Automatic Test Case Generation with SAMSTAG - A Test Suite for the Initiator Process of the Inres Protocol

Daniel Toggweiler Robert Nahm

Institut für Informatik
Universität Bern
Länggassstr. 51
CH-3012 Bern

toggweil@iam.unibe.ch nahm@iam.unibe.ch

IAM-93-026

December 1993

Abstract

Conformance testing is the basis for the harmonical cooperation of communication products. By means of test suites implementations are tested against protocol specifications in their most important parts for conformance. The generation of test suites is a big problem. Within the research project *Conformance Testing – A tool for the Generation of Test Cases*[1] at University of Berne, the test case generator SAMSTAG has been developed. SAMSTAG is the abbreviation for *Sdl And Msc baSed Test cAse Generator*. SAMSTAG generates a single test case for a protocol specification given by an SDL description and a test purpose given by an MSC [GHN93b]. The scope of this paper is to demonstrate the application of SAMSTAG . As an example the Inres protocol is chosen [Hog89]. We generate a test suite for the Initiator of the Inres protocol by means of SAMSTAG .

CR Categories and Subject Descriptors: C.2.0 [Computer-Communication Networks]: General; C.2.2 [Computer-Communication Networks]: Network Protocols; D.2.5 [Software Engineering]: Testing and Debugging

[1]Contract number 233/257, funded by Swiss PTT

Contents

List of Figures

1 Introduction

SAMSTAG generates a single test case for a system specification and a test purpose. The system specification is given by an SDL description [CCI92b]. The test purpose is given by an MSC [CCI92a]. The test case is printed in TTCN/MP [ISO91b].

A test case contains the observable behaviors of traces. We call them *observables*. There are three different kind of observables. According to the assigned test verdict we call them *pass*, *fail* and *inconclusive observables*. SAMSTAG calculates these observables.

For that purpose SAMSTAG simulates an SDL specification and an MSC in parallel. During the simulation the pass and inconclusive observables are calculated. The fail observable is added by TTCNs default construct. After the calculation the test case is printed. The underlying method is described in [GHN93a].

2 SaMsTaG

SAMSTAG consists of four components (cf Figure 1):

1 The **SDL simulation tool** consists of

  1.1 an **SDL transformator** that reads in the SDL specifications and generates the C++ files of the SDL simulator, and of an

  1.2 **SDL simulator** which simulates the SDL specification.

2 The **MSC simulation tool** consists of an

  2.1 **MSC transformator** that reads the MSC/PR file and creates a internal data structure, and an

  2.2 **MSC simulator** which interprets the internal data structure.

3 The **test case generator** which guides the whole test case calculation. It is separated into three parts:

  3.1 Calculation of **possible pass observables** (cf. section 4.2)

  3.2 Calculation of **unique pass observables** (cf. section 4.2)

  3.3 Calculation of **inconclusive observables**

4 The **TTCN handler**

  4.1 defines the **FAIL observables** and

  4.2 creates the **TTCN/MP test case**.

The commercial SDT [Tel93a] [Tel93b] serves as an SDL frontend and is taken to create the SDL specification in this paper. But any other SDL editor could be used too. As frontend for the MSC simulation tool an MSC editor of the University of Berne [Tog92] is used and ITEX serves as TTCN backend [Swe92].

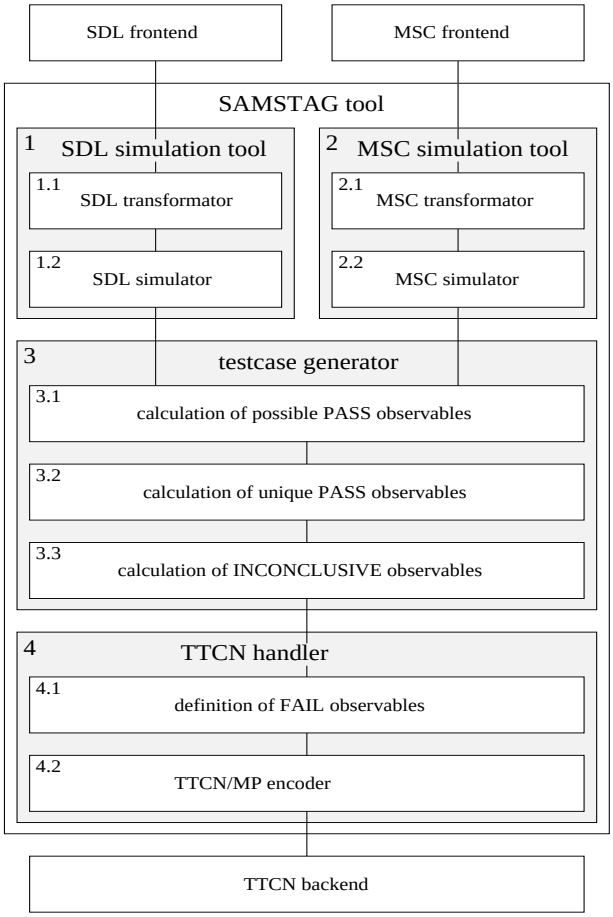

Figure 1: Architecture of SAMSTAG tool

3 Preparing the input

Several steps of work must be done before SAMSTAG is ready to use:

1. The SDL specification must be prepared (cf. Section 3.1).
2. SAMSTAG has to be adapted to the SDL specification (cf. Section 3.2).
3. The test purposes must be prepared (cf. Section 3.3).

3.1 Preparing the SDL specification

The SDL simulation tool requires an SDL specification. The SDL specification describes the test architecture. Thus, it is possible to generate test cases for different test architectures. [ISO91a] provides an overview over different test architectures. As an example the Inres protocol is used. We want to test an implementation of the Initiator process. We choose the distributed test method. There are several changes necessary to adapt the protocol specification [Hog89] to the specification of the test architecture. Only the system under test and the underlying medium appear in the test architecture. The block *Res_Station* is deleted. Additionally, tester processes have to be introduced. The upper

tester is placed at the upper service boundary of the block *Ini_Station* and the lower tester is placed at the upper service boundary of the *Medium*. Thus, the test system is closed and there is no signal exchange with the environment (cf. Fig. 6). Tester processes are processes which can send or receive every signal at every time (cf. Fig. 7 – 10). We assume the medium to be reliable (cf. Fig. 16 – 18). The blocks and processes shown in Fig. 11 – 15 are taken from Inres protocol [Hog91].

3.2 Adapting SaMsTaG to the SDL specification

The SAMSTAG tool has to be created specifically for the SDL specification. The batch file `createSamstag` adapts SAMSTAG to the SDL specification. Before it is started, we create a subdirectory that contains the SDL/PR file. In our example the actual position is `~/inres` (`~` indicates the users home directory) and the SDL/PR file is named `INRES.pr`. The batch `createSamstag` is parameterized by the filename of the SDL specification. The SAMSTAG tool is generated by running the command `createSamstag INRES.pr`. By executing this command, seven steps are performed:

1. Several subdirectories are created.
   - `SDL` contains the SDL/PR file
   - `MSC` contains test purposes and possible pass observables (cf. section 4.2) in MSC/PR format
   - `TTCN` contains test cases in TTCN/MP format
   - `src` contains system specific C++ code of the SDL simulator
   - `bin` contains the executable SAMSTAG
   - `obj` contains compiled object files
   - `template` contains additional code
   - `plot` contains data series of the simulator test, LaTeX tables and postscript figures
   - `log` contains log files which are created while running SAMSTAG in debug mode
2. The program `simgen.exe` generates the SDL system specific C++ code files (cf. [GGG+93]).
3. A Sparc C++ Compiler (at least version 3.0) compiles SAMSTAG and creates the executable code.
4. The performance of the SDL simulator is tested under several heuristics (cf. [GGG+93]). During the test a LaTeX table and several data series are created.
5. The program `gnuplot` creates a postscript figure from the data series.
6. LaTeX compiles a performance report which contains the LaTeX table and the postscript figure.
7. The previewer `ghostview` presents the performance report on the screen.

3.3 Preparing the test purposes

The Inres protocol consist of three phases:

- The connection establishment,
- the data transfer and
- the disconnection.

In the connection establishment and the data transfer phase the signals *connection request* and *data transfer* are transmitted and acknowledged. If the signals are not acknowledged, then the Initiator process can repeat them three times. After the third retransmission the connection is disconnected if no acknowledgement is returned in a certain time. We select eleven test purposes (TP):

TP1: The connection establishment with acknowledgement after the first *connection request*.

TP2: The connection establishment with acknowledgement after the first retransmission of the *connection request*.

TP3: The connection establishment with acknowledgement after the second retransmission of the *connection request*.

TP4: The connection establishment with acknowledgement after the third retransmission of the *connection request*.

TP5: The connection establishment without acknowledgement after the third retransmission of the *connection request*.

TP6: The data transfer with acknowledgement after the first transmission of the *data package*.

TP7: The data transfer with acknowledgement after the first retransmission of the *data package*.

TP8: The data transfer with acknowledgement after the second retransmission of the *data package*.

TP9: The data transfer with acknowledgement after the third retransmission of the *data package*.

TP10: The data transfer without acknowledgement after the third retransmission of the *data package*.

TP11: The disconnection.

MSCs are used to present the test purposes. They have to describe all processes of the SDL specification. The test purposes are presented in Appendix B.

4 Applying SaMsTaG

This section shows how SaMsTaG is started and how a single test case is calculated.

4.1 Starting SaMsTaG

SaMsTaG can be started in batch mode. The tool requires no further input during the generation of a test case. The command line is:

```
bin/samstag dat.sct -SCETG -test- -msc pdat -ttcn dat.tcn -b -t 120 -ppo
            FSFC SRT SRE MSCE NDFS NNC -upo SRT -d- -s 0 -i 10
```

The meaning the command line is:

- `bin/samstag` is the call of the executable code. It is in the subdirectory *bin*.
- `dat.sct` is the name of the MSC/PR file.
- SaMsTaG supports the two formats Z.120 (cf. [CCI92b]) and the MSC/PR format of the MSC editor SCETG [Tog92]. `-SCETG` denotes that the MSC is in the PR form of the SCETG.
- `-test-` turns the SDL simulator test off.
- During the calculation of the pass observable, several possible pass observables (cf. section 4.2) are found. They can be written into a file in MSC/PR form. `-msc pdat` defines their stock of file names. The possible pass observables are written into files with the names `pdat<x>.msc`, where `<x>` denotes a natural number.
- `-ttcn dat.tcn` sets the filename of the TTCN output to `dat.tcn`.
- `-b` turns the batch mode on (no questions during the calculation).
- `-t 120` sets a time limit. The search is aborted if it lasts longer than 120 seconds.
- `-ppo FSFC SRT SRE MSCE NDFS NNC` turns on all implemented heuristics for the calculation of possible pass observables (cf. section 4.2).
- `-upo SRT` turns only the Strong Reasonable Timer heuristic on for checking the uniqueness of the possible pass observables.
- `-d-` turns the debugging features off.
- `-s 0` defines the start level of the search to depth 0.
- `-i 10` sets the incrementation of the upper bound of the depth search.

4.2 Calculation of a single test case

The purpose of this section is to present a complete example how SAMSTAG generates a test case. The used example is the data transfer with acknowledgement after the first transmission of the *data package*. It is supposed that the steps in section 3 are performed. The test purpose is shown in Fig. 2. SAMSTAG calculates a test case in several phases.

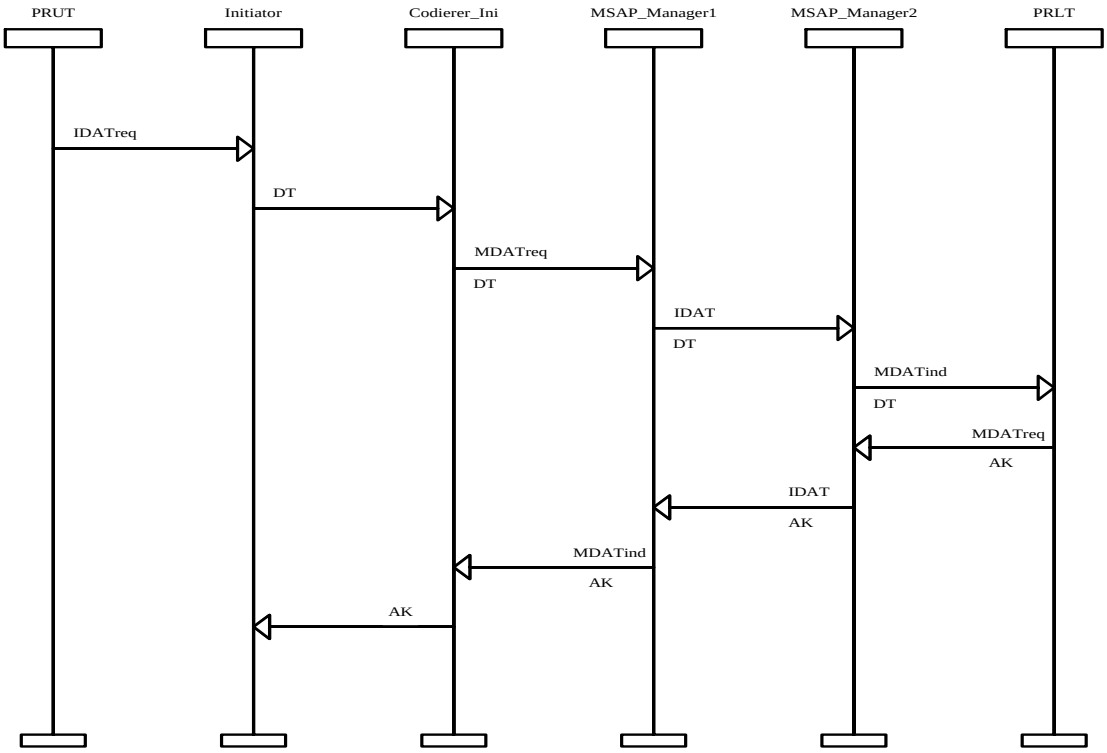

Figure 2: Data transfer

Phase 1: Initialization. In the first phase, the program initializes the parameters with the command line options. The parameters which get no value from the command line are set to default values, if batch mode is on. If batch mode is off, then the user can set the parameters interactively.

Phase 2.1: Search for possible pass observables. This phase looks for traces which start and end in the initial state and perform the signal exchange specified by the test purpose. It is not absolute compelling to attach the pass verdict to the observables of these traces. Therefore, we call these observables *possible pass observables*.

By means of a k-bounded depth search it is looked for such possible pass observables. If no possible pass observables are found, the upper boundary of the k-bounded depth search is incremented. In the example the first possible pass observable is found between depth 60 and 70. The depth is determined by the number of executed SDL statements.

When a possible pass observable is found it is written onto a file in MSC/PR format. The possible pass observable, which is found first, is shown in Fig. 3.

Phase 2.2: Check for uniqueness. This phase checks the uniqueness of a possible pass observable. A possible pass observable is unique if all traces, which perform the possible pass observable, start and end in the initial state and the test purpose is executed. A possible pass observable which shows these properties is called *unique pass observable*. The MSC shown in Fig. 4 disproves the uniqueness of the possible pass observable presented in Fig. 3.

If the possible pass observable is not unique the search for possible pass observables continues (Phase 2.1). The thirteenth possible pass observable is the unique one. It is shown in Figure 5.

During the check for uniqueness the inconclusive observables of the possible pass observable are calculated. An inconclusive observable corresponds with the possible pass observable for a while, but ends with an allowed but not expected input.

Phase 3: Initialization of the test case. The shortest unique pass observable is the pass observable of the test case. Its inconclusive behavior is the inconclusive behavior of the test case.

Phase 4: Extension of the test case. SAMSTAG tries to extend the test case by further unique pass observables of the same length and its inconclusive observables.

Phase 5: Print TTCN test case. In a last step the calculated test case is printed on a file in TTCN/MP format. A fail observable is a sequence of observable events, which can not be derived by the simulation of the SDL system. In this step the fail observables are added by the default behavior construct of TTCN. The generated dynamic behavior table is shown on page 19 of Appendix C.

5 Preparing the Output

All other test cases have to be generated in the same way as the test case for the data transfer. The test purposes are listed in Appendix B. After the test cases have to be merged together to one test suite. Because the test cases are generated full automatically the names of the test case and of the dynamic behavior are dummy names and have to be renamed. The test cases can be merged by the merge tool of ITEX into one test suite such that each test case, constraint, etc. appears once. The result of the merging is the test suite of Appendix C.

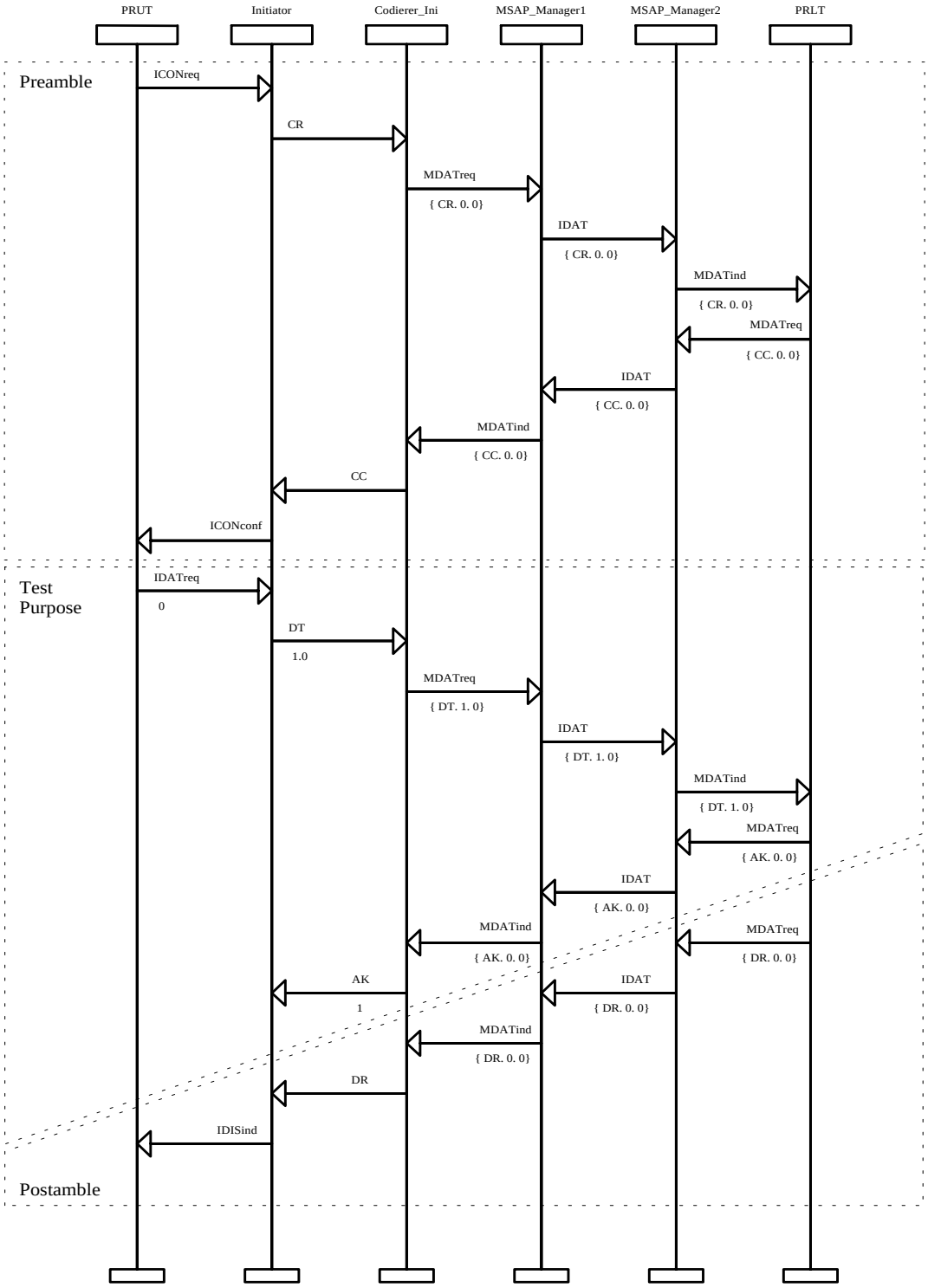

Figure 3: First possible pass observable that is found during the search

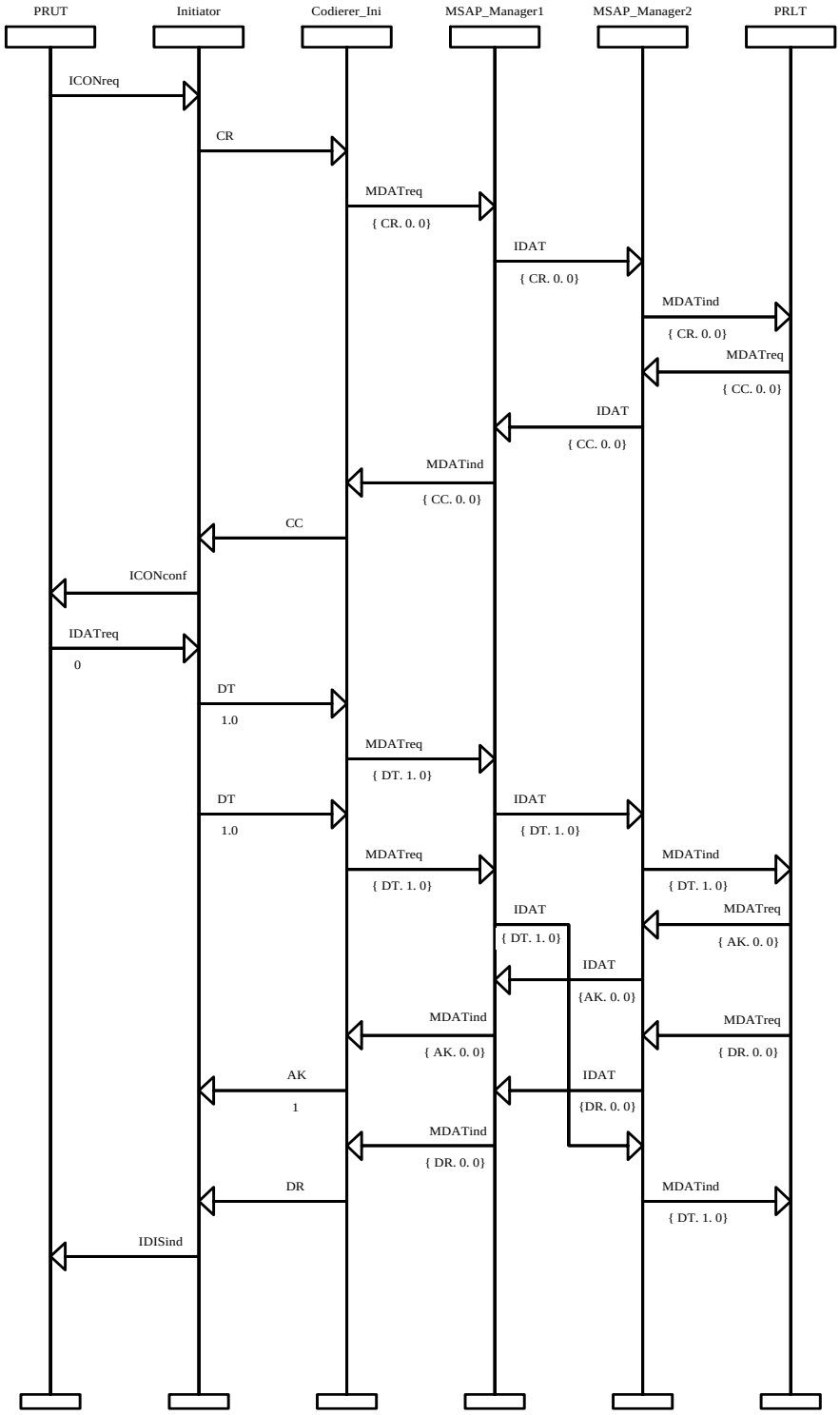

Figure 4: Contradiction against uniqueness of PPO1

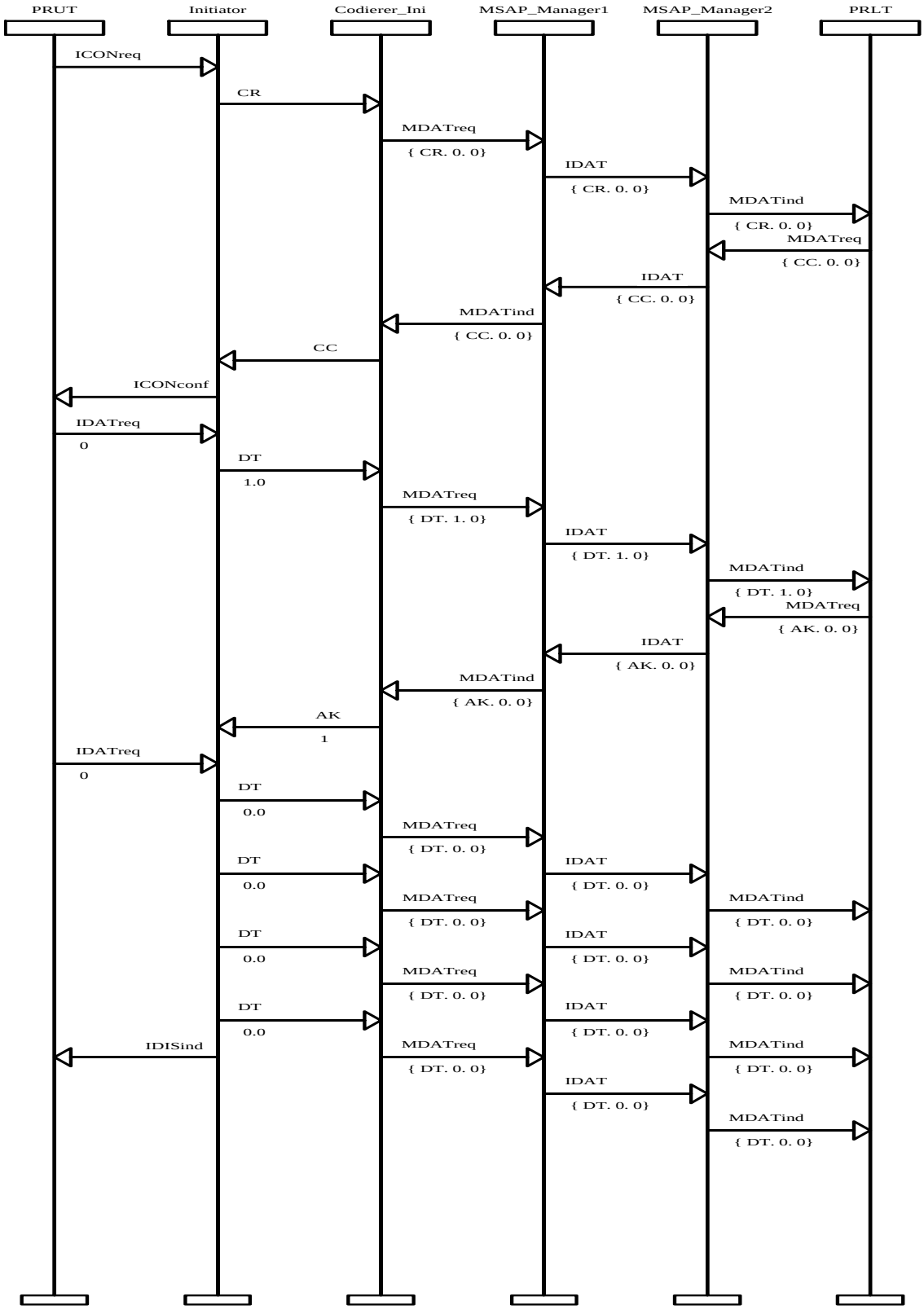

Figure 5: Unique Pass observable

References

[CCI92a] CCITT SG X. Message Sequence Chart (MSC). Recommendation Z.120, CCITT, 1992. Geneva.

[CCI92b] CCITT SG X. Specification and description language (SDL). Recommendation Z.100, CCITT, 1992. Geneva.

[GGG+93] Grabowski, Günter, Gurtner, Hogrefe, Nahm, Neuenschwander, Spichiger, and Toggweiler. Conformance Testing - Ein Werkzeug zur Generierung von Testfällen. Schlussbericht des F & E Projektes Vetragsnummer 233, finanziert durch die Schweizer PTT, November 1993.

[GHN93a] Jens Grabowski, Dieter Hogrefe, and Robert Nahm. A method for the generation of test cases based on SDL and MSCs. Technical Report IAM 93-010, University of Berne, Switzerland, April 1993.

[GHN93b] Jens Grabowski, Dieter Hogrefe, and Robert Nahm. Test case generation with test purpose specification by MSCs. In Ove Faergemand and Sarma Armadeo, editors, *6th SDL Forum*, SDL '93 : Using Objects, pages 253–266. North-Holland, October 1993.

[Hog89] Dieter Hogrefe. *Estelle, LOTOS und SDL - Standard Specifikationssprachen für verteilte Systeme*. Springer Verlag, 1989.

[Hog91] Dieter Hogrefe. OSI formal specification case study: The INRES protocol and service. Technical Report IAM-91-012, University of Berne, 1991.

[ISO91a] ISO/IEC JTC 1/SC 21 N. Information technology - Open System Interconnection - conformance testing methodology and framework - Part1-5. International Standard 9646, ISO/IEC, 1991.

[ISO91b] ISO/IEC JTC 1/SC21. Information technology - Open Systems Interconnection - conformance testing methodology and framework - Part 3: The Tree and Tabular Combined Notation. International Standard 9646-3, ISO, 1991.

[Swe92] Swedish Telecom, S-123 86 Farsta. *ITEX-DE version 2.0*, 1992.

[Tel93a] Telelogic Malmö AB, S-203 12 Malmö. *SDT 2.3, Reference Manual*, 1993.

[Tel93b] Telelogic Malmö AB. SDT, the SDL design tool. In Ove Faergemand and Amardeo Sarma, editors, *6th SDL Forum*, SDL '93: Using Objects, pages 513–514. North-Holland, October 1993.

[Tog92] Daniel Toggweiler. TTCN-Testfallgenerierung für mit Sequence Charts spezifizierte verteilte Systeme. Diploma thesis, University of Berne, March 1992.

A SDL specification

In this appendix the whole SDL specification of the test architecture is shown. The system diagram is presented in Fig. 6, the blocks in Fig. 7, 9, 11 and 16 and the processes in Fig. 8, 10, 12 - 15, 17 and 18.

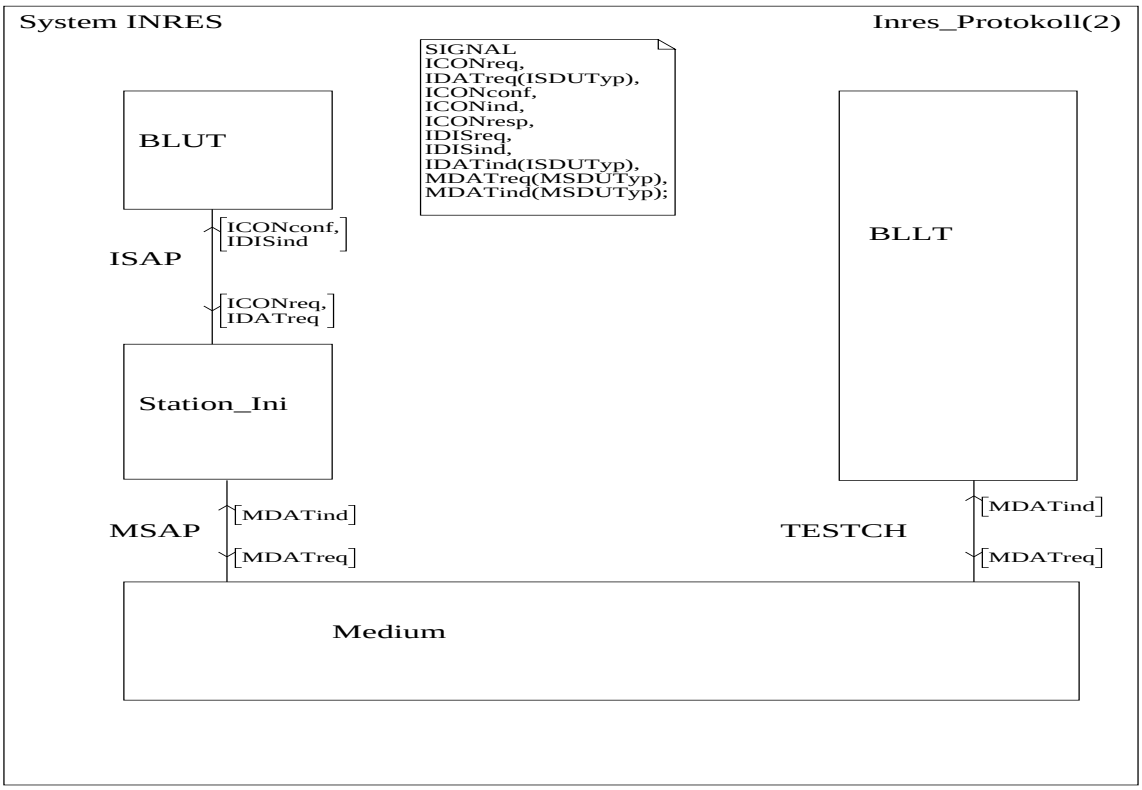

Figure 6: Test architecture

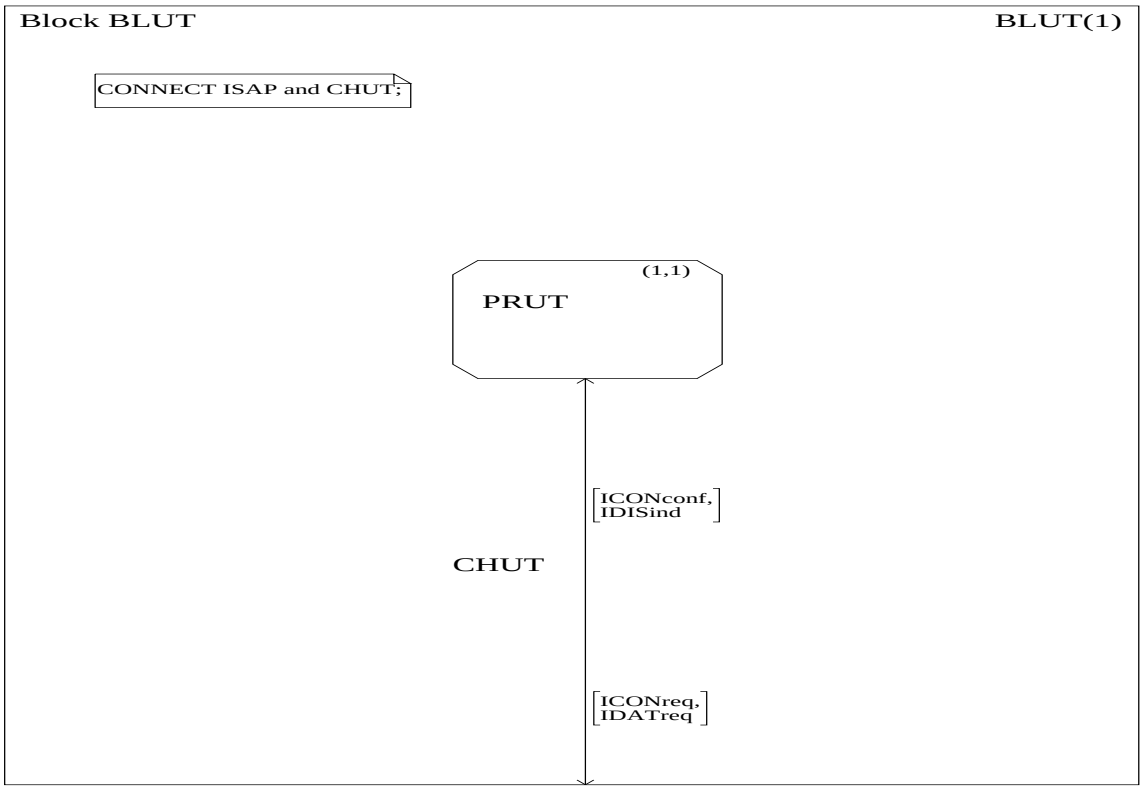

Figure 7: Block Upper Tester

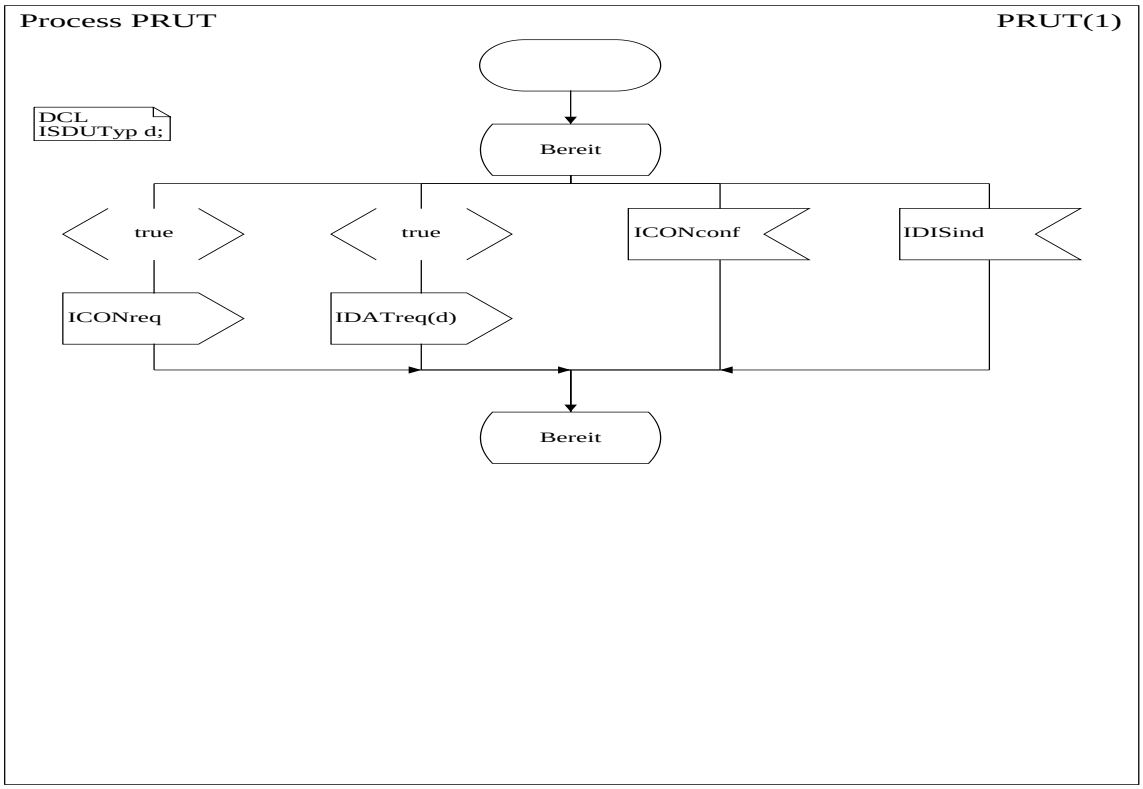

Figure 8: Process Upper Tester

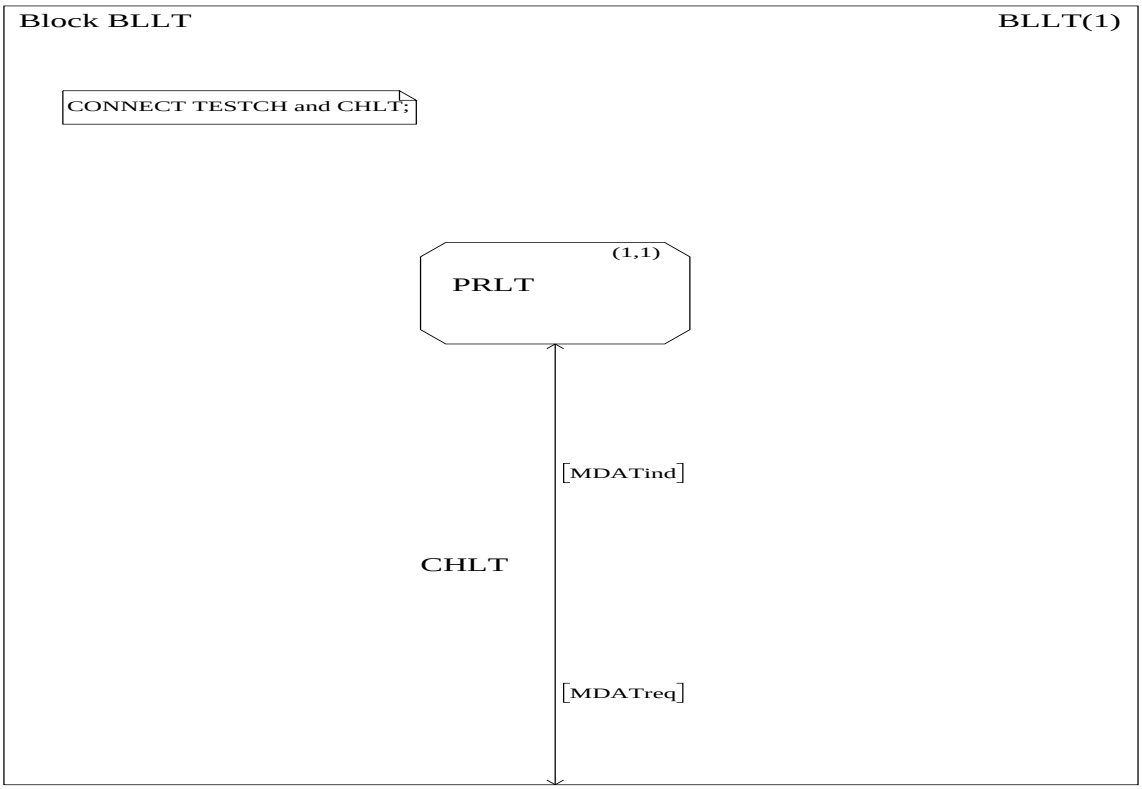

Figure 9: Block Lower Tester

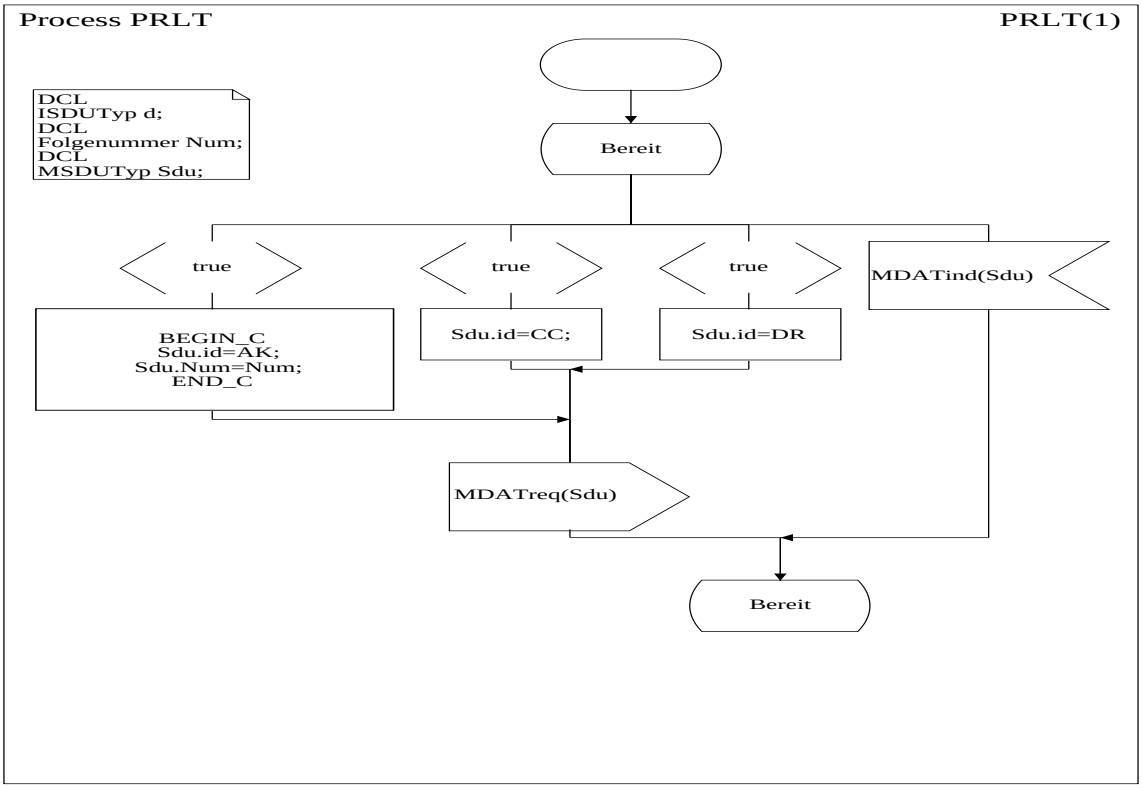

Figure 10: Process Lower Tester

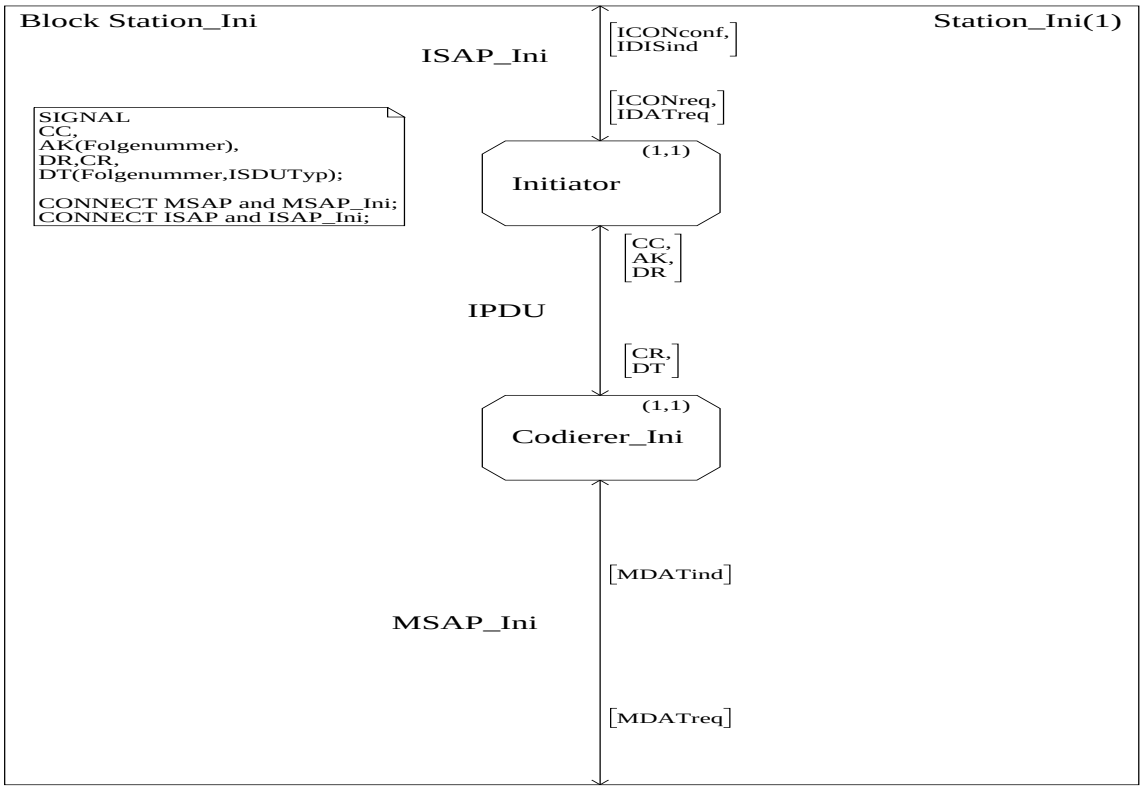

Figure 11: Block Station_Ini

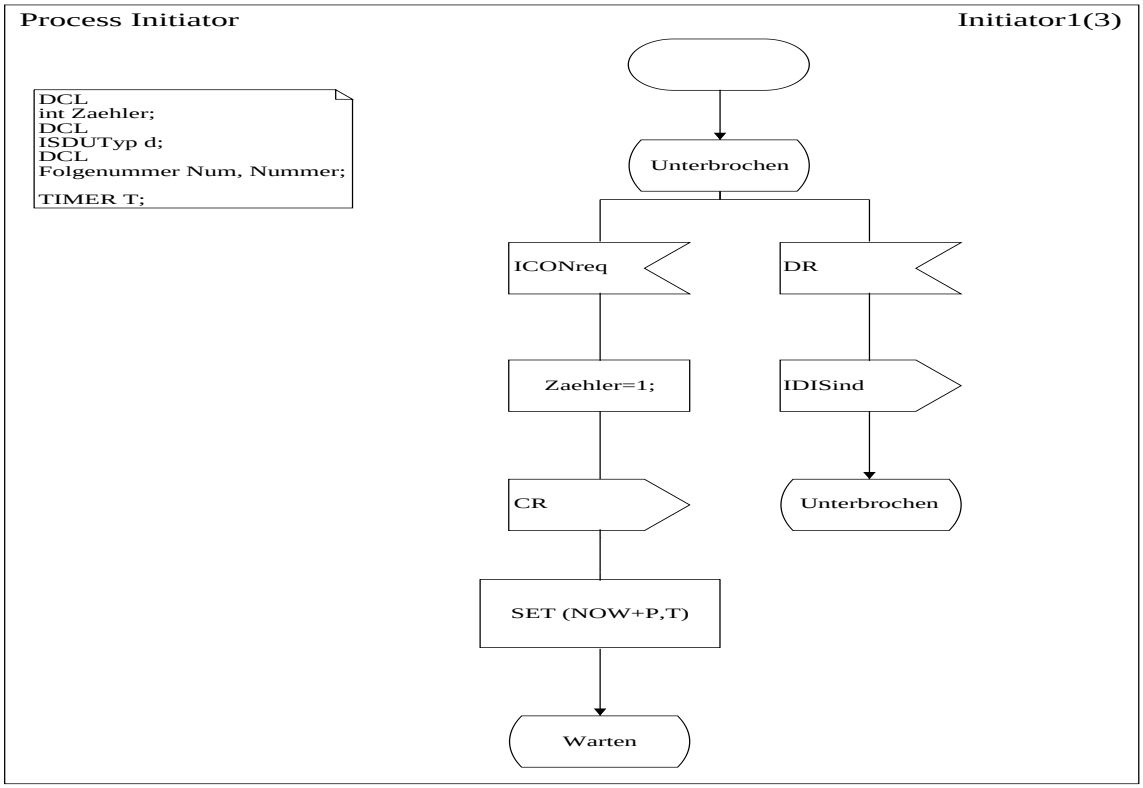

Figure 12: Process Initiator 1(3)

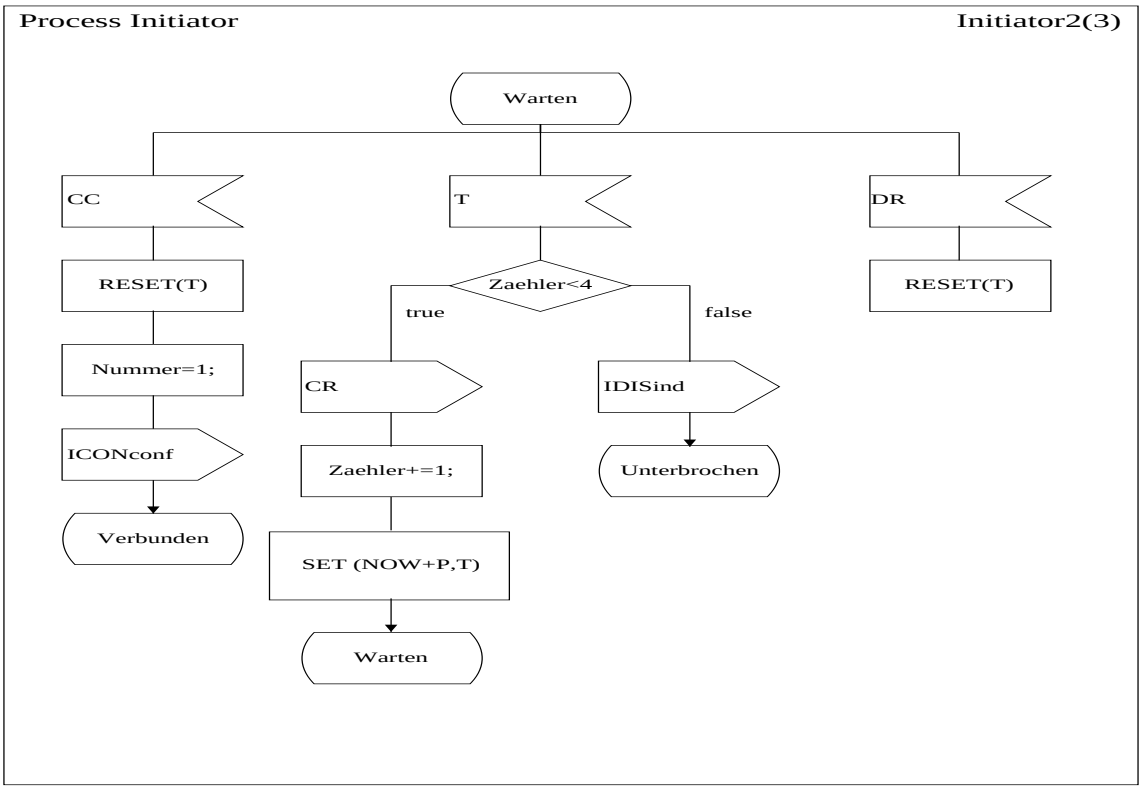

Figure 13: Process Initiator 2(3)

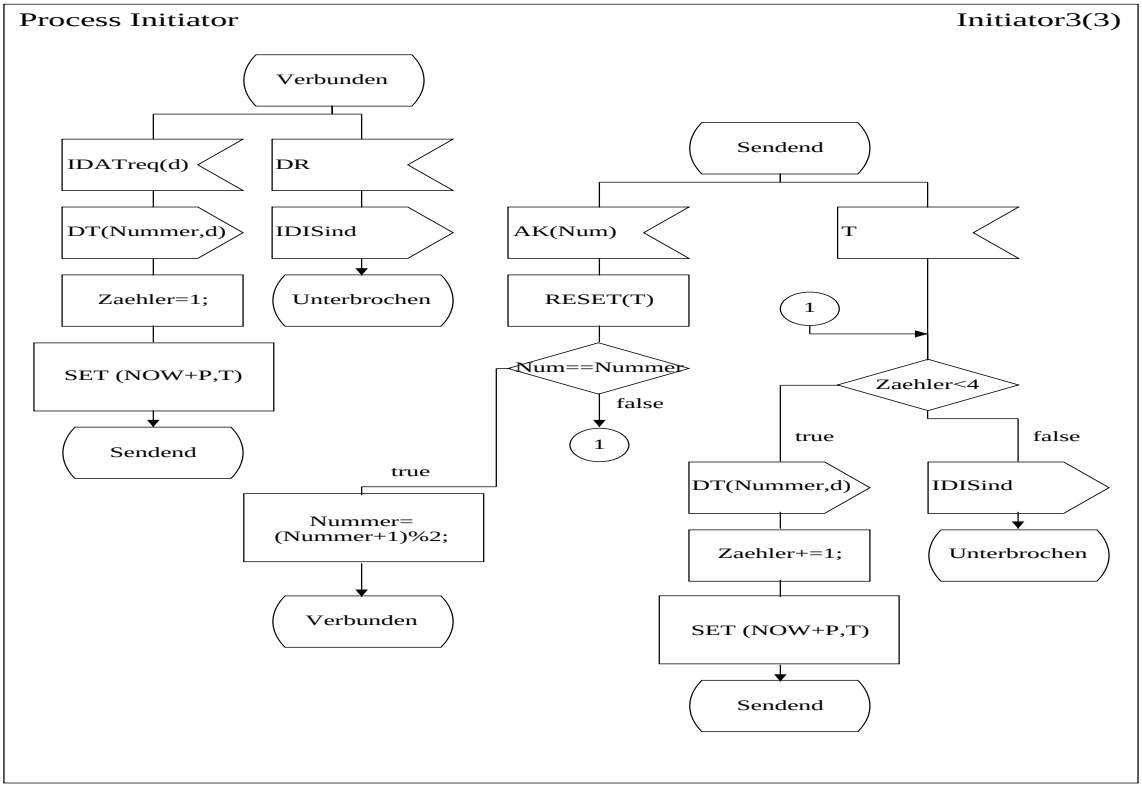

Figure 14: Process Initiator 3(3)

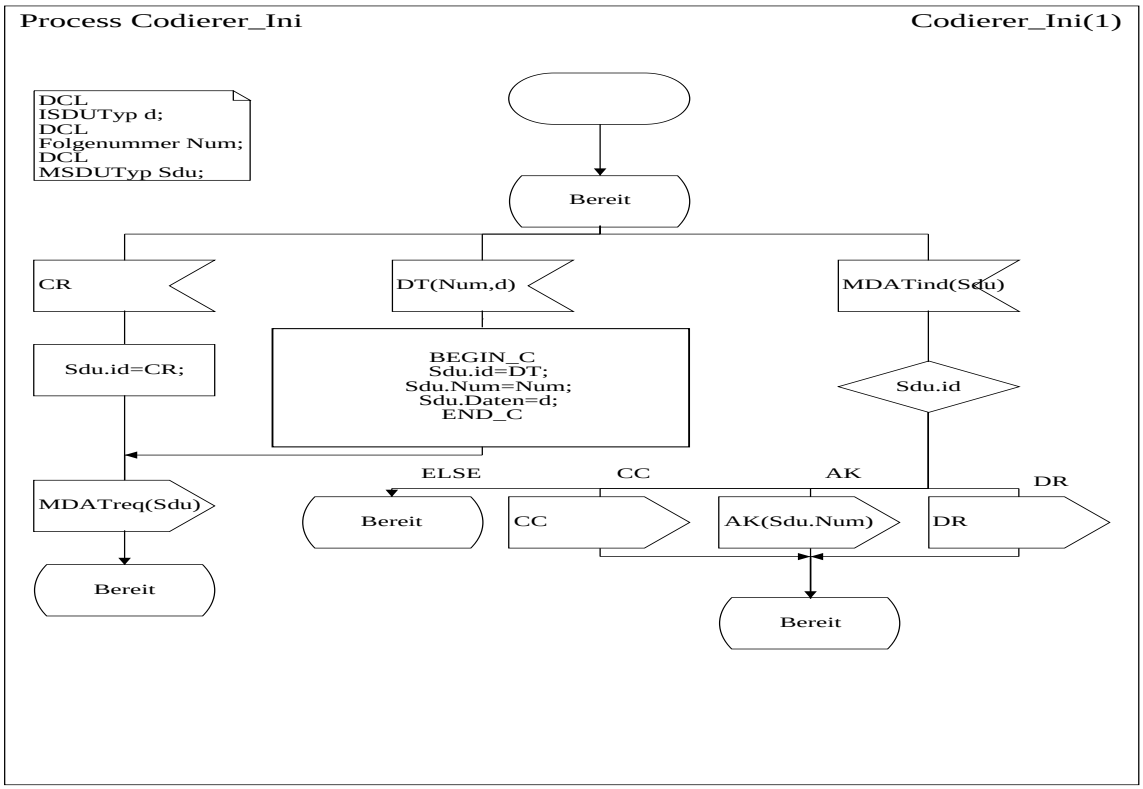

Figure 15: Process Coder Initiator

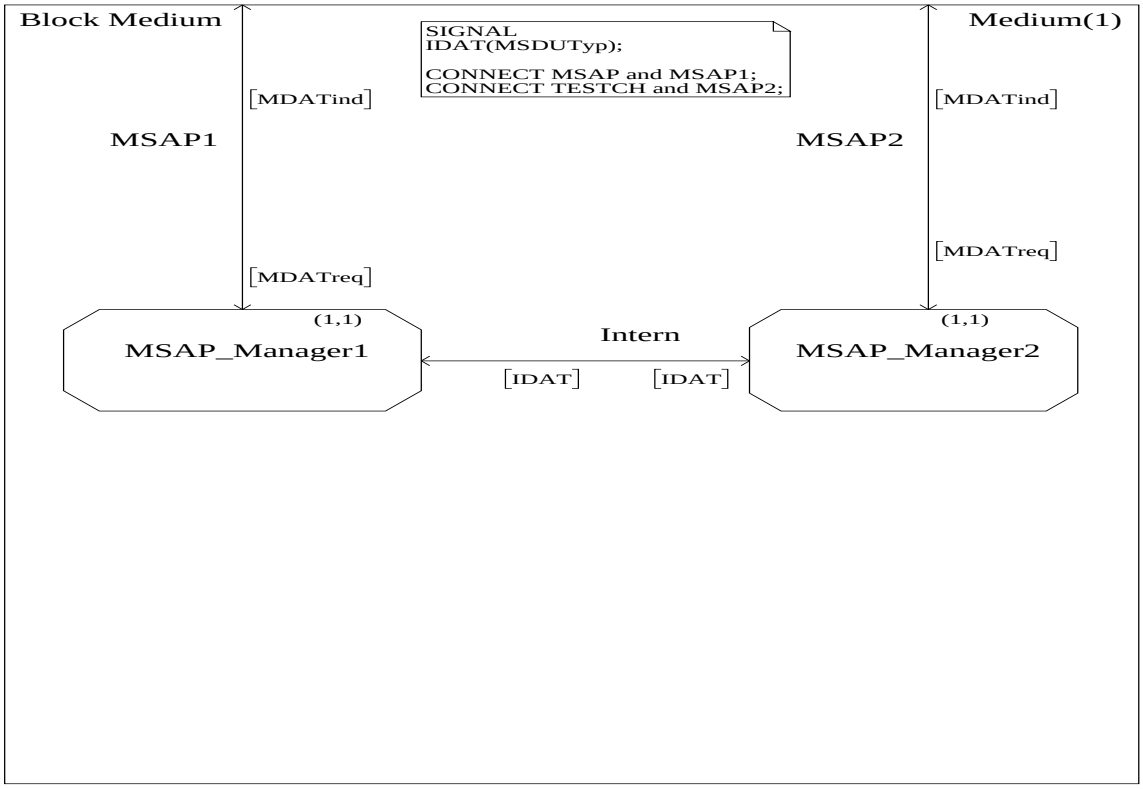

Figure 16: Block Medium

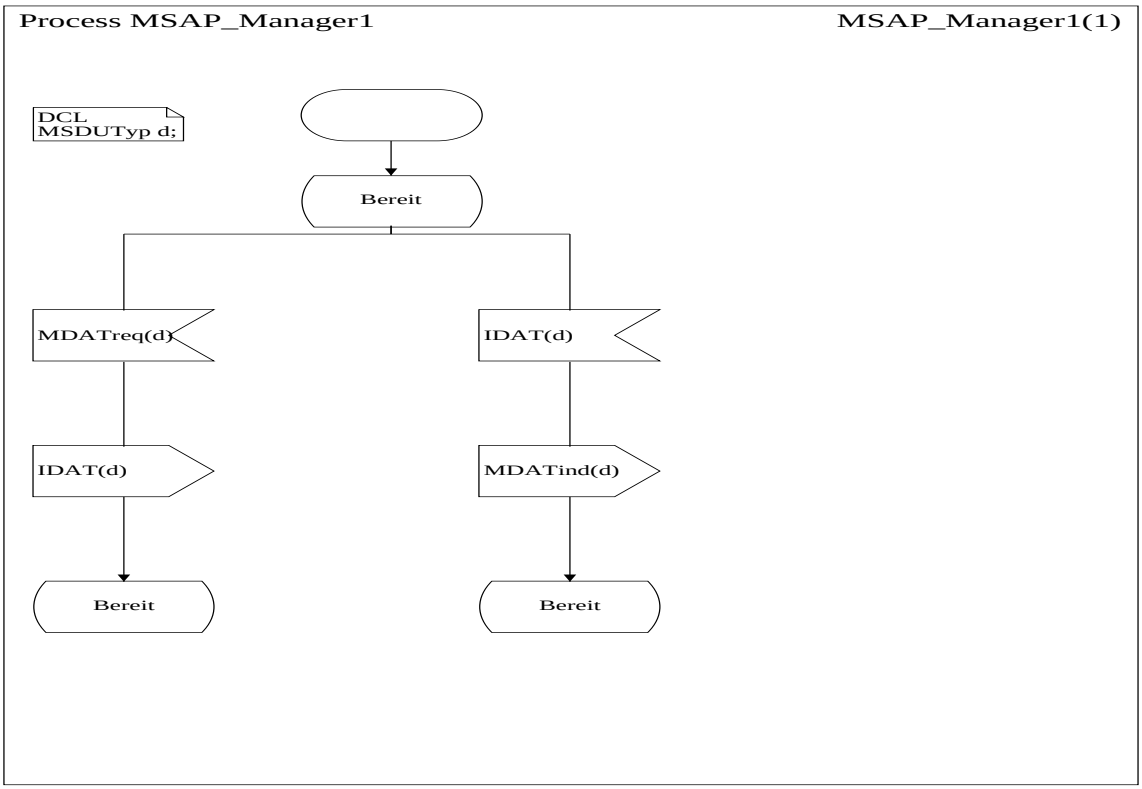

Figure 17: Process MSAP_Manager1

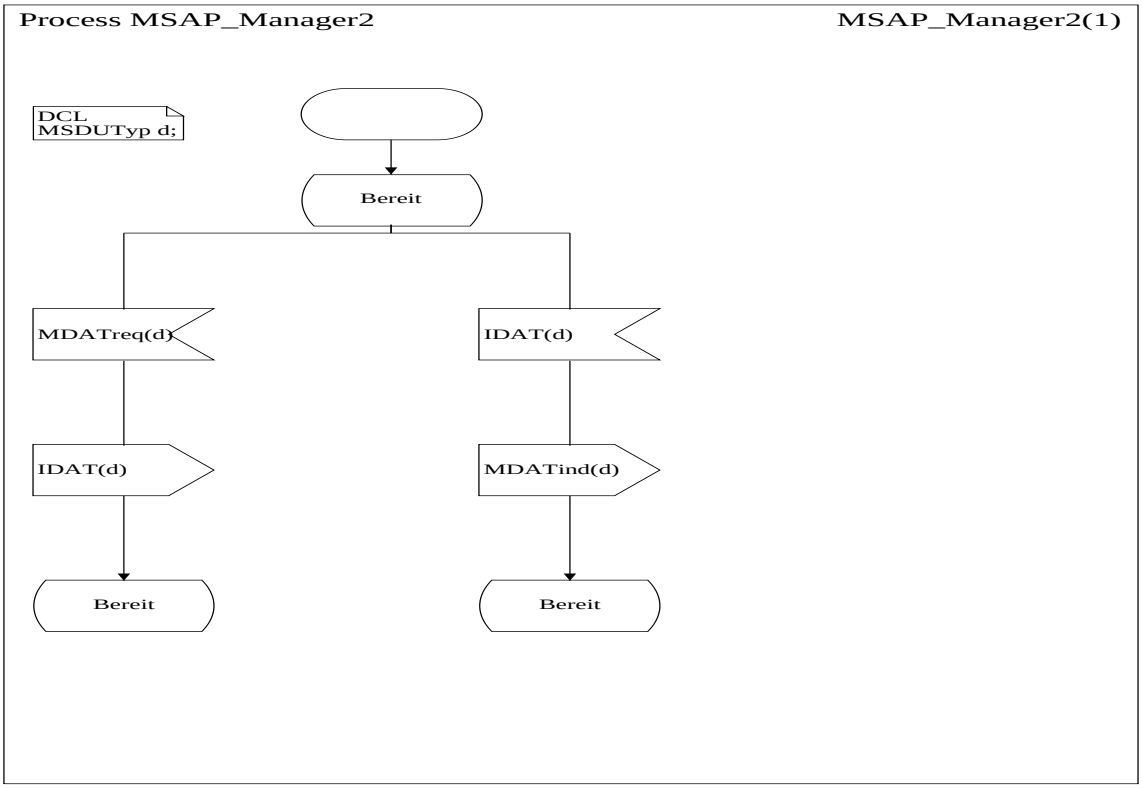

Figure 18: Process MSAP_Manager2

B Test Purposes

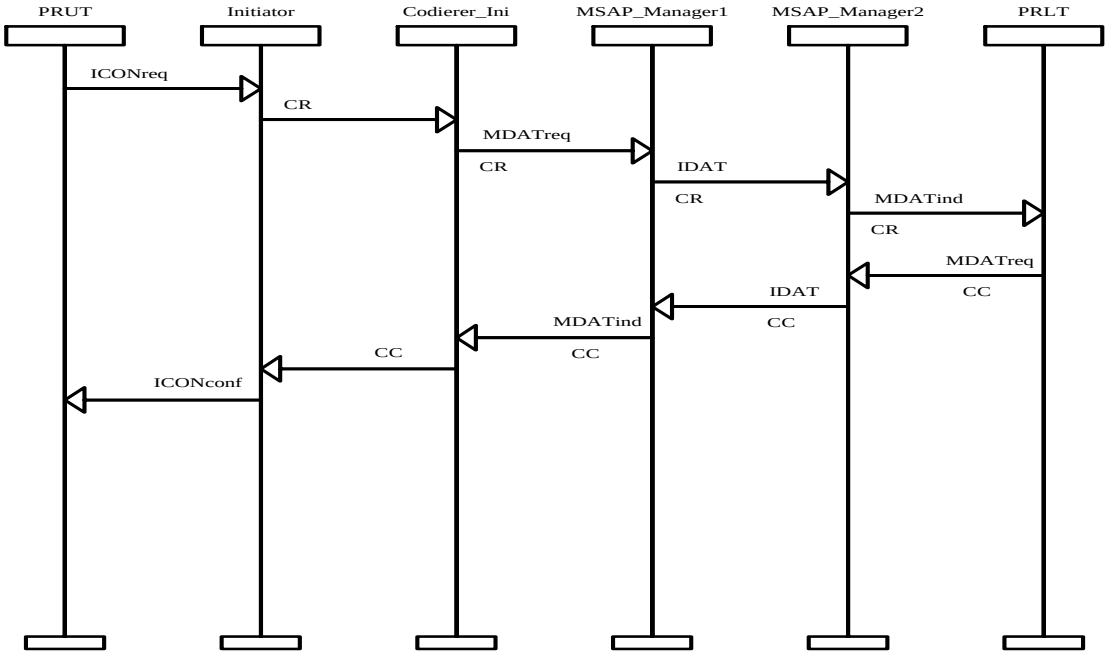

Figure 19: Connection establishment (CE) (TP1)

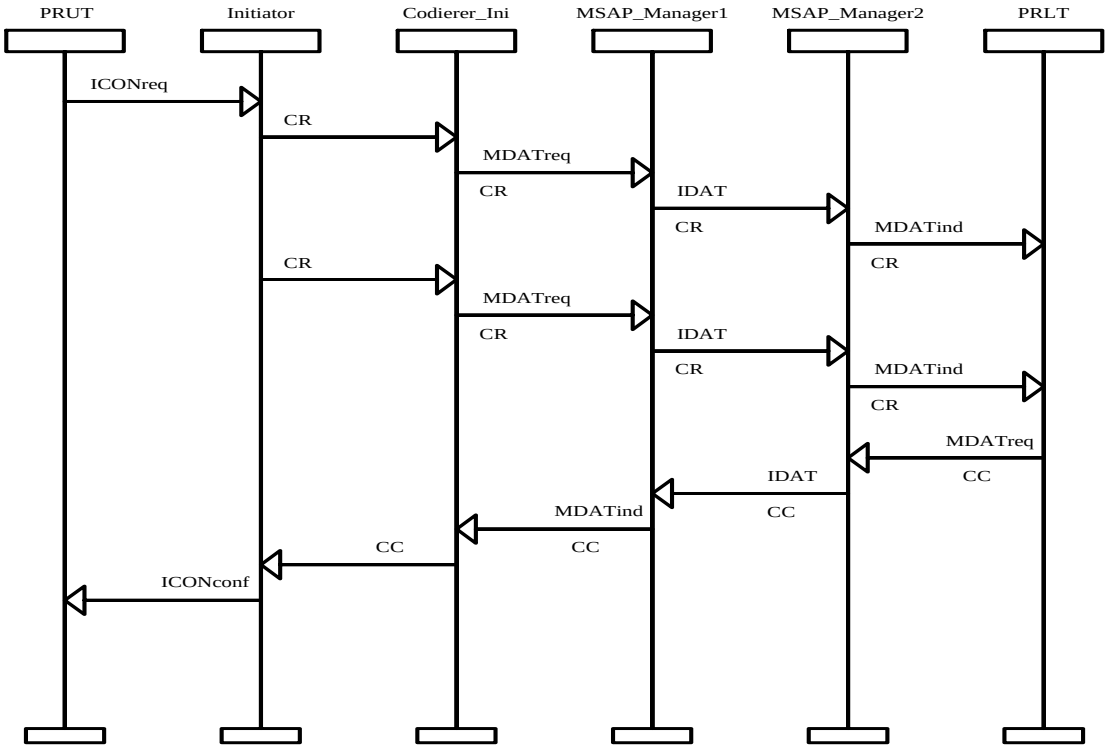

Figure 20: CE with single retransmission of *connection request* (TP2)

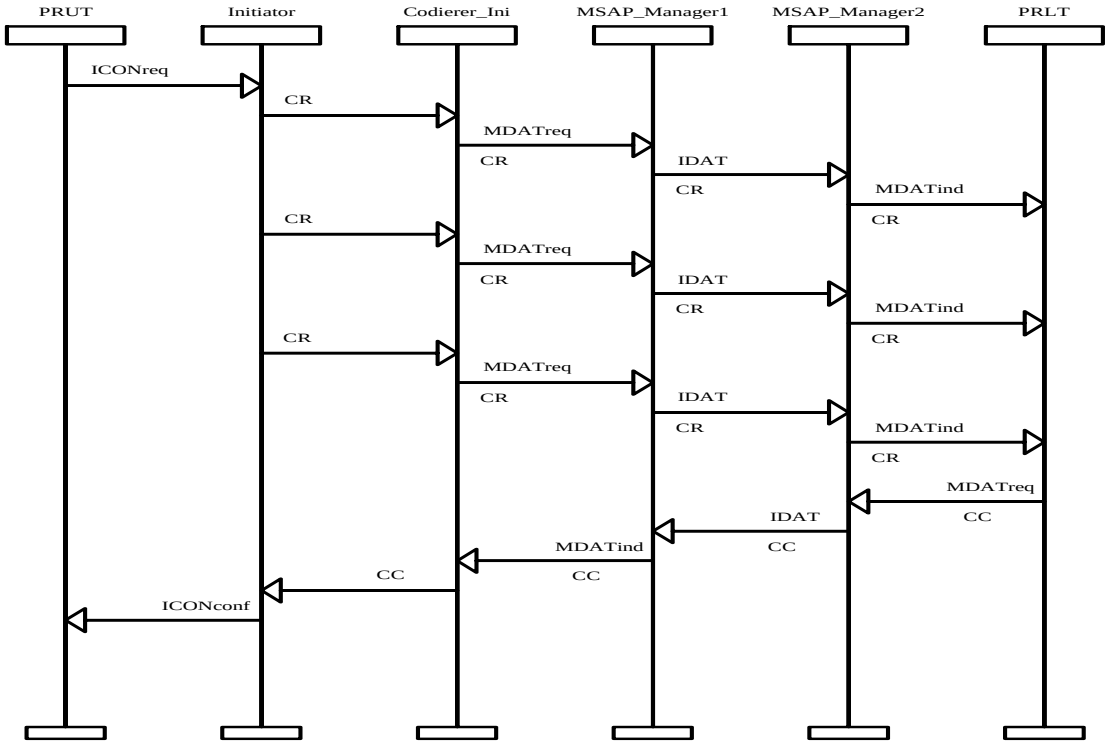

Figure 21: CE with twice retransmission of *connection request* (TP3)

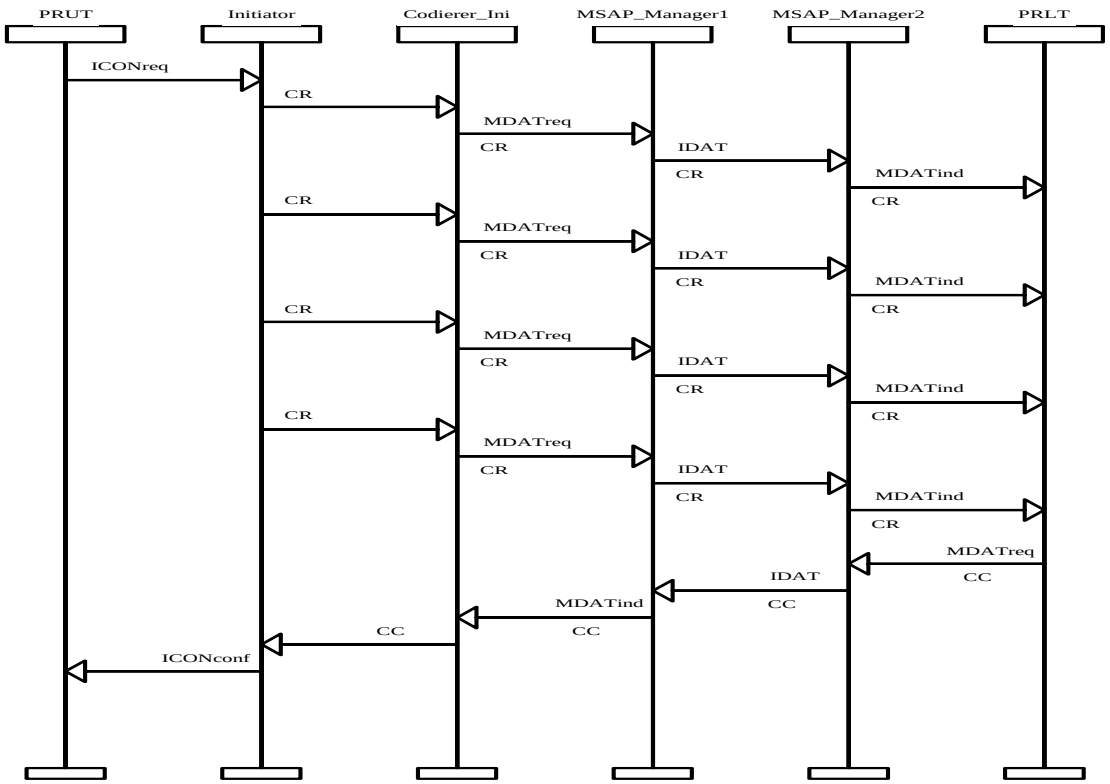

Figure 22: CE with thrice retransmission of *connection request* (TP4)

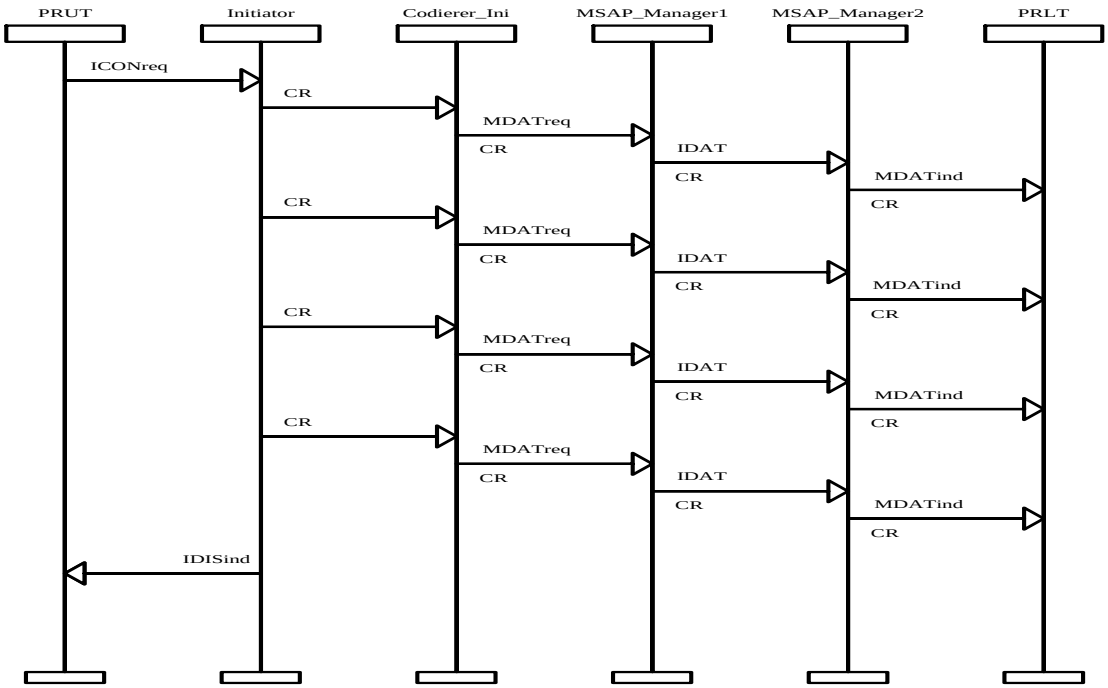

Figure 23: Not acknowledged connection establishment (TP5)

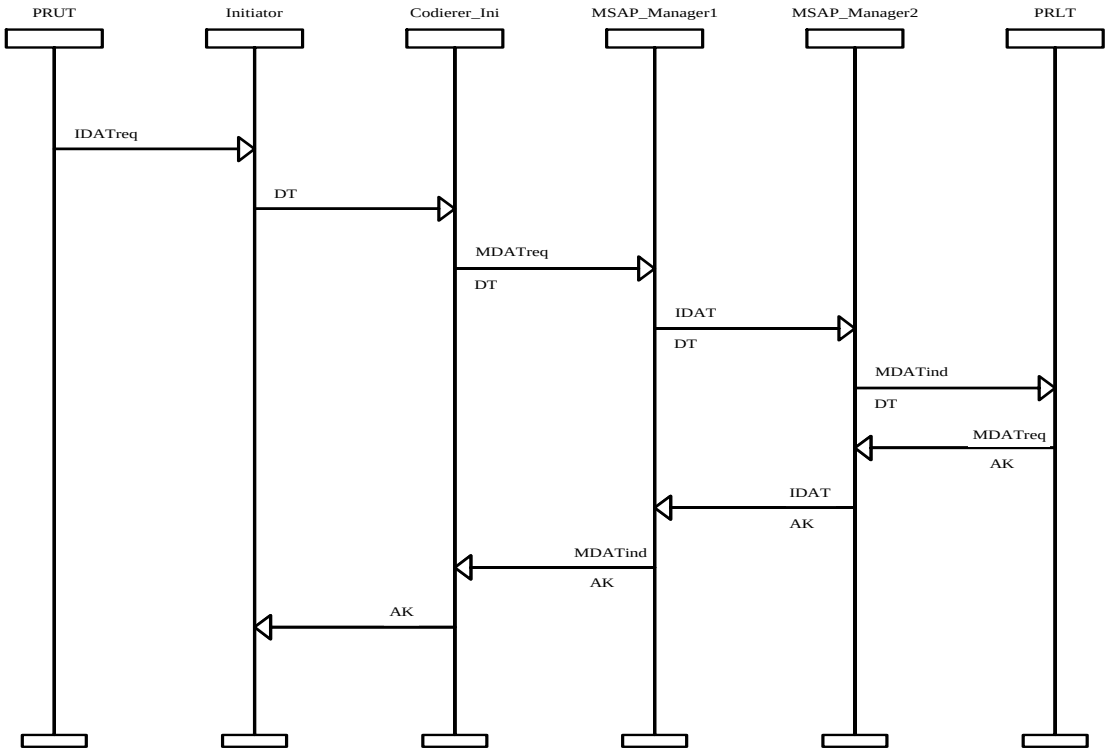

Figure 24: Data transfer (DT)(TP6)

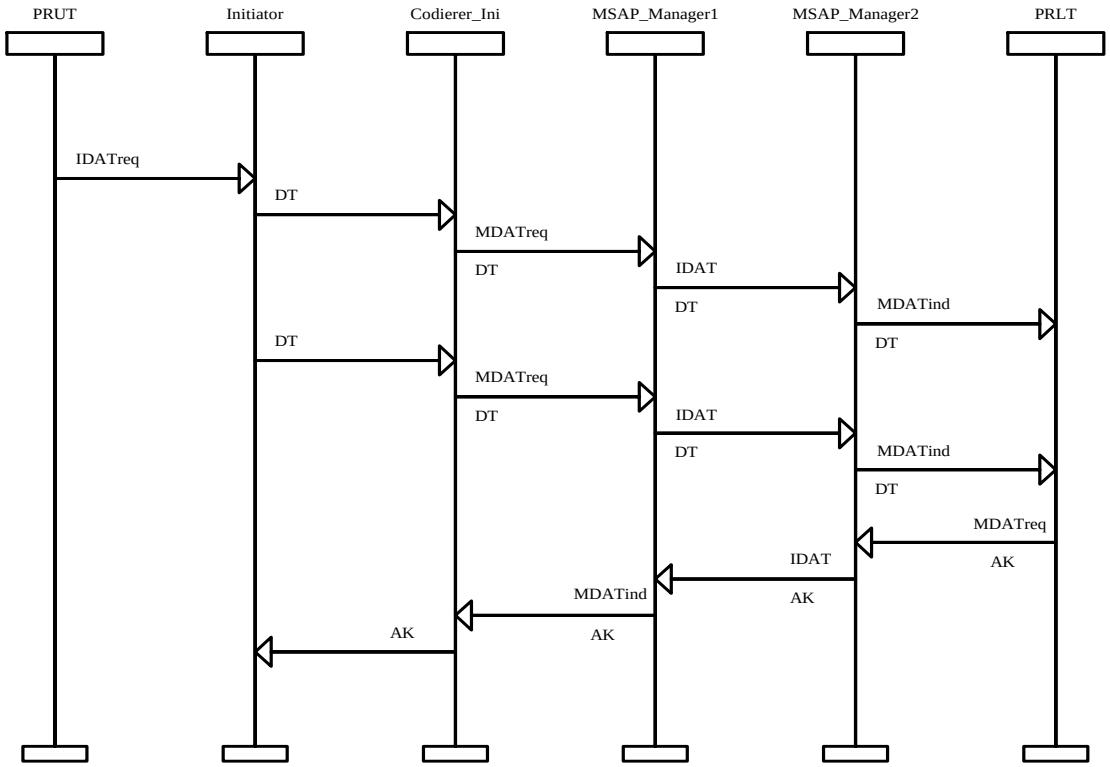

Figure 25: DT with single retransmission of data package (TP7)

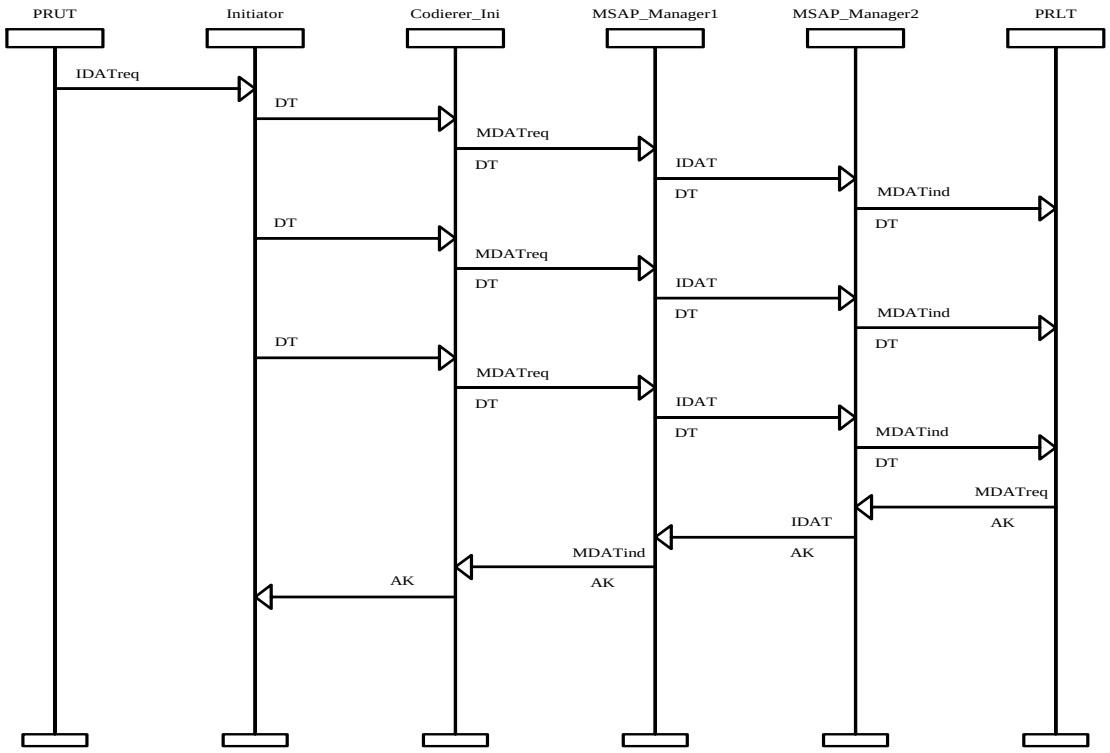

Figure 26: DT with twice retransmission of data package (TP8)

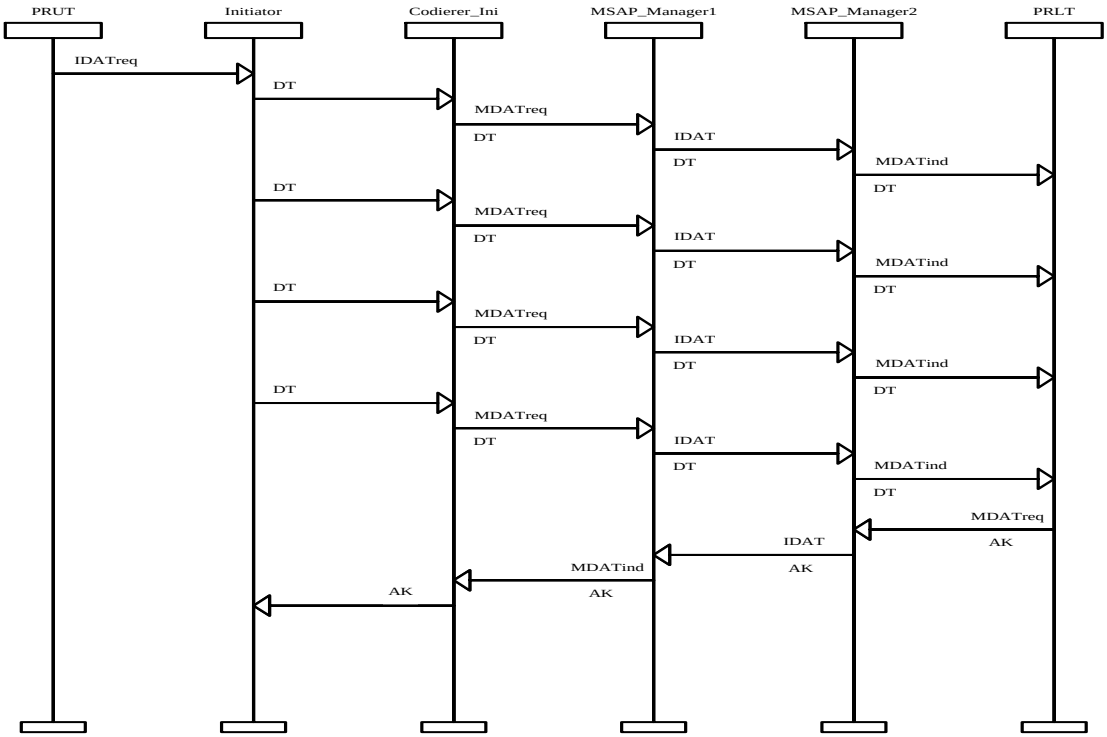

Figure 27: DT with thrice retransmission of data package (TP9)

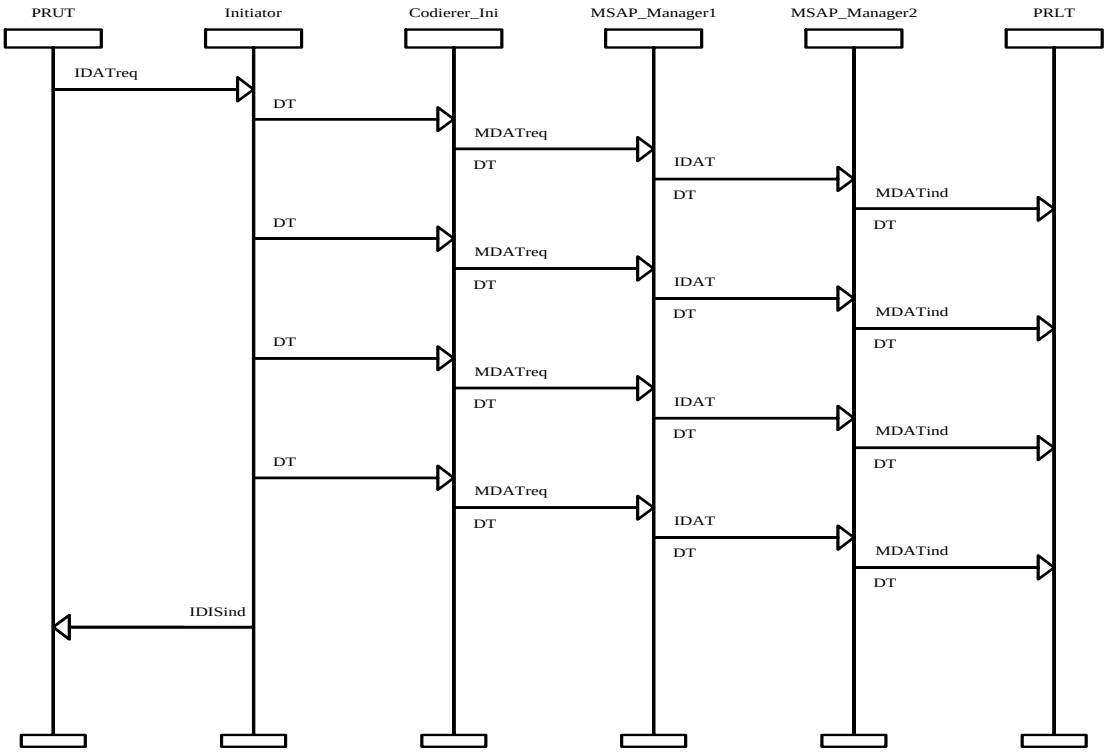

Figure 28: Not acknowledged data transfer (TP10)

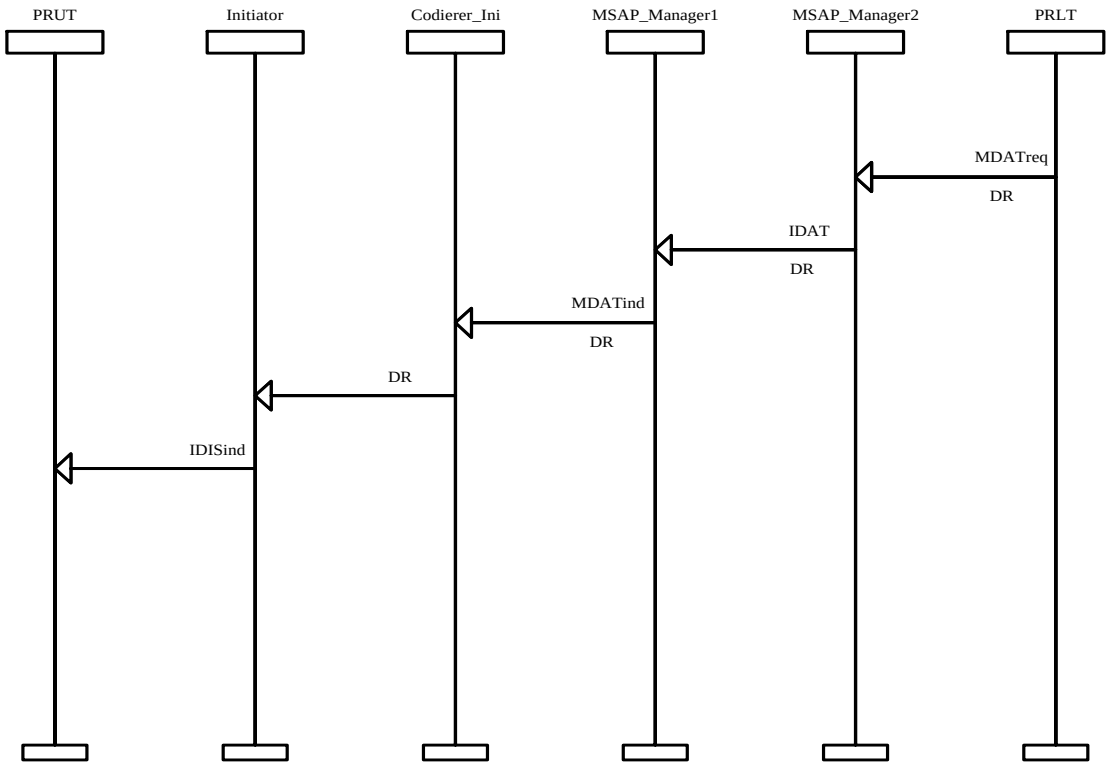

Figure 29: Disconnection (TP11)

C TTCN Test suite

The following pages are direct output from TELELOGICs ITEX tool.

I

Test Suite Overview

Test Suite Structure			
Suite Name : Inres			
Standards Ref :			
PICS Ref :			
PIXIT Ref :			
Test Method(s) :			
Comments :			
Test Group Reference	**Selection Ref**	**Test Group Objective**	**Page Nr**
Detailed Comments :			

Test Case Index

Detailed Comments :

Default Index			
Default Group Reference	**Default Id**	**Description**	**Page Nr**
	stddefault		24
Detailed Comments :			

II

Declarations Part

PCO Declarations			
PCO Name	**PCO Type**	**Role**	**Comments**
PRUT	P_SAP	UT	
PRLT	N_SAP	LT	
Detailed Comments :			

TTCN ASP Type Definition		
ASP Name : ICONreq **PCO Type :** P_SAP **Comments :**		
Parameter Name	**Parameter Type**	**Comments**
Detailed Comments :		

TTCN ASP Type Definition		
ASP Name : IDATreq **PCO Type :** P_SAP **Comments :**		
Parameter Name	**Parameter Type**	**Comments**
IDATreq_Par_1	ISDUTyp	
Detailed Comments :		

TTCN ASP Type Definition		
ASP Name : ICONconf **PCO Type :** P_SAP **Comments :**		
Parameter Name	**Parameter Type**	**Comments**
Detailed Comments :		

TTCN ASP Type Definition		
ASP Name : IDISind **PCO Type :** P_SAP **Comments :**		
Parameter Name	**Parameter Type**	**Comments**
Detailed Comments :		

TTCN ASP Type Definition		
ASP Name : MDATreq **PCO Type :** N_SAP **Comments :**		
Parameter Name	**Parameter Type**	**Comments**
MDATreq_Par_1_CC0_D	MSDUTyp	
Detailed Comments :		

TTCN ASP Type Definition		
ASP Name : MDATind **PCO Type :** N_SAP **Comments :**		
Parameter Name	**Parameter Type**	**Comments**
MDATind_Par_1	MSDUTyp	
Detailed Comments :		

TTCN PDU Type Definition		
PDU Name : MSDUTyp **PCO Type :** N_SAP **Comments :**		
Field Name	**Field Type**	**Comments**
field_name_1	field_type_1	
field_name_2	field_type_2	
field_name_3	field_type_3	
Detailed Comments :		

III

Constraints Part

TTCN ASP Constraint Declaration

Constraint Name : MDATreq_CC0_D
ASP Type : MDATreq
Derivation Path :
Comments :

Parameter Name	Parameter Value	Comments
MDATreq_Par_1_CC0_D	MDATreq_Par_1_CC0_D	

Detailed Comments :

TTCN ASP Constraint Declaration

Constraint Name : MDATind_DT1_J
ASP Type : MDATind
Derivation Path :
Comments :

Parameter Name	Parameter Value	Comments
MDATind_Par_1_DT1_J	MDATind_Par_1_DT1_J	

Detailed Comments :

TTCN ASP Constraint Declaration

Constraint Name : MDATind_CR0_F
ASP Type : MDATind
Derivation Path :
Comments :

Parameter Name	Parameter Value	Comments
MDATind_Par_1_CR0_F	MDATind_Par_1_CR0_F	

Detailed Comments :

TTCN ASP Constraint Declaration

Constraint Name : IDATreq_0_O
ASP Type : IDATreq
Derivation Path :
Comments :

Parameter Name	Parameter Value	Comments
ISDUTyp	0	

Detailed Comments :

TTCN ASP Constraint Declaration		
Constraint Name : MDATreq_DR0_G		
ASP Type : MDATreq		
Derivation Path :		
Comments :		
Parameter Name	**Parameter Value**	**Comments**
MDATreq_Par_1_DR0_G	MDATreq_Par_1_DR0_G	
Detailed Comments :		

TTCN ASP Constraint Declaration		
Constraint Name : MDATreq_AK0_J		
ASP Type : MDATreq		
Derivation Path :		
Comments :		
Parameter Name	**Parameter Value**	**Comments**
MDATreq_Par_1_CC0_D_AK0_J	MDATreq_Par_1_CC0_D_AK0_J	
Detailed Comments :		

TTCN ASP Constraint Declaration		
Constraint Name : MDATind_DT0_I		
ASP Type : MDATind		
Derivation Path :		
Comments :		
Parameter Name	**Parameter Value**	**Comments**
MDATind_Par_1_DT0_I	MDATind_Par_1_DT0_I	
Detailed Comments :		

TTCN PDU Constraint Declaration

Constraint Name : MDATreq_Par_1_CC0_D
PDU Type : MSDUTyp
Derivation Path :
Comments :

Field Name	Field Value	Comments
field_name_1	CC	
field_name_2	0	
field_name_3	0	

Detailed Comments :

TTCN PDU Constraint Declaration

Constraint Name : MDATind_Par_1_DT1_J
PDU Type : MSDUTyp
Derivation Path :
Comments :

Field Name	Field Value	Comments
field_name_1	DT	
field_name_2	1	
field_name_3	0	

Detailed Comments :

TTCN PDU Constraint Declaration

Constraint Name : MDATind_Par_1_CR0_F
PDU Type : MSDUTyp
Derivation Path :
Comments :

Field Name	Field Value	Comments
field_name_1	CR	
field_name_2	0	
field_name_3	0	

Detailed Comments :

TTCN PDU Constraint Declaration

Constraint Name : MDATreq_Par_1_DR0_G_CC0_D
PDU Type : MSDUTyp
Derivation Path :
Comments :

Field Name	Field Value	Comments
field_name_1	CC	
field_name_2	0	
field_name_3	0	

Detailed Comments :

TTCN PDU Constraint Declaration

Constraint Name : MDATreq_Par_1_DR0_G
PDU Type : MSDUTyp
Derivation Path :
Comments :

Field Name	Field Value	Comments
field_name_1	DR	
field_name_2	0	
field_name_3	0	

Detailed Comments :

TTCN PDU Constraint Declaration

Constraint Name : MDATreq_Par_1_CC0_D_AK0_J
PDU Type : MSDUTyp
Derivation Path :
Comments :

Field Name	Field Value	Comments
field_name_1	AK	
field_name_2	0	
field_name_3	0	

Detailed Comments :

TTCN PDU Constraint Declaration

Constraint Name : MDATind_Par_1_DT0_I
PDU Type : MSDUTyp
Derivation Path :
Comments :

Field Name	Field Value	Comments
field_name_1	DT	
field_name_2	0	
field_name_3	0	

Detailed Comments :

IV

Dynamic Part

Test Case Dynamic Behaviour

Test Case Name : Estab
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT!MDATreq	MDATreq_CC0_D		
4		PRUT?ICONconf			
5		PRUT!IDATreq	IDATreq_0_O		
6		PRLT?MDATind	MDATind_DT1_J		
7		PRLT?MDATind	MDATind_DT1_J		
8		PRLT?MDATind	MDATind_DT1_J		
9		PRLT?MDATind	MDATind_DT1_J		
10		PRUT?IDISind		PASS	
11		PRLT?MDATind	MDATind_CR0_F	INCONC	
12		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Estab01
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT?MDATind	MDATind_CR0_F		
4		PRLT!MDATreq	MDATreq_CC0_D		
5		PRUT?ICONconf			
6		PRUT!IDATreq	IDATreq_0_O		
7		PRLT?MDATind	MDATind_DT1_J		
8		PRLT?MDATind	MDATind_DT1_J		
9		PRLT?MDATind	MDATind_DT1_J		
10		PRLT?MDATind	MDATind_DT1_J		
11		PRUT?IDISind		PASS	
12		PRLT?MDATind	MDATind_CR0_F	INCONC	
13		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Estab02
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT?MDATind	MDATind_CR0_F		
4		PRLT?MDATind	MDATind_CR0_F		
5		PRLT!MDATreq	MDATreq_CC0_D		
6		PRUT?ICONconf			
7		PRUT!IDATreq	IDATreq_0_O		
8		PRLT?MDATind	MDATind_DT1_J		
9		PRLT?MDATind	MDATind_DT1_J		
10		PRLT?MDATind	MDATind_DT1_J		
11		PRLT?MDATind	MDATind_DT1_J		
12		PRUT?IDISind		PASS	
13		PRLT?MDATind	MDATind_CR0_F	INCONC	
14		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Estab03
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT?MDATind	MDATind_CR0_F		
4		PRLT?MDATind	MDATind_CR0_F		
5		PRLT?MDATind	MDATind_CR0_F		
6		PRLT!MDATreq	MDATreq_CC0_D		
7		PRUT?IDISind		INCONC	
8		PRUT?ICONconf			
9		PRLT!MDATreq	MDATreq_DR0_G		
10		PRUT?IDISind		PASS	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : NAEstab
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT?MDATind	MDATind_CR0_F		
4		PRLT?MDATind	MDATind_CR0_F		
5		PRLT?MDATind	MDATind_CR0_F		
6		PRUT?IDISind		PASS	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Dat
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT!MDATreq	MDATreq_CC0_D		
4		PRUT?ICONconf			
5		PRUT!IDATreq	IDATreq_0_O		
6		PRLT?MDATind	MDATind_DT1_J		
7		PRLT!MDATreq	MDATreq_AK0_J		
8		PRUT!IDATreq	IDATreq_0_O		
9		PRLT?MDATind	MDATind_DT0_I		
10		PRLT?MDATind	MDATind_DT0_I		
11		PRLT?MDATind	MDATind_DT0_I		
12		PRLT?MDATind	MDATind_DT0_I		
13		PRUT?IDISind		PASS	
14		PRLT?MDATind	MDATind_CR0_F	INCONC	
15		PRLT?MDATind	MDATind_DT1_J		
16		PRLT!MDATreq	MDATreq_AK0_J		
17		PRUT!IDATreq	IDATreq_0_O		
18		PRLT?MDATind	MDATind_DT1_J	INCONC	
19		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Dat01
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT!MDATreq	MDATreq_CC0_D		
4		PRUT?ICONconf			
5		PRUT!IDATreq	IDATreq_0_O		
6		PRLT?MDATind	MDATind_DT1_J		
7		PRLT?MDATind	MDATind_DT1_J		
8		PRLT!MDATreq	MDATreq_AK0_J		
9		PRUT!IDATreq	IDATreq_0_O		
10		PRLT?MDATind	MDATind_DT0_I		
11		PRLT?MDATind	MDATind_DT0_I		
12		PRLT?MDATind	MDATind_DT0_I		
13		PRLT?MDATind	MDATind_DT0_I		
14		PRUT?IDISind		PASS	
15		PRLT?MDATind	MDATind_CR0_F	INCONC	
16		PRLT?MDATind	MDATind_DT1_J		
17		PRLT?MDATind	MDATind_DT1_J		
18		PRLT!MDATreq	MDATreq_AK0_J		
19		PRUT!IDATreq	IDATreq_0_O		
20		PRLT?MDATind	MDATind_DT1_J	INCONC	
21		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Dat02
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT!MDATreq	MDATreq_CC0_D		
4		PRUT?ICONconf			
5		PRUT!IDATreq	IDATreq_0_O		
6		PRLT?MDATind	MDATind_DT1_J		
7		PRLT?MDATind	MDATind_DT1_J		
8		PRLT?MDATind	MDATind_DT1_J		
9		PRLT!MDATreq	MDATreq_AK0_J		
10		PRUT!IDATreq	IDATreq_0_O		
11		PRLT?MDATind	MDATind_DT0_I		
12		PRLT?MDATind	MDATind_DT0_I		
13		PRLT?MDATind	MDATind_DT0_I		
14		PRLT?MDATind	MDATind_DT0_I		
15		PRUT?IDISind		PASS	
16		PRLT?MDATind	MDATind_CR0_F	INCONC	
17		PRLT?MDATind	MDATind_DT1_J		
18		PRLT?MDATind	MDATind_DT1_J		
19		PRLT?MDATind	MDATind_DT1_J		
20		PRLT!MDATreq	MDATreq_AK0_J		
21		PRUT!IDATreq	IDATreq_0_O		
22		PRLT?MDATind	MDATind_DT1_J	INCONC	
23		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Dat03
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT!MDATreq	MDATreq_CC0_D		
4		PRUT?ICONconf			
5		PRUT!IDATreq	IDATreq_0_O		
6		PRLT?MDATind	MDATind_DT1_J		
7		PRLT?MDATind	MDATind_DT1_J		
8		PRLT?MDATind	MDATind_DT1_J		
9		PRLT?MDATind	MDATind_DT1_J		
10		PRLT!MDATreq	MDATreq_AK0_J		
11		PRUT!IDATreq	IDATreq_0_O		
12		PRLT?MDATind	MDATind_DT0_I		
13		PRLT?MDATind	MDATind_DT0_I		
14		PRLT?MDATind	MDATind_DT0_I		
15		PRLT?MDATind	MDATind_DT0_I		
16		PRUT?IDISind		PASS	
17		PRLT?MDATind	MDATind_CR0_F	INCONC	
18		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : NADat
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRUT!ICONreq			
2		PRLT?MDATind	MDATind_CR0_F		
3		PRLT!MDATreq	MDATreq_CC0_D		
4		PRUT?ICONconf			
5		PRUT!IDATreq	IDATreq_0_O		
6		PRLT?MDATind	MDATind_DT1_J		
7		PRLT?MDATind	MDATind_DT1_J		
8		PRLT?MDATind	MDATind_DT1_J		
9		PRLT?MDATind	MDATind_DT1_J		
10		PRUT?IDISind		PASS	
11		PRLT?MDATind	MDATind_CR0_F	INCONC	
12		PRLT?MDATind	MDATind_CR0_F	INCONC	

Detailed Comments :

Test Case Dynamic Behaviour

Test Case Name : Dis
Group :
Purpose :
Default : stddefault
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRLT!MDATreq	MDATreq_DR0_G		
2		PRUT?IDISind		PASS	

Detailed Comments :

Default Dynamic Behaviour

Default Name : stddefault
Group :
Objective :
Comments :

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		PRLT? OTHERWISE		FAIL	
2		PRUT? OTHERWISE		FAIL	

Detailed Comments :