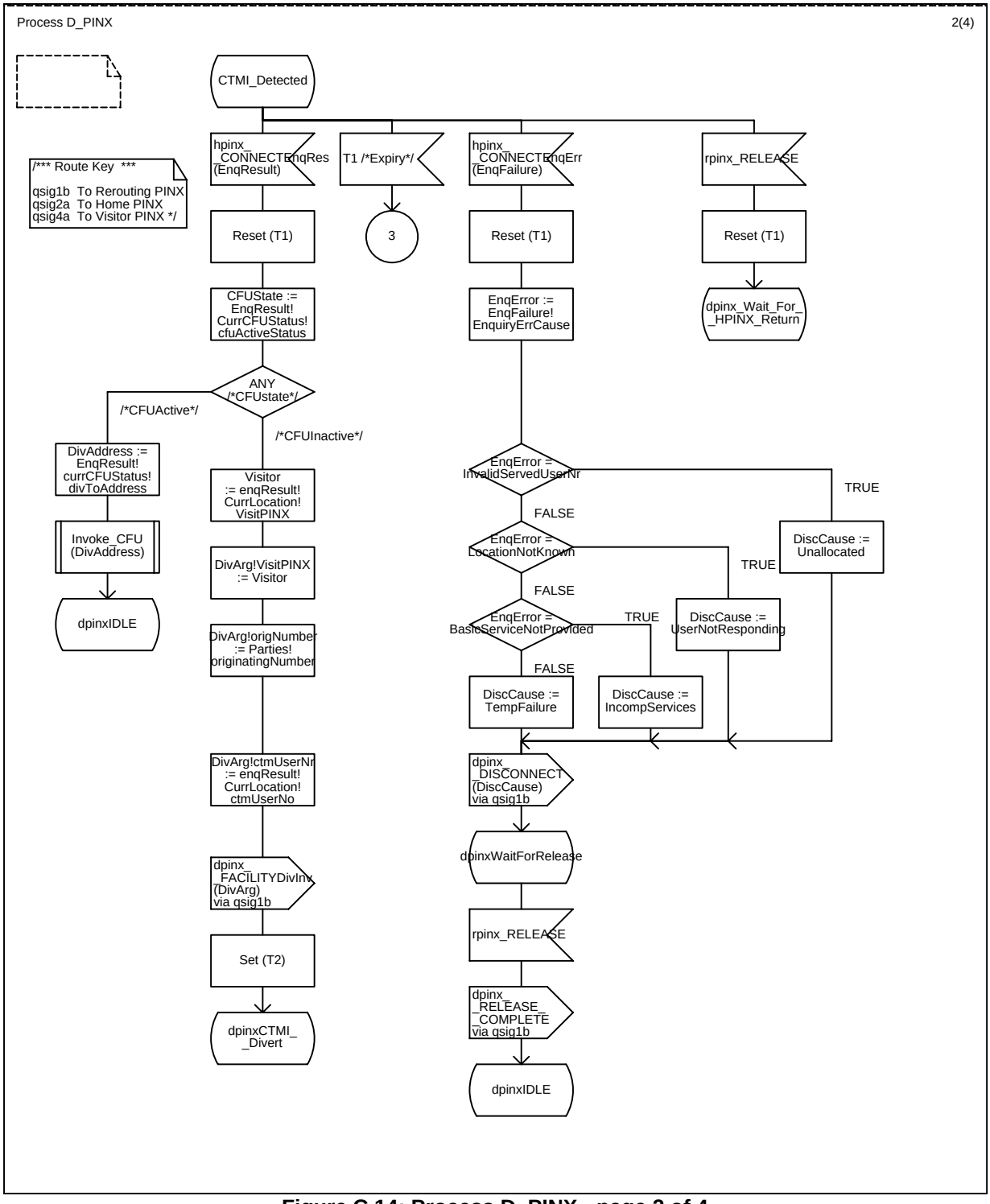

Figure C.14: Process D_PINX - page 2 of 4

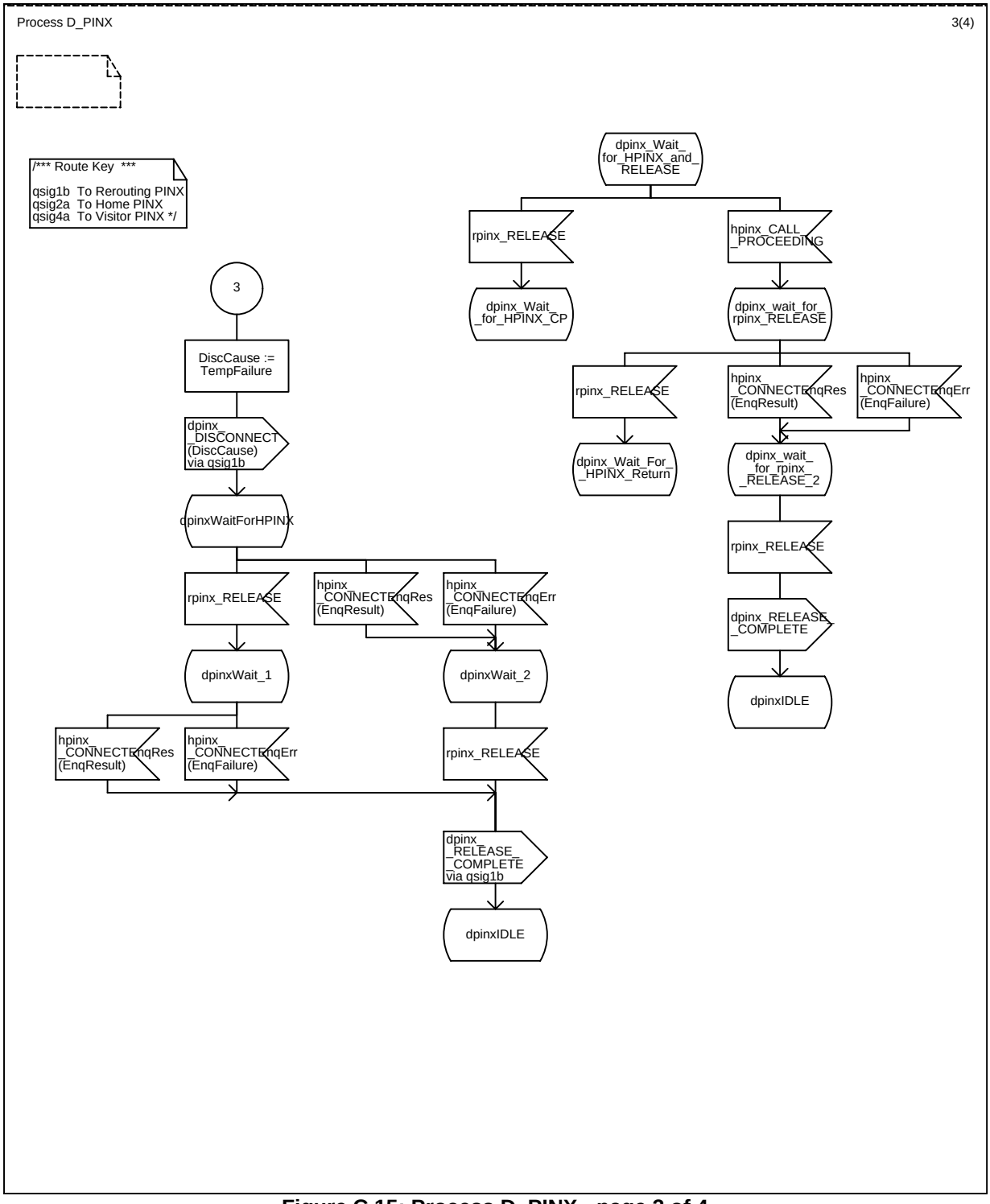

Figure C.15: Process D_PINX - page 3 of 4

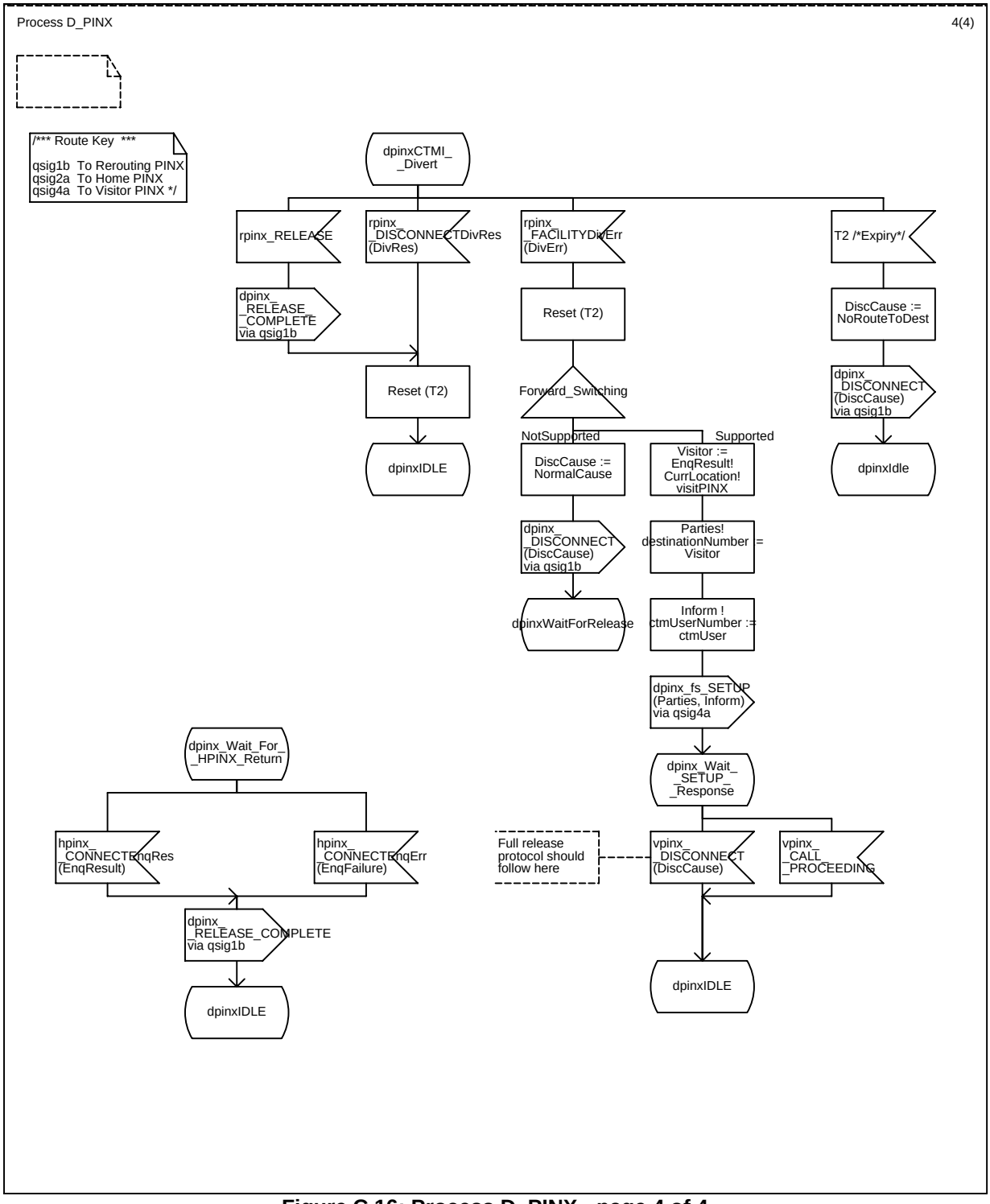

Figure C.16: Process D_PINX - page 4 of 4

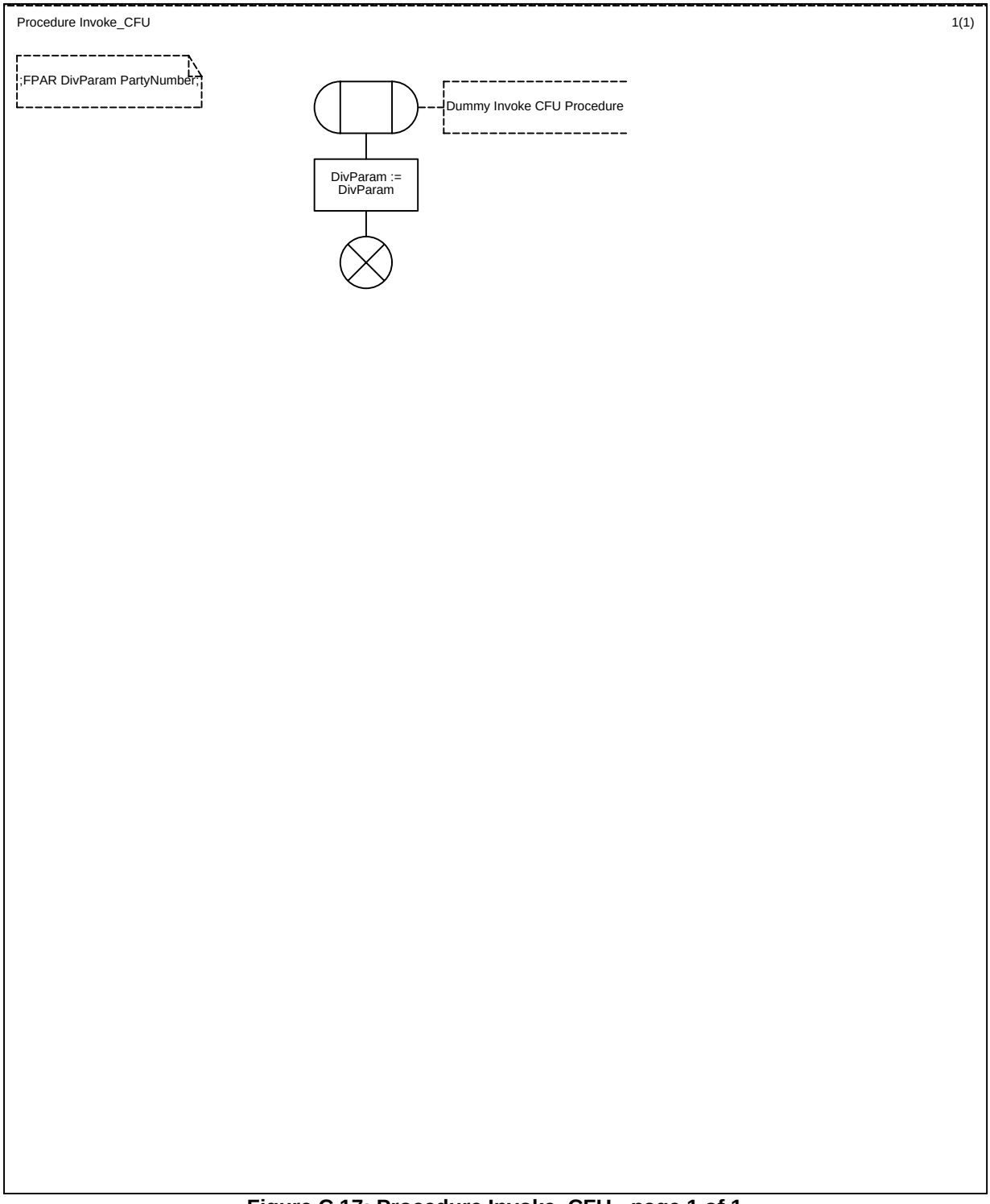

Figure C.17: Procedure Invoke_CFU - page 1 of 1

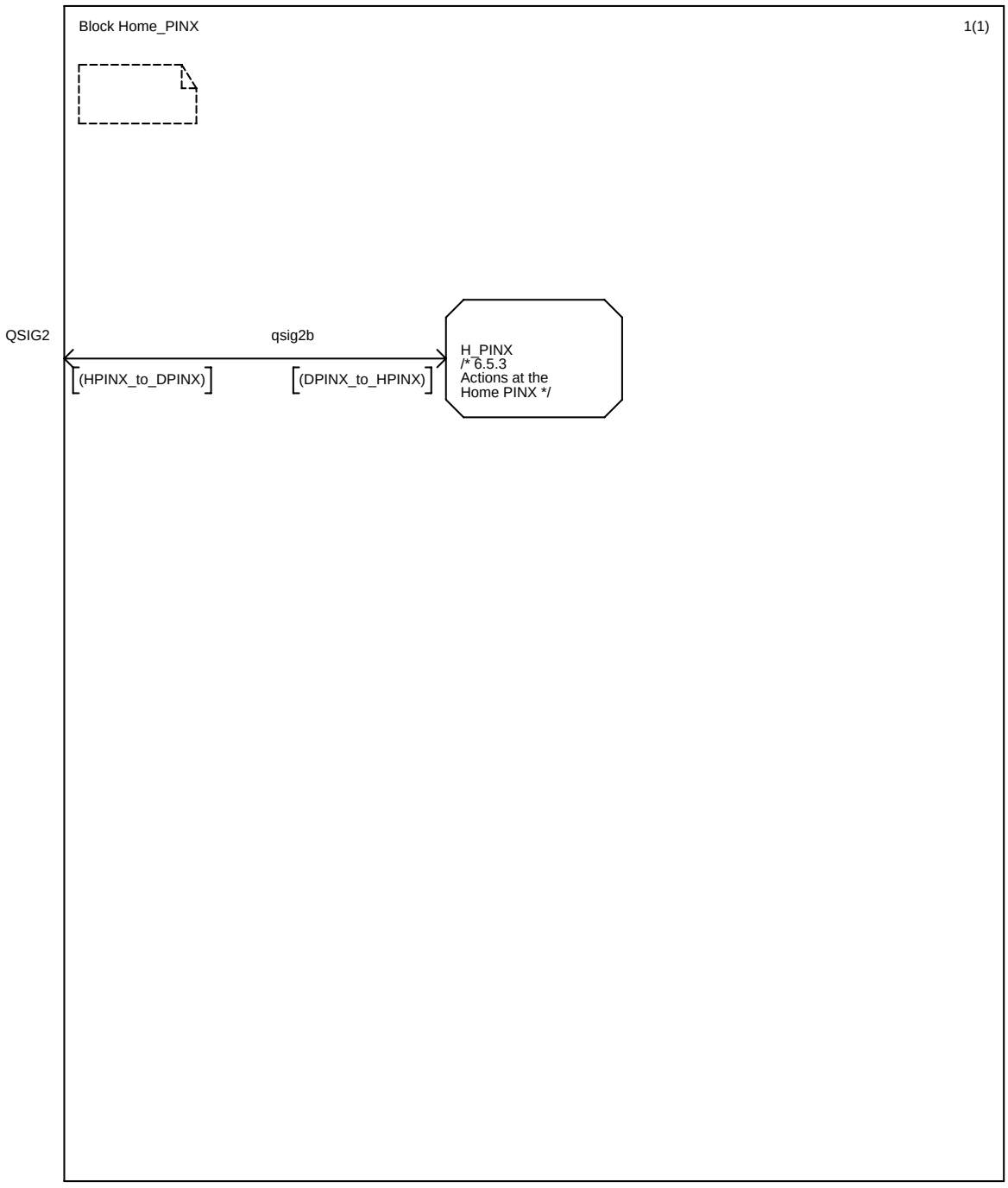

Figure C.18: ANF-CTMI Home PINX Block

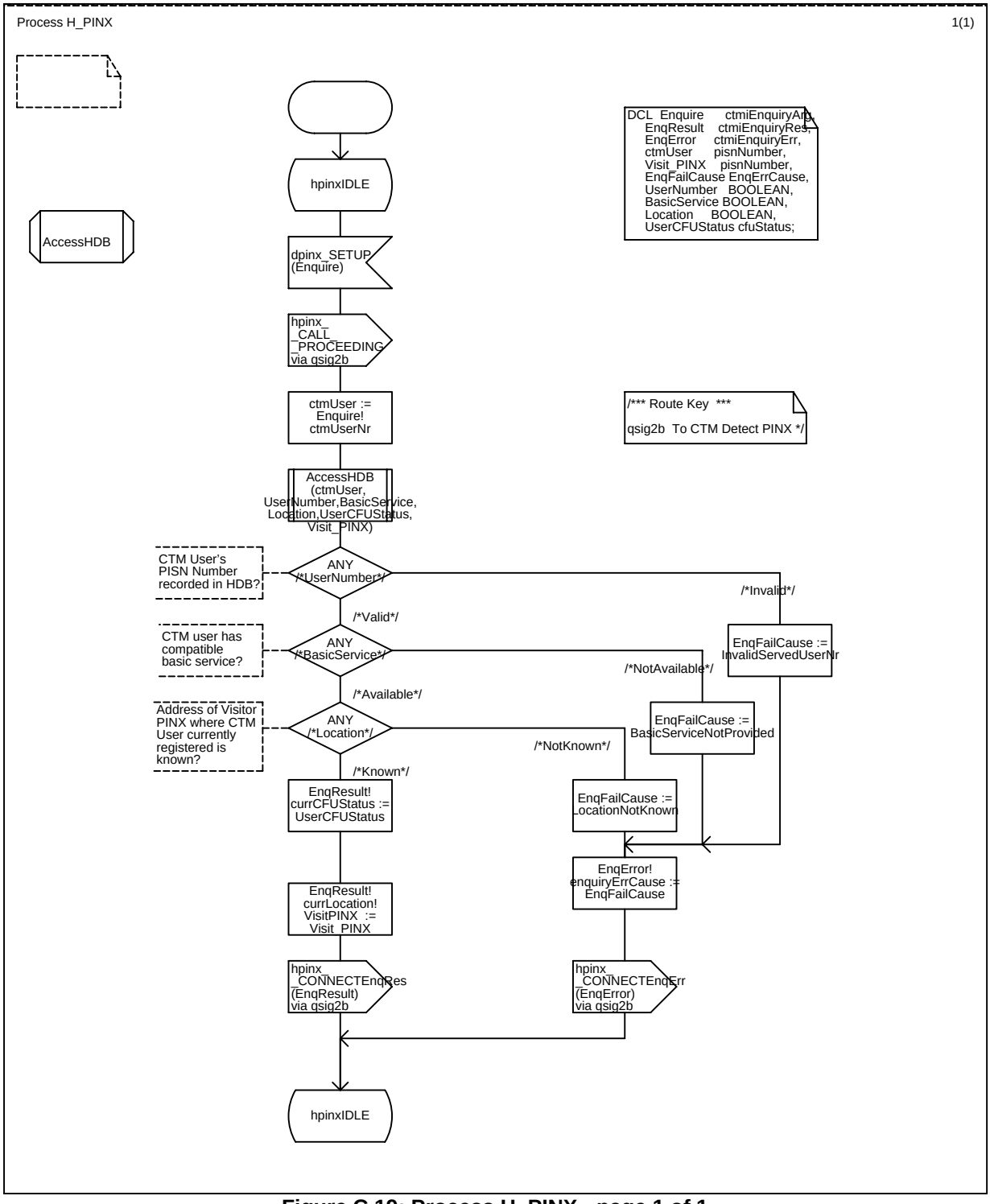

Figure C.19: Process H_PINX - page 1 of 1

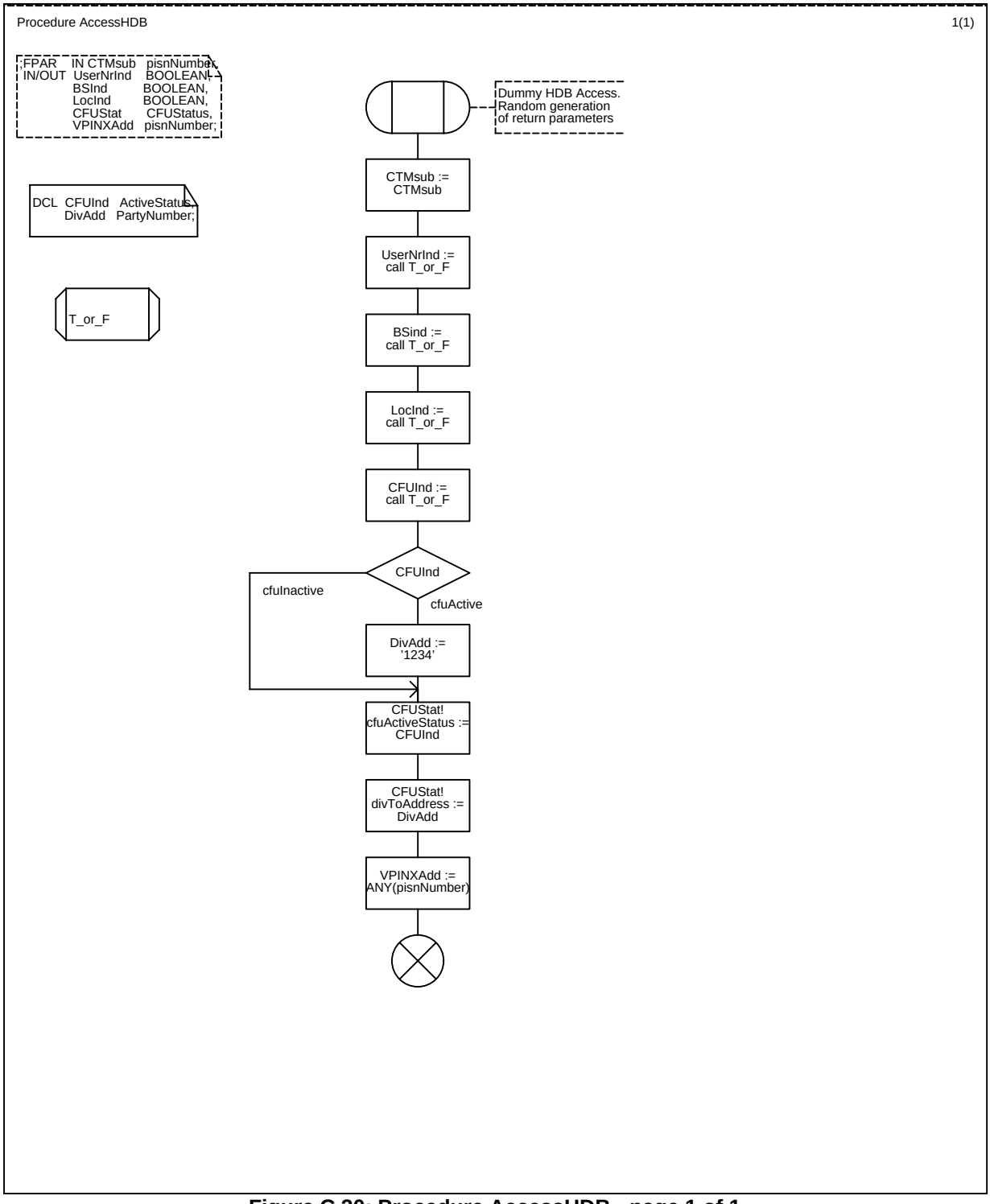

Figure C.20: Procedure AccessHDB - page 1 of 1

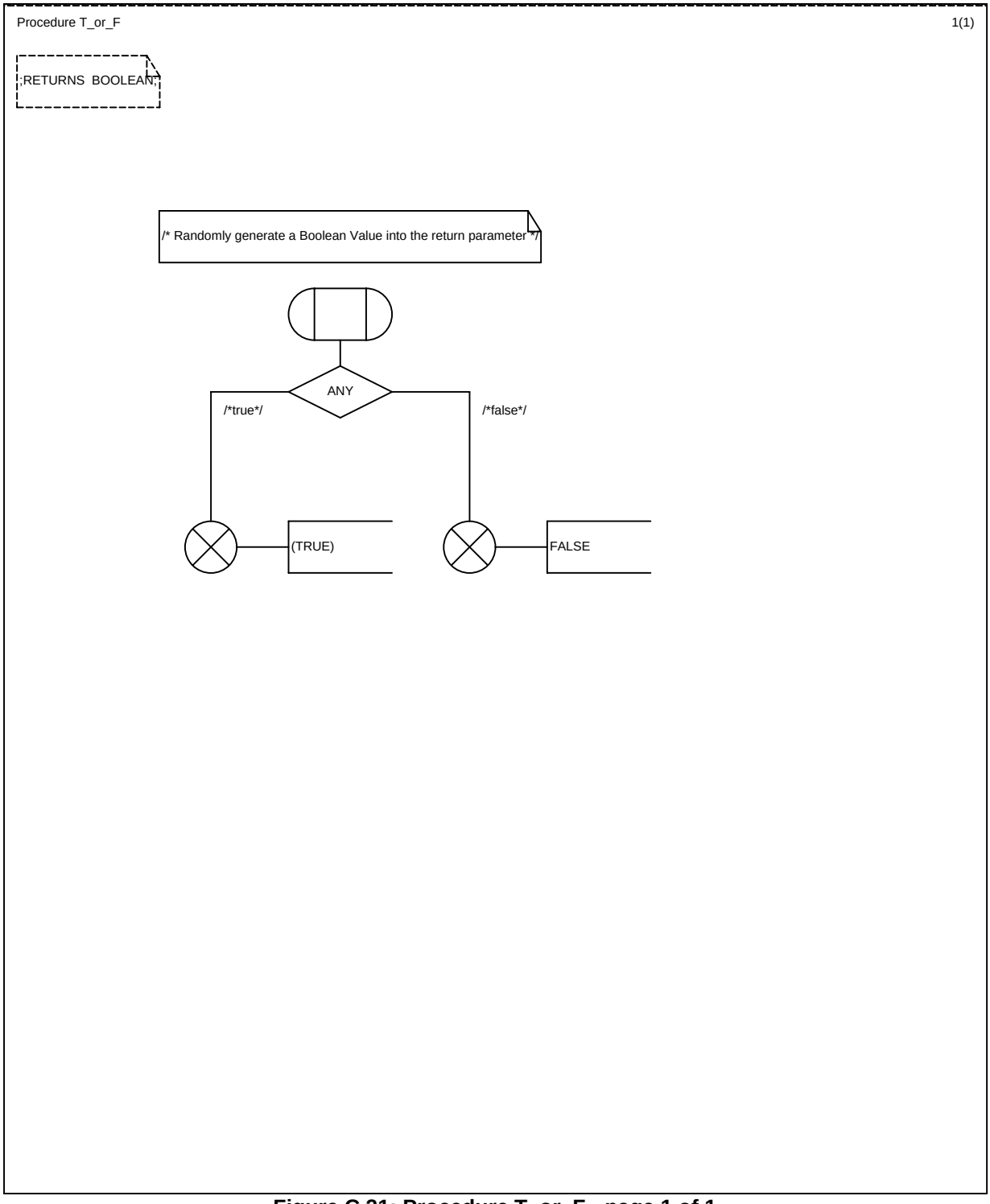

Figure C.21: Procedure T_or_F - page 1 of 1

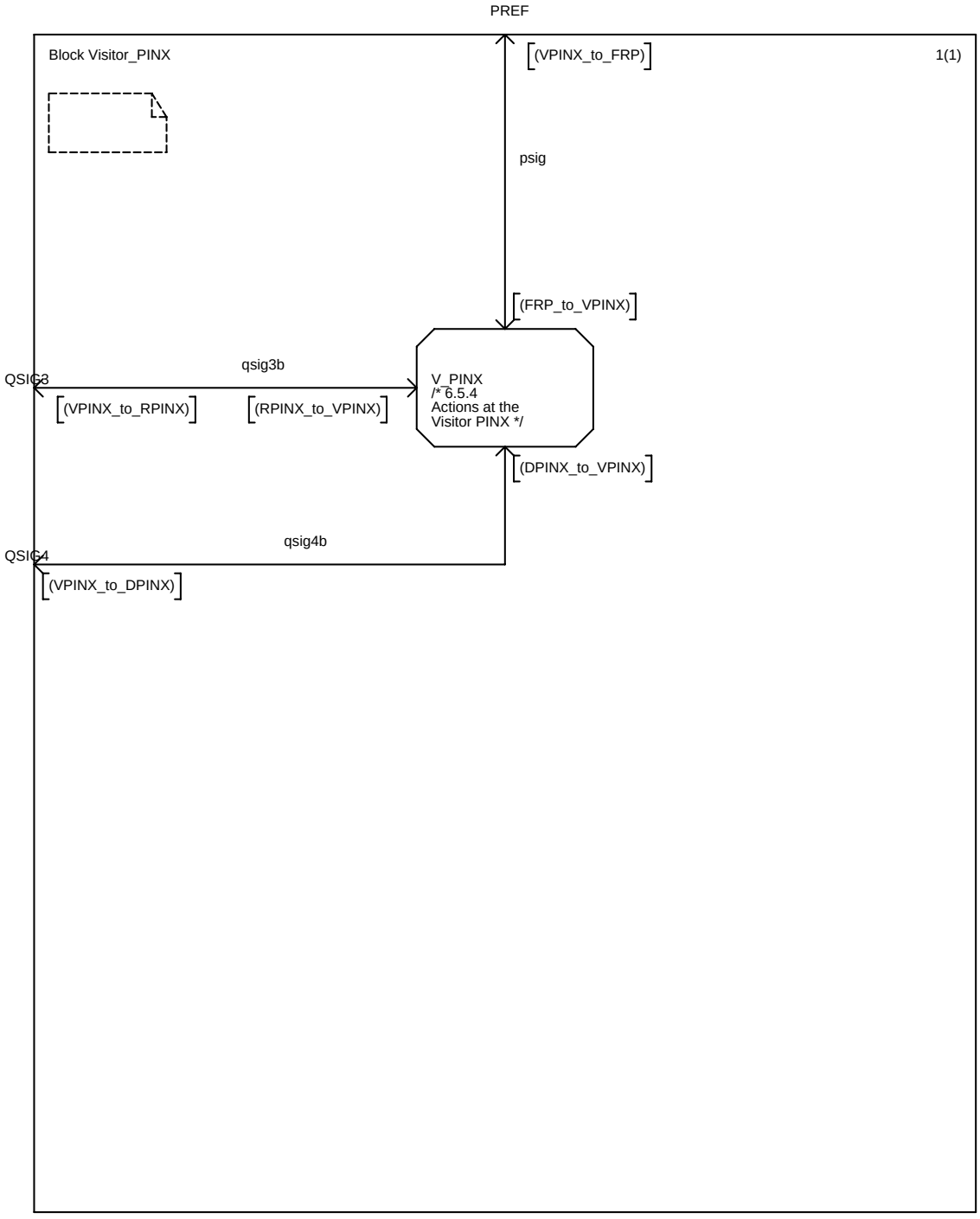

Figure C.22: ANF-CTMI Visitor PINX Block

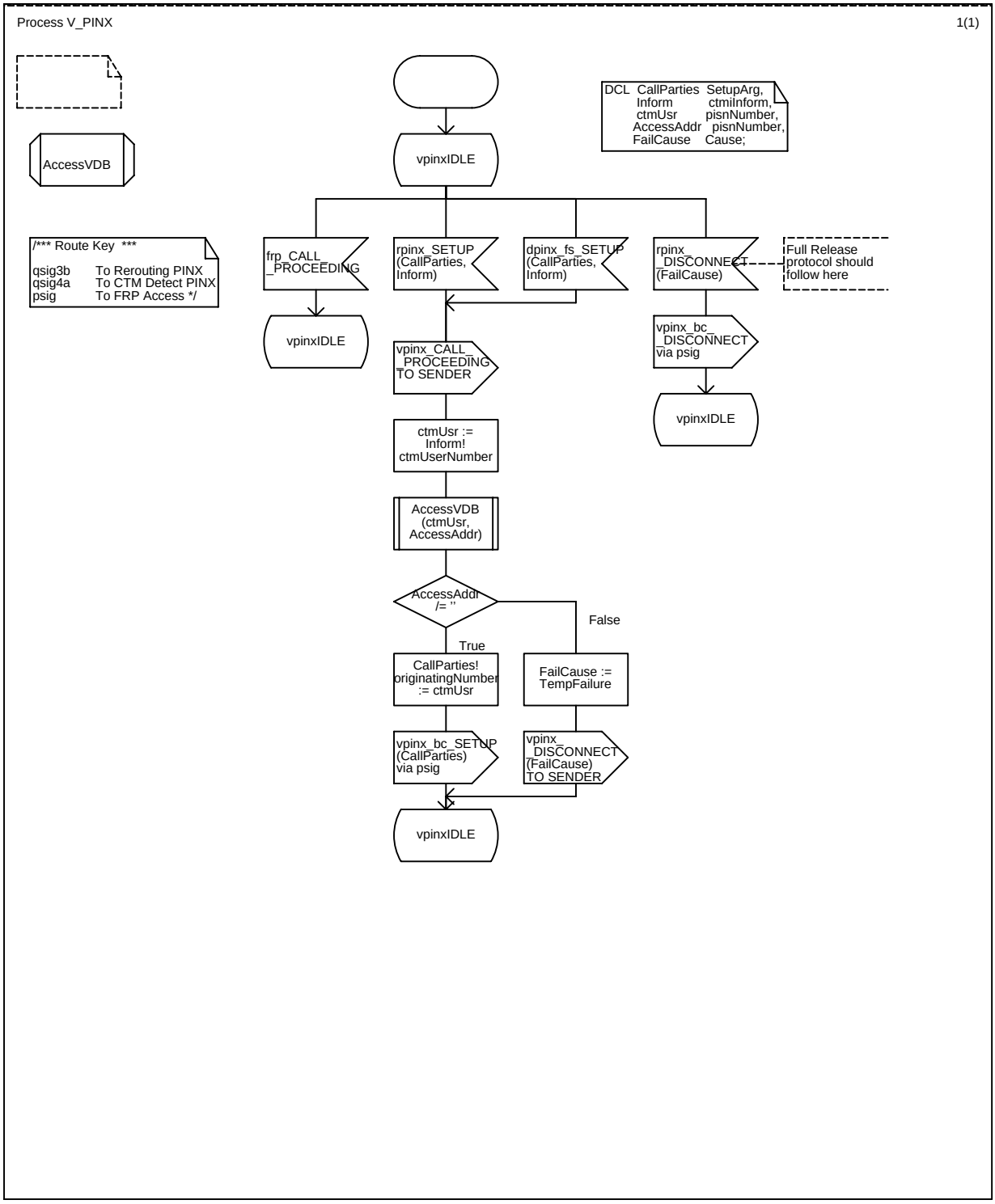

Figure C.23: Process V_PINX - page 1 of 1

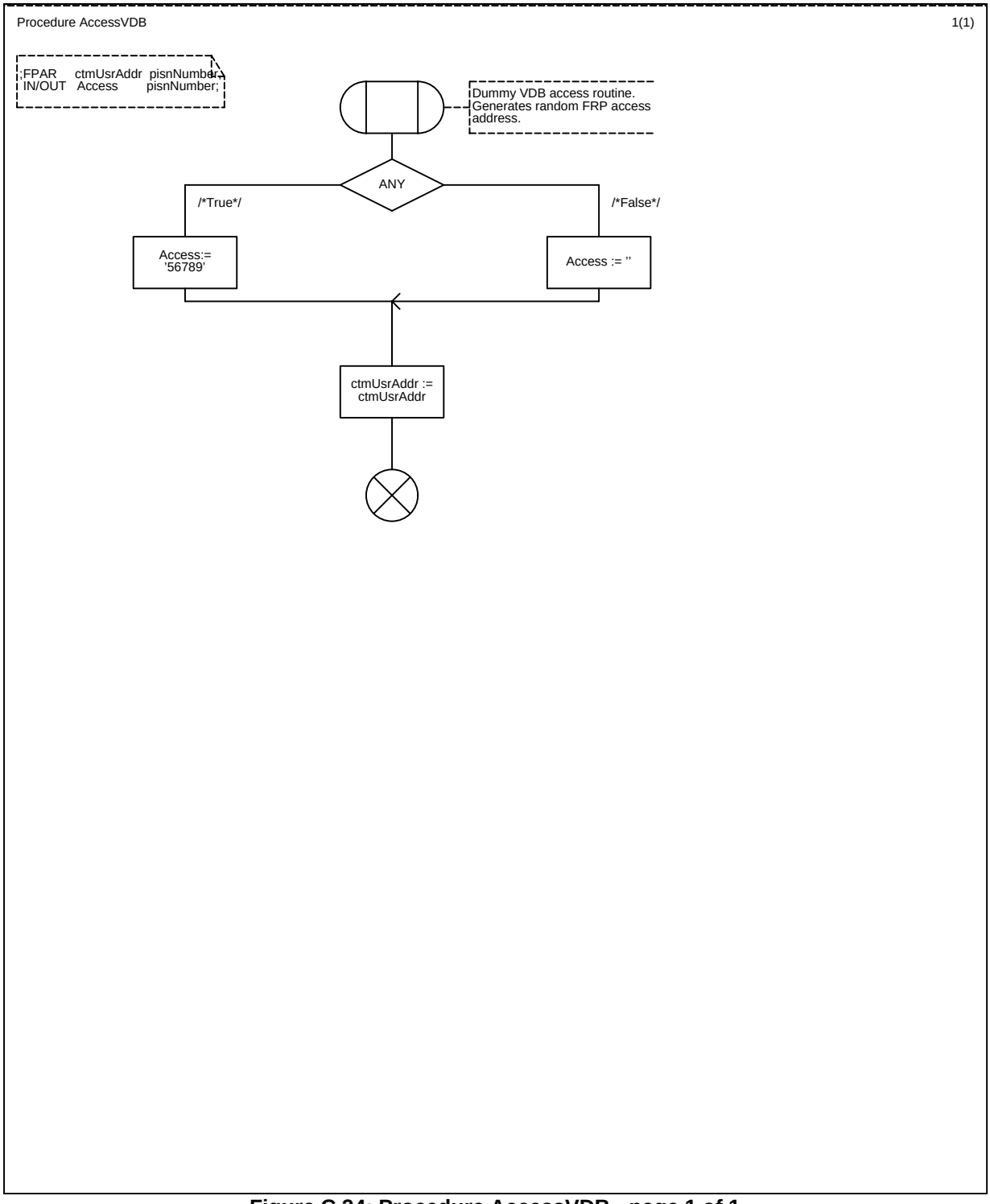

Figure C.24: Procedure AccessVDB - page 1 of 1

History

Document history		
V1.1.1	May 1997	Publication

ISBN 2-7437-1422-0
Dépôt légal : Mai 1997