NISTIR 90-4267

# A CONFORMANCE TEST FOR FDDI MEDIUM ACCESS CONTROL (MAC)

Zuqiu Liu
William E. Burr

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 90-4267

# A CONFORMANCE TEST FOR FDDI MEDIUM ACCESS CONTROL (MAC)

**Zuqiu Liu**
**William E. Burr**

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

July 1990

# A Conformance Test for FDDI Medium Access Control (MAC)

Zuqiu Liu
and
William E. Burr

This report describes a conformance test for the Fiber Distributed Data Interface (FDDI) Medium Access Control (MAC) standard [1]. FDDI is a layered OSI protocol consisting of four sublayers at the data link or physical layers as illustrated in Figure 1. At the lowest level is Physical Medium Dependent (PMD) standard [2], which is concerned primarily with the optical fiber interface to FDDI stations. The Physical Layer Protocol (PHY) standard [3] deals with the coding of data and control symbols. The MAC sublayer, which is a data link layer protocol, is concerned with the definition of frames (packets), and the token passing mechanism used to grant permission to transmit frames. Figure 2 illustrates the MAC frame and token formats. MAC interfaces with a Logical Link Control (LLC) sublayer, typically IEEE 802.2, above it. Finally Station Management (SMT) [4] is a vertical management sublayer, which manages the other three sublayers and provides an interface to the OSI System Management Application (SMAP) process.

Together these four sublayers specify the physical interface and data link protocol for a node in a 100 Mbit/s fiber optic token ring LAN network, which supports OSI communications. It is similar to the IEEE 802.3, 802.4 and 802.5 protocols in its services, and is logically interchangeable with them under the 802.2 LLC.

ISO 9646, *OSI Conformance Testing Methodology and Framework*, is the basis for conformance testing of OSI protocol standards. It consists of five documents, all draft international standards:

| | |
|---|---|
| Part 1: | General Concepts |
| Part 2: | Abstract Test Suite Specification |
| Part 3: | The Tree and Tabular Combined Notation (TTCN) |
| Part 4: | Test Realization |
| Part 5: | Requirements on Test Laboratories and Clients for the Conformance Assessment Process |

In this conformance testing methodology, each OSI standard should contain a *protocol implementation conformance statement* (PICS) *proforma*, which specifically enumerates the features, requirements, and options of the standard. The supplier of an *implementation under test* (IUT) is then required to complete a PICS questionnare or checklist. The completed PICS checklist then specifically states which of those features and options of the protocol have been implemented. Tests are prepared in the form of an *abstract test suite*, using the TTCN language specified in DIS 9646-3. There should be a test or tests for every feature and option specified in the PICS proforma.

FDDI MAC is specified in the equivelent ANSI X3.137-1988 and ISO 9314-2 standards. It was conceived and developed before the development of DIS 9646, without any thought for conformance testing. It does not contain a PICS proforma. The traditional means for identifying the requirements of a standard is the occurrence of the verb "shall." In a layered standard, such as MAC, which can only be tested in conjunction with other layers, it is difficult or impossible to isolate the particulars of many of the layer interfaces, and a simple enumeration of the "shalls" is not particularly helpful to identify the requirements for a meaningful test.

ISO DIS 9646 recognizes that the actual requirements of a protocol standard are often contained largely in formal state machine definition tables, and this is the case with FDDI MAC. In the absence of a PICS, the tester is left to infer which properties of the state machine are essential requirements of the standard and which properties are simply artifacts of the state machine definition. Ideally, the MAC standard and state machine would be specified so that everything which must be specified for interoperability or performance is fully specified, but only those things are specified. The MAC standard would then not constrain implementations any more than strictly necessary.

For state machine definitions as complex as MAC, this sort of perfection is very elusive. For example there is a considerable debate among the implementors of FDDI about whether the state machine requires that a station which strips a frame must begin stripping by the end of the Source Address, or if some later point will do. If the former is the correct reading of the literal MAC state machine, should that be considered a requirement of the standard or an incidental property of the state machine? The text from which the inference about when stripping must begin is less than crystal clear. There is no interoperability requirement to begin stripping so soon, since MACs must correctly handle packet fragments of any length. There may, however, be performance implications *depending upon the design assumptions of other network stations*. If an FDDI MAC PICS clearly stated the conformance requirement for stripping, this issue would be settled.

In this test suite, the authors have largely avoided such issues, or at least not intentionally tried to settle them. No attempt was made to define a PICS for FDDI MAC ( which, if provided, ought to settle the question posed above). If these tests become the definitive test for FDDI MAC conformance, they would, however, probably inadvertently make such choices on some points. It would be well for the FDDI standards developers to explicitly prepare a PICS for FDDI MAC, to provide a more explicit forum for addressing such issues, rather than to leave them to the imiagination of testers.

If the daunting task of preparing a PICS for MAC was not attempted, how did the authors determine what was to be tested? The method used was to go through the state machine specifications and attempt to devise a test which forces the state machine to go through every defined state and transition, and to observe what could be externally observed. It should be possible to work backwards from the test observations, to construct a sort of PICS. This would not be a complete PICS, because a lower level remote test cannot test every property of MAC, but it would be a start. A PICS should, however, be developed as a conscious act by the group which developed the standard.

DIS 9646 describes four abstract test methods:
(a) Local
(b) Distributed (external)
(c) Coordinated (external)

(d) Remote (external)

ISO DIS 9646 also defines both lower and upper tests, requiring test procedures to be run using a Point of Control and Observation (PCO) either above or below the IUT in the protocol stack. It is not practical to control what resides above FDDI MAC in an arbitrary IUT, at least not at any point below the applications layer. The MAC to LLC interface is not available to the tester, and several of the intervening layers may be either unknown or have many different options. Although there is no specification in the standard of which features of the MAC to LLC interface may be optional, an FDDI station is not required to implement those it does not use. Therefore, a systematic upper level test of FDDI MAC is impractical.

The MAC to PHY interface is also not directly accessible to the tester, however PHY and PMD are relatively simple, with few options, so it is possible to produce a lower level test which produces a predictable stimulus to MAC at the PHY to MAC interface. Figure 3 illustrates the practical PCO's for testing FDDI MAC and SMT. MAC Protocol Data Units (PDUs) are frames (see Figure 2). The MAC test uses a Lower Level Test with the Remote External Test Method at the MAC PDU level. The MAC test consists only of MAC PDUs sent to the FDDI input port. No assumptions are made about what may reside above FDDI in the total protocol stack, so there is no way to induce any behavior from layers above FDDI. MAC, itself, does not originate any frames except the special Beacon and Claim frames, used to initialize the FDDI ring. MAC includes no "loop-back" function which would allow a test of its ability to correctly receive a frame addressed to it, and respond by transmitting a correct response frame.

SMT, however, operates on two levels. The lower level of SMT, Connection Management (CMT), uses line state services provided by PHY, at a level below where MAC frames are apparent, and would require a PCO at the PHY level. The higher level of SMT, which uses MAC PDUs, would be tested with frames in a manner somewhat similar to MAC. However, since SMT does receive and orignate MAC frames, it is possible to test most of the frame processing capabilities of MAC by sending frames to SMT and observing the SMT originated response frames. A systematic conformance test of both MAC and SMT would begin with the CMT test, followed by a MAC test, and then a test of the SMT Frame Services.

In effect, this test is primarily a test of a MAC's ability to coexist in an FDDI network and not interfere with the proper operation of the network. It assumes the existence of a special FDDI test instrument with all the capabilities of any FDDI station plus many others. In particular, the tester requires the ability to generate a number of errors or unusual conditions, which a normal FDDI station would not be designed to generate. For example, the tester must be able to generate Frames with erroneous Frame Check Sequences, or remove tokens from the ring, or strip frames which it did not originate, which are all violations of the FDDI protocol. These functions are required of the tester to test the response of the MAC under test to specific error conditions. No FDDI tester is yet available for use to execute the tests and no actual FDDI implementations have yet been tested. It is expected that FDDI testers will become commercially available, and it is hoped that these testers will generally be implemented with microprocessors for which a C compiler is available.

The FDDI MAC conformance test is written in the Tree and Tabular Combined Notation (TTCN) defined in ISO DIS 9646-3. Two representations of TTCN are defined, a graphical form, intended to be intelligible to humans and a machine processable form, as illustrated in Figure 4. The MAC test was written in the graphical form of TTCN which has been automatically converted by the TTCN workbench developed by the University of Ottawa [5] to the machine processable form of TTCN. That machine processable TTCN form of the test was, in turn, processed by a TTCN to C translator written by David Su of the National Institute of Standards and Technology [6], which translates a TTCN test into a C program, as illustrated in Figure 5. The correctness of the syntax of that C program has in turn, been verified by compiling it on the SUN UNIX C compiler. A special FDDI tester with a C compiler would be required to execute the MAC conformance test. No such tester is yet commercially available, however the authors expect that FDDI testors will become available, as they have for other standards. Several tester specific routines which are called by the C test program must be coded for that tester to execute the conformance test.

Coding the conformance test in the TTCN language, then using a translator to convert that to C is a practical way to make a consistent, portable test. Most testors will probably use a microprocessor for which a C compiler exists. Hand translation of the TTCN test specification to some compiler or assembly language for each tester is likely to be inconsistent.

Both the workbench and the compiler implement somewhat different subsets of the TTCN language, and neither are totally current with ISO DIS 9646. The MAC conformance test, then, is coded in the intersection of the TTCN subsets implemented by the workbench and compiler. To the extent that these accurately parse TTCN, there has been a fairly strong check of the correctness of the syntax of the test. The TTCN language is still evolving, and the test presented here may not totally conform to the final version of the standard when it is approved.

Figure 6 illustrates the components of a complete test suite for FDDI protocols MAC and SMT protocols (PMD and PHY are not considered suitable for the TTCN language). SMT is not yet final and only the MAC test has been written. Figure 7 illustrates the structure of the MAC test. There are seven Test Groups:

- *Basic,* which tests the ability of the IUT MAC to transmit, receive, repeat and strip frames.

- *Claim Token,* which tests the IUT's ability to participate in the Claim Token Process used to initialize the network. The tester generates cases where the MAC under test should win the claim, and cases where it should lose.

- *Beaconing,* which tests IUT's participation in the Beacon Process, which is a test of ring continuity.

- *Timed Token Rotation Protocol*, which tests the IUT's response to the token rotation delay. When the token is late, the IUT must not use it for certain classes of traffic.

- *Monitoring*, which tests the IUT's ability to detect certain network error conditions (basically lost token or station stuck transmitting conditions) and reinitialize the ring.

- *Frame Error Detection*, which tests the ability of the IUT to detect a number of errors in frames, such as bad Frame check sequences, and respond appropriately.

- *Token Error Detection*, which tests the ability of the IUT to detect incorrectly formed tokens.

Appendix A is the TTCN graphical representation of the test, Appendix B is the machine processable version from the TTCN workbench, and Appendix C is the C program version generated by the TTCN compiler. Appendix D is an example of implementations of the host specific subroutines which must be modified or recoded in terms of the timer and I/O services provided by the particular tester used.

While a complete, systematic test of FDDI MAC implementations is probably not possible without an accessible MAC to LLC interface and a higher level MAC conformance test, this test does provide a good degree of assurance that the MAC can participate in an FDDI network without harming the network, and react properly to various network conditions which may arise. This, in conjunction with some more general application level test of the complete station, such as a file transfer test, which exercise the MAC to LLC interface, should provide a reasonable assurance that a MAC implementation conforms to the FDDI standard.

**References**

[1] ANSI X3.139-1987, *Fiber Distributed Data Interface (FDDI) Medium Access Control (MAC)*, American National Standards Institute, New York.

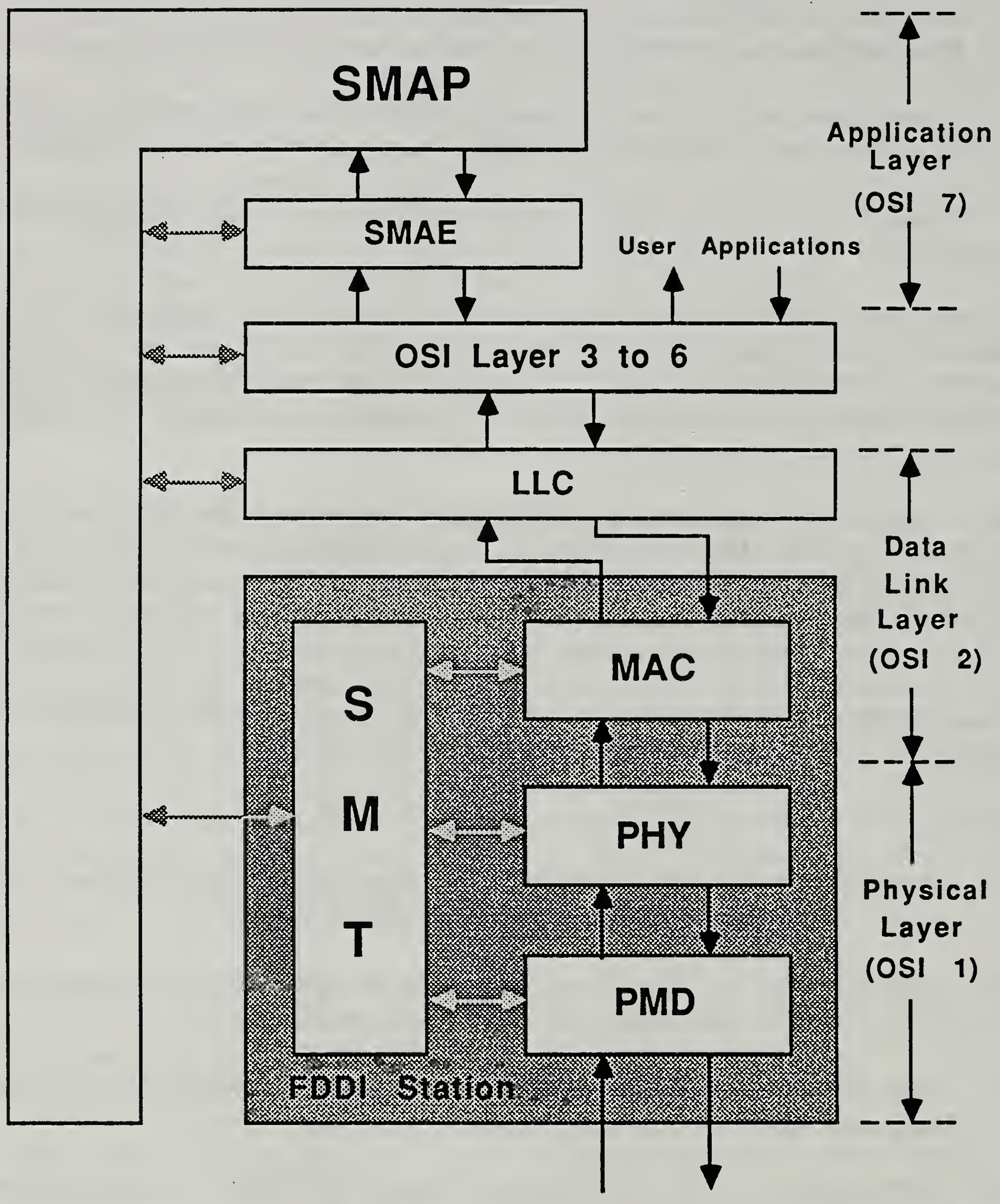
[2] ANSI X3.148-1988, *Fiber Distributed Data Interface (FDDI) Physical Layer Protocol (PHY)*, American National Standards Institute, New York.

[3] ANSI X3.166-1989, *Fiber Distributed Data Interface (FDDI) Physical Layer Medium Dependent (PMD)*, American National Standards Institute, New York.

[4] X3T9.5/84-49, rev. 5.1, *FDDI Station Management (SMT)*, a draft standard available from Global Engineering Documents, Irvine, CA.

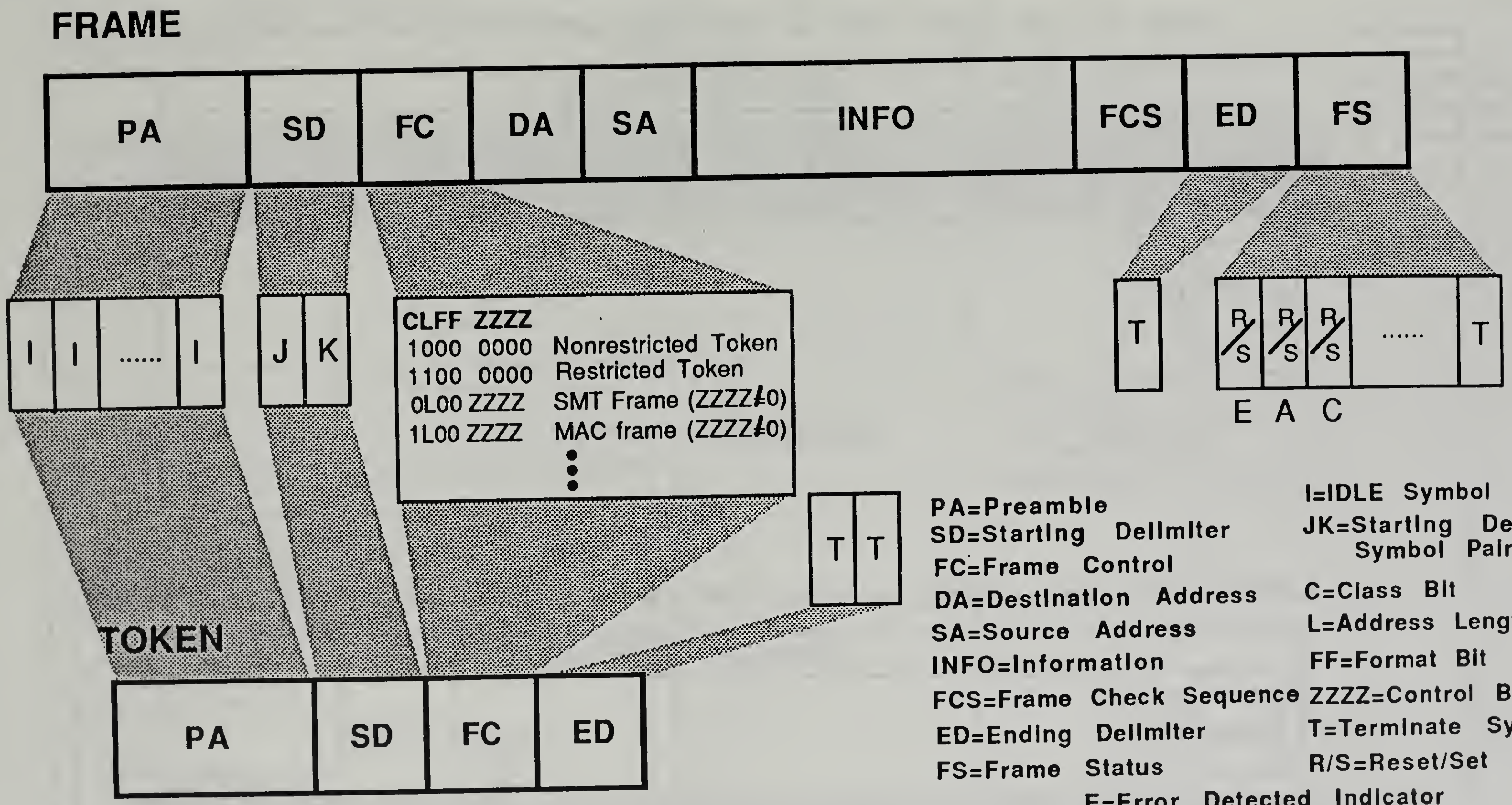
[5] "The TTCN Workbench User Guide," version 1.4, Protocols Research Group, Department of Computer Sciences, University of Ottawa, 1989.

[6] "Specification for a TTCN Translator," David H. Su, National Institute of Standards and Technology, 1989.



SMAP: System Management Application Process
SMAE: System Management Application Entity

**Figure 1. FDDI Station Relationship to OSI Model**


FRAME
PA
SD
FC
DA
SA
INFO
FCS
ED
FS
I
I
......
I
J
K
CLFF ZZZZ
1000 0000 Nonrestricted Token
1100 0000 Restricted Token
0L00 ZZZZ SMT Frame (ZZZZ≠0)
1L00 ZZZZ MAC frame (ZZZZ≠0)
T
R/S
R/S
R/S
......
T
E A C
T T
TOKEN
PA
SD
FC
ED

PA=Preamble
SD=Starting Delimiter
FC=Frame Control
DA=Destination Address
SA=Source Address
INFO=Information
FCS=Frame Check Sequence
ED=Ending Delimiter
FS=Frame Status
E=Error Detected Indicator
A=Address Recognized Indicator
C=Frame Copied Indicator

I=IDLE Symbol
JK=Starting Delimiter Symbol Pair
C=Class Bit
L=Address Length Bit
FF=Format Bit
ZZZZ=Control Bit
T=Terminate Symbol
R/S=Reset/Set Symb.

Figure 2. FDDI MAC Frame and Token Formats

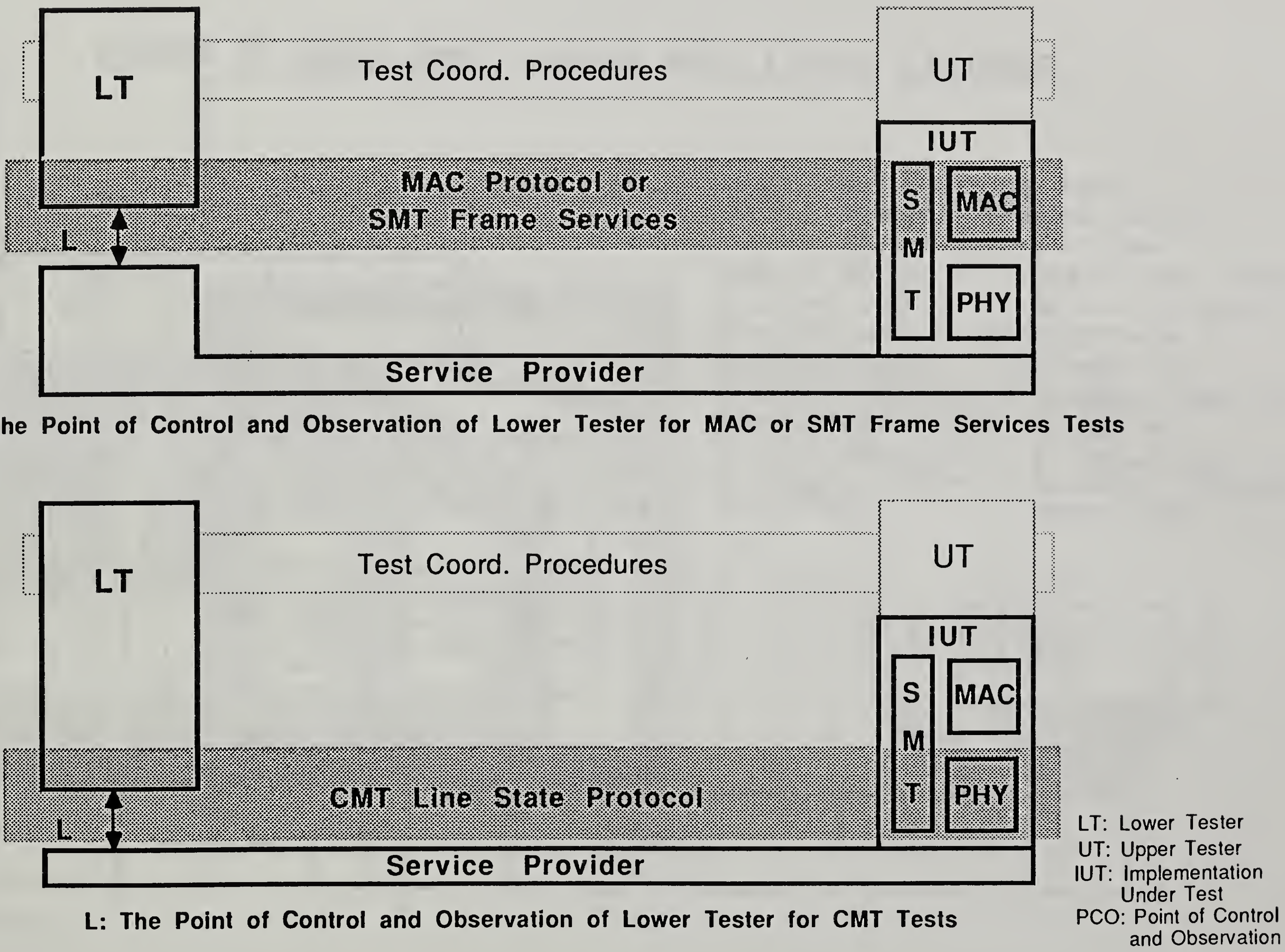
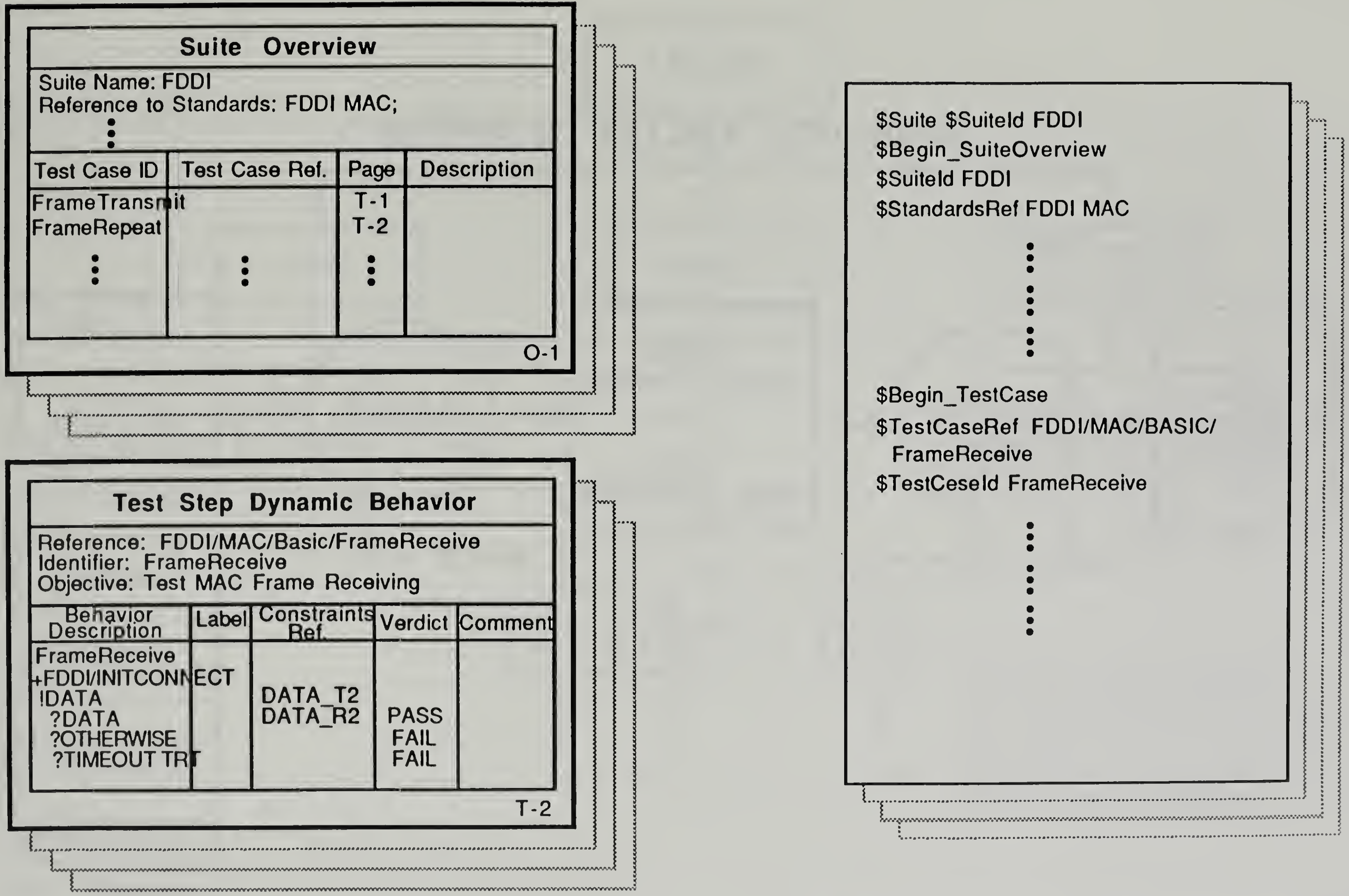

LT
Test Coord. Procedures
UT
IUT
MAC Protocol or
SMT Frame Services
S
M
T
MAC
PHY
L
Service Provider
L: The Point of Control and Observation of Lower Tester for MAC or SMT Frame Services Tests
LT
Test Coord. Procedures
UT
IUT
S
M
T
MAC
PHY
CMT Line State Protocol
L
Service Provider
L: The Point of Control and Observation of Lower Tester for CMT Tests
LT: Lower Tester
UT: Upper Tester
IUT: Implementation Under Test
PCO: Point of Control and Observation

**Figure 3. The Remote Test Method**



**Suite Overview**

Suite Name: FDDI
Reference to Standards: FDDI MAC;
⋮

| Test Case ID | Test Case Ref. | Page | Description |
|---|---|---|---|
| FrameTransmit | | T-1 | |
| FrameRepeat | | T-2 | |
| ⋮ | ⋮ | ⋮ | |

O-1

**Test Step Dynamic Behavior**

Reference: FDDI/MAC/Basic/FrameReceive
Identifier: FrameReceive
Objective: Test MAC Frame Receiving

| Behavior Description | Label | Constraints Ref. | Verdict | Comment |
|---|---|---|---|---|
| FrameReceive | | | | |
| +FDDI/INITCONNECT | | | | |
| !DATA | | DATA_T2 | | |
| ?DATA | | DATA_R2 | PASS | |
| ?OTHERWISE | | | FAIL | |
| ?TIMEOUT TRT | | | FAIL | |

T-2

```
$Suite $SuiteId FDDI
$Begin_SuiteOverview
$SuiteId FDDI
$StandardsRef FDDI MAC
        ⋮
        ⋮
$Begin_TestCase
$TestCaseRef FDDI/MAC/BASIC/
  FrameReceive
$TestCeseId FrameReceive
        ⋮
        ⋮
```

TTCN.GR(the Graphical form)

TTCN.MP(the machine processable form)

Figure 4. The Two Forms of TTCN

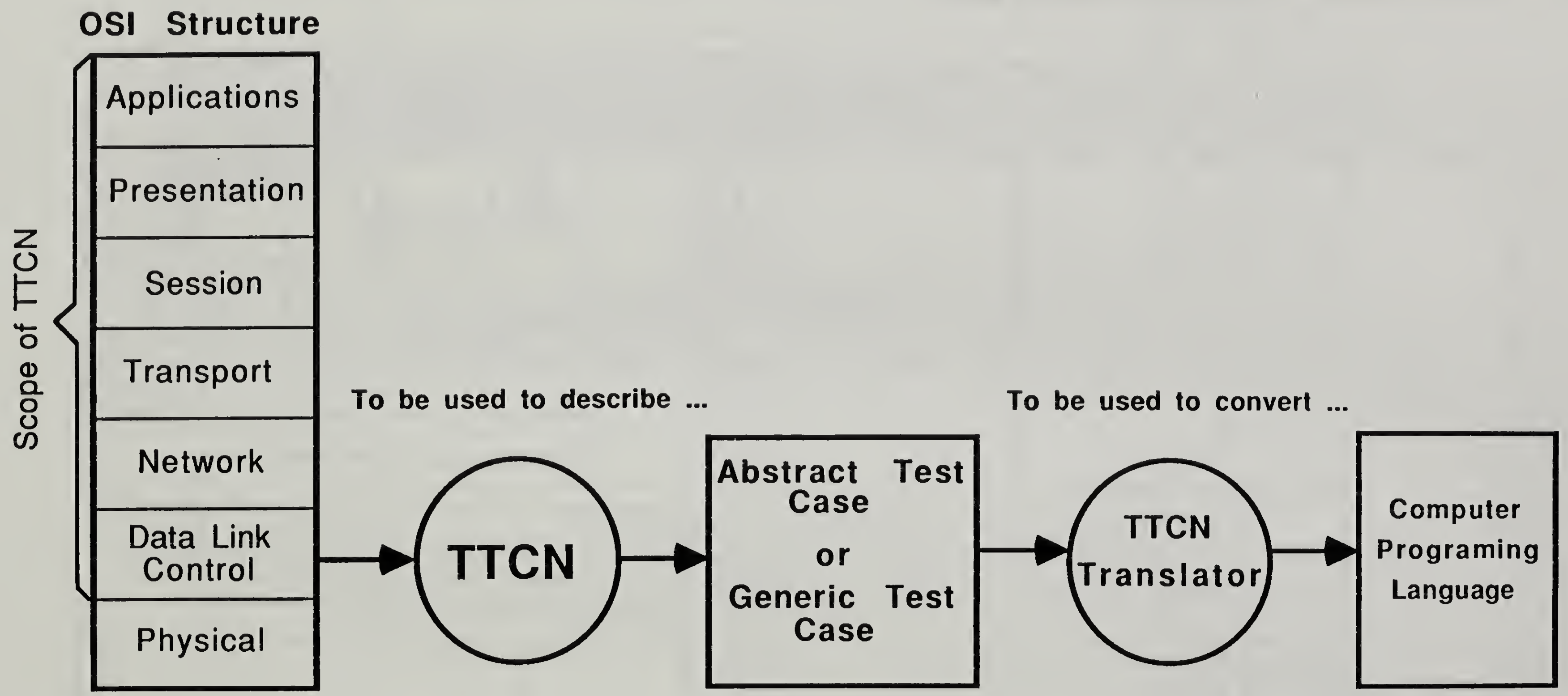

OSI Structure
Applications
Presentation
Session
Transport
Network
Data Link Control
Physical
Scope of TTCN
To be used to describe ...
TTCN
Abstract Test Case or Generic Test Case
To be used to convert ...
TTCN Translator
Computer Programing Language

Figure 5. TTCN's Functions

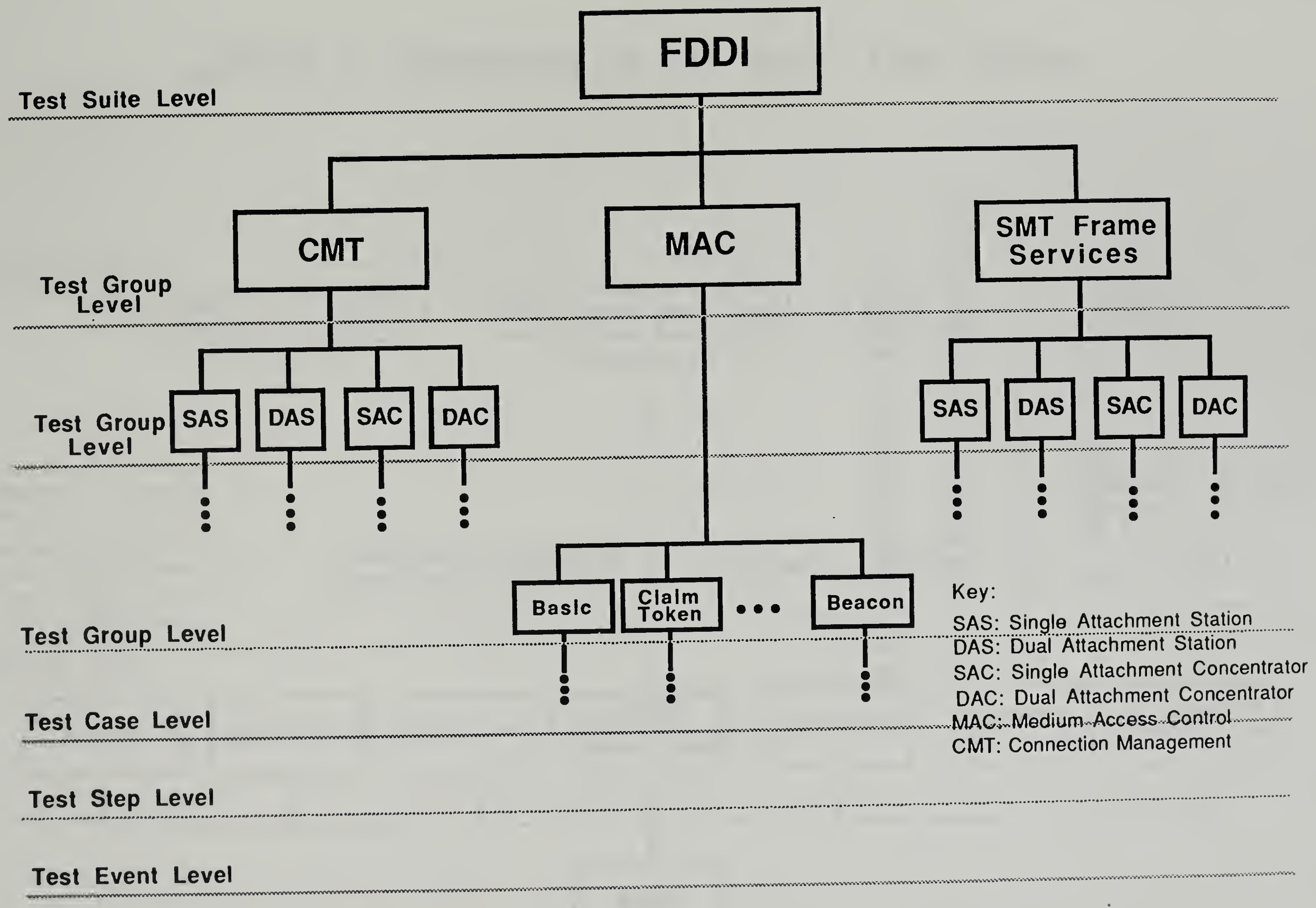


Figure 6. The FDDI Test Suite Structure

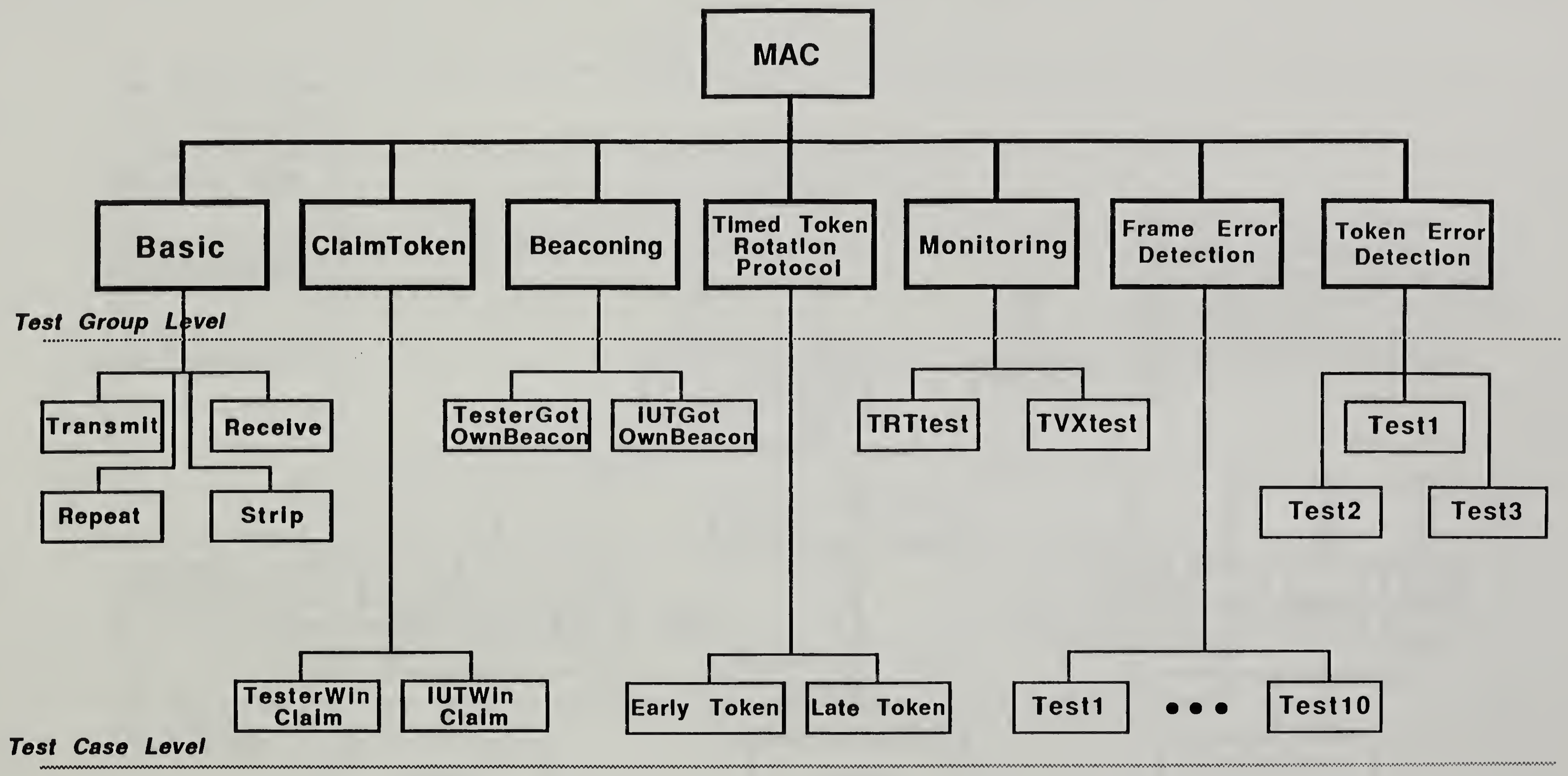


Figure 7. Structure of the MAC Test Cases

# Appendix A

## FDDI MAC Conformance Test

## TTCN Graphical Form

| Suite Overview | |
|---|---|
| Suite Name: | FDDI |
| Standards ref: | FDDI MAC (X3T9.5/88-139) |
| PICS proforma ref: | |
| PIXIT proforma ref: | |
| How Used: | |
| Test Method(s): | The Remote Test Method |
| Comments: | Test Suite for FDDI MAC Conformance Verification |

| Test Case Identifier | Test Case Reference | Page | Description |
|---|---|---|---|
| FrameTransmit # | FDDI/MAC/BASIC/FrameTransmit | T-0001 | Test MAC Frame Transmission |
| FrameRepeat # | FDDI/MAC/BASIC/FrameRepeat | T-0002 | Test MAC Frame Repeating |
| FrameReceive # | FDDI/MAC/BASIC/FrameReceive | T-0002 | Test MAC Frame Receiving |
| FrameStrip | FDDI/MAC/BASIC/FrameStrip | T-0003 | Test MAC Frame Stripping |
| TesterWinClaim # | FDDI/MAC/CLAIMTOKEN/TesterWinClaim | T-0004 | Test MAC Claim Token Process 1: Tester wins Claim Token |
| IUTWinClaim # | FDDI/MAC/CLAIMTOKEN/IUTWinClaim | T-0005 | Test MAC Claim Token process 2: IUT wins Claim Token |
| TesterGotOwnBeacon # | FDDI/MAC/BEACONING/TesterGotOwnBeacon | T-0006 | Test MAC Beacon process 1: Tester receives its own Beacon |
| IUTGotOwnBeacon # | FDDI/MAC/BEACONING/IUTGotOwnBeacon | T-0008 | Test MAC Beacon process 2: IUT receives its own Beacon |
| EarlyToken # | FDDI/MAC/TTRP/EarlyToken | T-0009 | Test MAC Timed Token Rotation protocol(TTRP) 1: Tester receives Early Token |
| LateToken # | FDDI/MAC/TTRP/LateToken | T-0010 | Test MAC Timed Token Rotation Protocol(TTRP) 2: Tester receives Late Token |
| TRTtesting # | FDDI/MAC/MONITORING/TRTtesting | T-0011 | Test MAC Monitoring function 1: TRT Monitoring Function |
| TVXtesting # | FDDI/MAC/MONITORING/TVXtesting | T-0012 | Test MAC Monitoring function 2: TVX Monitoring |
| PHInvalidR10b # | FDDI/MAC/FED/PHInvalidR10b | T-0013 | Test MAC Frame Error Detection 1: R(10b) -- a transition at MAC receive state machine |
| PHInvalidR20b # | FDDI/MAC/FED/PHInvalidR20b | T-0014 | Test MAC Frame Error Detection 2: R(20b) -- a transition at MAC receive state machine |

Continued on next page .....

..... Continued from previous page.

| Test Case Identifier | Test Case Reference | Page | Description |
|---|---|---|---|
| PHInvalidR30b<br>\# | FDDI/MAC/FED/PHInvalidR30<br>b | T-0015 | Test MAC Frame Error Detection 3: R(30b) -- a transition at MAC receive state machine |
| PHInvalidR40b<br>\# | FDDI/MAC/FED/PHInvalidR40<br>b | T-0016 | Test MAC Frame Error Detection 4: R(40b) -- a transition at MAC receive state machine |
| DetectSD | FDDI/MAC/FED/DetectSD | T-0017 | Test MAC Frame Error Detection 5: Detect SD |
| DetectFC | FDDI/MAC/FED/DetectFC | T-0018 | Test MAC Frame Error Detection 6: Detect FC |
| DetectFrameBody1<br>\# | FDDI/MAC/FED/DetectFrameB<br>ody1 | T-0019 | Test MAC Frame Error Detection 7: Detect Frame Body 1 |
| DetectFrameBody2<br>\# | FDDI/MAC/FED/DetectFrameB<br>ody2 | T-0020 | Test MAC Frame Error Detection 8: Detect Frame Body 2 |
| DetectInvalidLength<br>\# | FDDI/MAC/FED/DetectInvali<br>dLength | T-0021 | Test MAC Frame Error Detection 9: Detect Invalid Data Length Frame |
| DetectFCS | FDDI/MAC/FED/DetectFCS | T-0022 | Test MAC Frame Error Detection 10: Detect FCS Errors |
| PHInvalidR50b<br>\# | FDDI/MAC/TED/PHInvalidR50<br>b | T-0023 | Test MAC Token Error Detection 1: R(50b) -- a transition at MAC receive state machine |
| DetectTokenED1<br>\# | FDDI/MAC/TED/DetectTokenE<br>D1 | T-0023 | Test MAC Token Error Detection 2: Test 1 for ED of Token |
| DetectTokenED2<br>\# | FDDI/MAC/TED/DetectTokenE<br>D2 | T-0024 | Test MAC Token Error Detection 3: Test 2 for the ED of Token |
| **Test Step Identifier** | **Test Step Reference** | **Page** | **Description** |
| INITCONNECT<br>\# | FDDI/INITCONNECT | T-0024 | To complete ring initialization and form a Token path for test cases |
| **Default Identifier** | **Default Reference** | **Page** | **Description** |
|  |  |  |  |

| User Type Definitions | | | |
|---|---|---|---|
| Name | Base Type | Definition | Comments |
| Symbol | BITSTRING | BITSTRING[5] | The smallest signaling |
| # | | | element used by MAC. |
| Sstring | BITSTRING | ('11110'B, | SymbolString(Sstring) is |
| # | | '01001'B, | the TTCN new type |
| # | | '10100'B, | defined by user. |
| # | | '10101'B, | |
| # | | '01010'B, | |
| # | | '01011'B, | |
| # | | '01110'B, | |
| # | | '01111'B, | |
| # | | '10010'B, | |
| # | | '10011'B, | |
| # | | '10110'B, | |
| # | | '10111'B, | |
| # | | '11010'B, | |
| # | | '11011'B, | |
| # | | '11100'B, | |
| # | | '11101'B, | |
| # | | '00000'B, | |
| # | | '11111'B, | |
| # | | '00100'B, | |
| # | | '11000'B, | |
| # | | '10001'B, | |
| # | | '10101'B, | |
| # | | '01010'B, | |
| # | | '01011'B) | |

| Test Suite Parameters | | | |
|---|---|---|---|
| Name | Type | PICS/PIXIT | Comments |
| IUT_Address | INTEGER | | The address of Station |
| # | | | Under Test. |
| Tester_Address | INTEGER | | The address of Tester. |
| # | | | Get this value from |
| # | | | Tester. |
| TTRT | INTEGER (4000 | | The operative Target |
| # | TO 167772) | | Token Rotation Time. |
| T_Req_Tester | INTEGER (4000 | | Tester's Requested TTRT. |
| # | TO 167772) | | |
| T_Req_IUT | INTEGER (4000 | | IUT's Requested TTRT. |
| # | TO 167772) | | |
| T_Bid_Max | INTEGER (4000 | | Highest Bidding Value of |
| # | TO 167772) | | Tester in Claim Token |
| # | | | Process. |
| T_Bid_Min | INTEGER (4000 | | Lowest Bidding Value of |
| # | TO 167772) | | Tester in Claim Token |
| # | | | process. |
| T_FCS_Tramt | Sstring | | FCS field for the frame |
| # | | | transmitted. |
| T_FCS_Rev | Sstring | | FCS field for the frame |
| # | | | received. |

| Global Constants | | | |
|---|---|---|---|
| Name | Type | Value | Comments |
| T_Max | INTEGER | 167772 | The maximum value |
| # | | | (default value) of TRT. |
| T_Min | INTEGER | 4000 | The minimum value |
| # | | | (default value) of TRT. |
| S0 | Sstring | '11110'B | Data Symbol 0 |
| S1 | Sstring | '01001'B | Data Symbol 1 |
| S2 | Sstring | '10100'B | Data Symbol 2 |
| S3 | Sstring | '10101'B | Data Symbol 3 |
| S4 | Sstring | '01010'B | Data Symbol 4 |
| S5 | Sstring | '01011'B | Data Symbol 5 |
| S6 | Sstring | '01110'B | Data Symbol 6 |
| S7 | Sstring | '01111'B | Data Symbol 7 |
| S8 | Sstring | '10010'B | Data Symbol 8 |
| S9 | Sstring | '10011'B | Data Symbol 9 |
| sA | Sstring | '10110'B | Data Symbol A |
| SB | Sstring | '10111'B | Data Symbol B |
| SC | Sstring | '11010'B | Data Symbol C |
| sD | Sstring | '11011'B | Data Symbol D |
| SE | Sstring | '11100'B | Data Symbol E |
| SF | Sstring | '11101'B | Data Symbol F |
| SQ | Sstring | '00000'B | Line State Symbol Q |
| SI | Sstring | '11111'B | Line State Symbol I |
| SH | Sstring | '00100'B | Line State Symbol H |
| SJ | Sstring | '11000'B | Starting Delimiter J |
| SK | Sstring | '10001'B | Starting Delimiter K |
| ST | Sstring | '10101'B | Ending Delimiter T |
| SR | Sstring | '01010'B | Reset Symbol R |
| SS | Sstring | '01011'B | Set Symbol S |
| S01234567 | HEXSTRING | 'F269552DCF'H | The special data pattern |
| # | | | for INFO field. |
| SIIQIIIIIIII | HEXSTRING | 'FFC1FFFFFFFFFF | A Invalid Symbol in |
| #IIII | | FFFFFF'H | Preamble Field. |
| S0Q234567 | HEXSTRING | 'F029552DCF'H | A Invalid Symbol in the |
| # | | | data pattern of INFO |
| # | | | field. |
| S0I234567 | HEXSTRING | 'F7E9552DCF'H | A Idle Symbol in the data |
| # | | | pattern of INFO field. |
| S0T234567 | HEXSTRING | 'F569552DCF'H | A Symbol (not Idle or |
| # | | | data) in the data pattern |
| # | | | of INFO field. |
| S00000000 | HEXSTRING | 'F7BDEF7BDE'H | FCS in errors |
| SQH | BITSTRING | '0000000100'B | A symbol sequence (not |
| # | | | Idle(s) or nn) after K in |
| # | | | FC. |
| SJK | BITSTRING | '1100010001'B | Starting Delimiter. |
| SJQ | BITSTRING | '1100000000'B | A Invalid Symbol after J |
| # | | | in SD. |
| SJI | BITSTRING | '1100011111'B | A Idle Symbol after J in |
| # | | | SD. |
| SIK | BITSTRING | '1111110001'B | Other Symbol (not K |
| # | | | symbol) before K in SD. |

Continued on next page .....

..... Continued from previous page.

| Name | Type | Value | Comments |
|---|---|---|---|
| SQT<br>#| BITSTRING | '0000010101'B | A Invalid Symbol in ED of Token. |
| STT<br>#| BITSTRING | '1010110101'B | Two terminate symbols in ED of Token. |
| SIT<br>#| BITSTRING | '1111110101'B | A Idle Symbol in ED of Token. |
| SOT<br>#| BITSTRING | '1111010101'B | Any symbol(not Idles or T) in ED of Token. |
| SRRR<br>#| BITSTRING | '01010010100101<br>0'B | E=R, A=R, C=R |
| SRSS<br>#| BITSTRING | '01010010110101<br>1'B | E=R, A=S, C=S |
| SRQR<br>#| BITSTRING | '01010000000101<br>0'B | E=R, A=Q(a Invalid Symbol), C=R |
| SSSR<br>#| BITSTRING | '01011010110101<br>0'B | E=S, A=S, C=R |
| SSRR<br>#| BITSTRING | '01011010100101<br>0'B | E=S, A=R, C=R |
| SSSS<br>#| BITSTRING | '01011010110101<br>1'B | E=S, A=S, C=S |
| P_16<br>#<br>#| OCTETSTRING | 'FFFFFFFFFFFFFF<br>FFFFFF'H | The Preamble field in frame transmitted by Tester. |

| Test Suite Variables | | | |
|---|---|---|---|
| Name | Type | Value | Comments |
| a<br># | INTEGER | 0 | A variable for the test suite. |

| PCO Declarations | |
|---|---|
| Name | Role |
| L<br># | The Point of Control and Observation of lower tester for MAC, CMT or SMT frame tests. |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: DATA | | Comments: General MAC Data Frame |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| DA | Sstring[12] | Destination Address |
| SA | Sstring[12] | Source Address |
| INFO | Sstring[8956] | INFOrmation |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | End Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: TOKEN | | Comments: MAC Nonrestricted Token |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| ED | Sstring[2] | End Delimiter |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: DATA_InvLen | | Comments: The MAC frame with the Invalid Data Length |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| DA | Sstring[12] | Destination Address |
| SA | Sstring[12] | Source Address |
| INFO # | Sstring[8955] | INFOrmation field with the Invalid Data Length. |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | End Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: DATA_OverMaxLen | | Comments: The MAC frame whose symbol times is greater than TVX |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| DA | Sstring[12] | Destination Address |
| SA | Sstring[12] | Source Address |
| INFO | Sstring[8958] | INFOrmation field. This |
| # | | frame's symbol time is |
| # | | greater than TVX. |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | End Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: CLAIM | | Comments: MAC Claim Frame |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| DA | Sstring[12] | Destination Address |
| SA | Sstring[12] | Source Address |
| INFO | Sstring[8956] | INFOrmation |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | End Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: BEACON | | Comments: MAC Beacon Frame |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| DA | Sstring[12] | Destination Address |
| SA | Sstring[12] | Source Address |
| INFO | Sstring[8956] | INFOrmation |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | End Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: ECHO_Req | | Comments: SMT ECHO request Frame |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | OCTETSTRING[1] | Frame Control |
| DA | OCTETSTRING[6] | Destination Address |
| SA | OCTETSTRING[6] | Source Address |
| Frame_Class | OCTETSTRING[1] | To identify the function of |
| # | | the frame. |
| Frame_Type | OCTETSTRING[1] | To designate the type of the |
| # | | frame |
| Version_ID | OCTETSTRING[2] | The value will not change |
| # | | when upward compatible |
| # | | changes are made to the SMT |
| # | | frames. |
| Transaction_ID | OCTETSTRING[4] | To be used to pair SMT |
| # | | responses with their |
| # | | requests. |
| Station_ID | OCTETSTRING[8] | The unique identifier for an |
| # | | FDDI station (or |
| # | | concentrator). |
| Pad | OCTETSTRING[2] | |
| InfoField_Length | OCTETSTRING[2] | The length of the SMT |
| # | | Information field. |
| Parameter_Type | OCTETSTRING[2] | |
| Parameter_Length | OCTETSTRING[2] | The length of the Echo_data. |
| Echo_data | OCTETSTRING[1168] | SMT InfoField |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | Ending Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: ECHO_Resp | | Comments: SMT ECHO Response Frame |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | OCTETSTRING[1] | Frame Control |
| DA | OCTETSTRING[6] | Destination Address |
| SA | OCTETSTRING[6] | Source Address |
| Frame_Class | OCTETSTRING[1] | To identify the function of |
| # | | the frame. |
| Frame_Type | OCTETSTRING[1] | To designate the type of the |
| # | | frame. |
| Version_ID | OCTETSTRING[2] | The value will not change |
| # | | when upward compatible |
| # | | changes are made to the SMT |
| # | | frames. |
| Transaction_ID | OCTETSTRING[4] | To be used to pair SMT |
| # | | responses with their |
| # | | requests. |
| Station_ID | OCTETSTRING[8] | The unique identifier for an |
| # | | FDDI station (or |
| # | | concentrator). |
| Pad | OCTETSTRING[2] | |
| InfoField_Length | OCTETSTRING[2] | The length of the SMT |
| # | | Information field. |
| Parameter_Type | OCTETSTRING[2] | |
| Parameter_Length | OCTETSTRING[2] | The length of the Echo_data. |
| Echo_data | OCTETSTRING[1168] | SMT InfoField |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | Ending Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: DATA_Strip | | Comments: MAC Data Frame Stripped |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| DA | Sstring[12] | Destination Address |
| SA | Sstring[12] | Source Address |
| INFO | Sstring[8956] | INFOrmation |
| FCS | Sstring[8] | Frame Check Sequence |
| ED | Sstring[1] | Ending Delimiter |
| FS | Sstring[3] | Frame Status |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: TOKEN_Strip | | Comments: MAC Nonrestricted Token Stripped |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| PA | Sstring[16] | Preamble |
| SD | Sstring[2] | Starting Delimiter |
| FC | Sstring[2] | Frame Control |
| ED | Sstring[2] | Ending Delimiter |

| Data Type Declaration | | |
|---|---|---|
| PDU Name: IDLE | | Comments: |
| Protocol Control Information | | |
| Field Name | Type | Comments |
| SingleIdle # | Sstring[1] | A single Idle symbol is sent by Tester. |

| Timer Declarations | | |
|---|---|---|
| Timer Type Name | Duration | Comments |
| TVX<br>#<br># | 2621 us | Valid Transmission Timer.<br>This is the default value of<br>timeout. |
| TRT | TTRT us | Token-Rotation Timer. |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/BASIC/FrameTransmit<br>Identifier: FrameTransmit<br>Purpose: Test MAC Frame Transmission<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| FrameTransmit | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !ECHO_Req | | ECHO_Req_T1 | | Tester's |
| # | | | | ECHO Reques |
| # | | | | t Frame |
| # | | | | (DA=IUT |
| # | | | | Address). |
| !TOKEN | | TOKEN_T1 | | Tester |
| # | | | | releases |
| # | | | | Token to |
| # | | | | IUT. |
| ?ECHO_Resp | | ECHO_Resp_R1 | | IUT sends |
| # | | | | ECHO Respon |
| # | | | | se back. |
| ?TOKEN | | TOKEN_R1 | P | Tester |
| # | | | | receives |
| # | | | | Token from |
| # | | | | IUT. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| Extended Comments: | | | | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/BASIC/FrameRepeat<br>Identifier: FrameRepeat<br>Purpose: Test MAC Frame Repeating<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| FrameRepeat | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !DATA | | DATA_T1 | | Sending |
| # | | | | Tester |
| # | | | | Frame(DA<>I |
| # | | | | UT Address) |
| # | | | | . |
| ?DATA | | DATA_R1 | P | Tester gets |
| # | | | | the Tester |
| # | | | | frame repea |
| # | | | | ted (A=R; |
| # | | | | C=R; E=R). |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | I | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/BASIC/FrameReceive<br>Identifier: FrameReceive<br>Purpose: Test MAC Frame Receiving<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| FrameReceive | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !DATA | | DATA_T2 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address). |
| ?DATA | | DATA_R2 | P | Tester gets |
| # | | | | the frame |
| # | | | | (A=S; C=S; |
| # | | | | E=R) receiv |
| # | | | | ed by IUT. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/BASIC/FrameStrip | | | | |
| Identifier: FrameStrip | | | | |
| Purpose: Test MAC Frame Stripping | | | | |
| Defaults Reference: | | | | |
| **Behaviour Description** | **Label** | **Constraint Reference** | **V** | **Comments** |
| FrameStrip | | | | |
|   +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n. |
|     !ECHO_Req | | ECHO_Req_T1 | | Tester |
| # | | | | sends SMT |
| # | | | | ECHO Reques |
| # | | | | t frame, |
|       !TOKEN | | TOKEN_T1 | | and then |
| # | | | | issues |
| # | | | | Token. |
|         ?ECHO_Resp | | ECHO_Resp_R1 | | IUT sends |
| # | | | | SMT ECHO |
| # | | | | Response |
| # | | | | frame, |
|           ?TOKEN START TRT | | TOKEN_R1 | | and issues |
| # | | | | Token to |
| # | | | | Tester. |
|             !ECHO_Resp | | ECHO_Resp_T1 | | Tester |
| # | | | | received |
| # | | | | ECHO Respon |
| # | | | | se and |
| # | | | | returns |
| # | | | | this frame |
| # | | | | to IUT. |
|               ?DATA_Strip | | DATA_Strip_R6 | P | IUT strips |
| # | | | | this frame. |
|               ?OTHERWISE | | | F | |
|               ?TIMEOUT TRT | | | F | |
|           ?OTHERWISE | | | I | |
|           ?TIMEOUT TRT | | | I | |
|         ?OTHERWISE | | | I | |
|         ?TIMEOUT TRT | | | I | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/CLAIMTOKEN/TesterWinClaim | | | | |
| Identifier: TesterWinClaim | | | | |
| Purpose: Test MAC Claim Token Process 1: Tester wins Claim Token | | | | |
| Defaults Reference: | | | | |
| **Behaviour Description** | **Label** | **Constraint Reference** | **V** | **Comments** |
| TesterWinClaim | | | | |
|  +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
|   ?TIMEOUT TVX(a:=1) | A | | | Tester |
| # | | | | holds Token |
| # | | | | and sends |
| # | | | | Idle symbol |
| # | | | | s until TVX |
| # | | | | expires. |
|   !IDLE | | IDLE_I | | |
|    [a=0] | | | | |
|     GOTO A | | | | |
|    START TRT, TRTclaim, T_Max | | | | |
|     ?CLAIM START TRT, TRTclaim, T_Max | | CLAIM_R1 | | IUT issues |
| # | | | | Claim frame |
| # | | | | . |
|      !CLAIM | | CLAIM_T2 | | Tester |
| # | | | | issues its |
| # | | | | Claim frame |
| # | | | | with highes |
| # | | | | t T_Bid. |
|       ?CLAIM START TRT | | CLAIM_R2 | | IUT repeats |
| # | | | | Tester |
| # | | | | Claim Frame |
| # | | | | . |
|        !TOKEN | | TOKEN_T1 | | Tester |
| # | | | | issues |
| # | | | | Token. |
|         ?TOKEN | | TOKEN_R1 | P | IUT repeats |
| # | | | | Token. |
|         ?OTHERWISE | | | F | |
|         ?TIMEOUT TRT | | | F | |
|       ?OTHERWISE | | | F | |
|       ?TIMEOUT TRT | | | F | |
|     ?OTHERWISE | | | F | |
|     ?TIMEOUT TRT | | | F | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/CLAIMTOKEN/IUTWinClaim
Identifier: IUTWinClaim
Purpose: Test MAC Claim Token process 2: IUT wins Claim Token
Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| IUTWinClaim | | | | |
|  +FDDI/INITCONNECT | | | | |
|   ?TIMEOUT TVX(a:=1) | B | | | |
|   !IDLE | | IDLE_I | | Tester |
| # | | | | holds Token |
| # | | | | and sends |
| # | | | | Idle symbol |
| # | | | | s until TVX |
| # | | | | expires. |
|    [a=0] | | | | |
|     GOTO B | | | | |
|    START TRT, TRTclaim, T_Max | | | | |
|     ?CLAIM START TRT, TRTclaim, T_Max | | CLAIM_R1 | | IUT issues |
| # | | | | Claim frame |
| # | | | | . |
|      !CLAIM | | CLAIM_T3 | | Tester |
| # | | | | issues its |
| # | | | | Claim frame |
| # | | | | with lowest |
| # | | | | T_Bid. |
|       ?CLAIM | | CLAIM_R1 | | IUT issues |
| # | | | | its Claim |
| # | | | | frame again |
| # | | | | . |
|        !CLAIM START TRT | | CLAIM_T1 | | Tester |
| # | | | | repeats |
| # | | | | IUT's Claim |
| # | | | | frame. |
|         ?TOKEN | | TOKEN_R1 | P | IUT issues |
| # | | | | Token. |
|         ?OTHERWISE | | | F | |
|         ?TIMEOUT TRT | | | F | |
|       ?OTHERWISE | | | F | |
|       ?TIMEOUT TRT | | | F | |
|     ?OTHERWISE | | | F | |
|     ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/BEACONING/TesterGotOwnBeacon<br>Identifier: TesterGotOwnBeacon<br>Purpose: Test MAC Beacon process 1: Tester receives its own Beacon<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| TesterGotOwnBeacon | | | | |
|  +FDDI/INITCONNECT<br>#<br># | | | | Initializin<br>g Connectio<br>n |
|   ?TIMEOUT TVX(a:=1)<br>#<br>#<br>#<br>#<br># | C | | | Tester<br>holds Token<br>and sends<br>Idle symbol<br>s until TVX<br>expires. |
|   !IDLE | | IDLE_I | | |
|    [a=0] | | | | |
|     GOTO C | | | | |
|    START TRT, TRTclaim, T_Max | | | | |
|     ?CLAIM<br>#<br># | | CLAIM_R1 | | IUT issues<br>Claim frame<br>. |
|      ?TIMEOUT TRT<br>#<br>#<br>#<br>#<br>#<br>#<br># | | | | Tester<br>holds IUT's<br>Claim frame<br>to cause<br>that the<br>Claim Token<br>process<br>fails. |
|       START TRT, TRTbeacon, T_Max | | | | |
|        ?BEACON<br>#<br># | | BEACON_R1 | | IUT issues<br>Beacon<br>frame. |
|         !BEACON<br>#<br>#<br>#<br>#<br>#<br># | | BEACON_T2 | | Tester<br>holds IUT's<br>Beacon<br>frame and<br>issues its<br>own Beacon<br>frame. |
|          ?BEACON START TRT, TRTclaim,<br>#T_Max<br>#<br># | | BEACON_R2 | | IUT repeats<br>Tester's<br>Beacon<br>frame. |
|           !CLAIM<br>#<br>#<br>#<br>#<br># | | CLAIM_T4 | | After Teste<br>r receives<br>its own<br>Beacon<br>frame, it<br>sends Claim |

Continued on next page .....

..... Continued from previous page.

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| # | | | | frame. |
| ?CLAIM START TRT | | CLAIM_R4 | | IUT repeats |
| # | | | | Tester's |
| # | | | | Claim frame |
| # | | | | . |
| !TOKEN | | TOKEN_T1 | | Tester |
| # | | | | issues |
| # | | | | Token after |
| # | | | | it receives |
| # | | | | its own |
| # | | | | Claim frame |
| # | | | | . |
| ?TOKEN | | TOKEN_R1 | P | IUT repeats |
| # | | | | Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | I | |
| ?TIMEOUT TRT | | | I | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/BEACONING/IUTGotOwnBeacon
Identifier: IUTGotOwnBeacon
Purpose: Test MAC Beacon process 2: IUT receives its own Beacon
Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| IUTGotOwnBeacon | | | | |
| +FDDI_INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| ?TIMEOUT TVX(a:=1) | D | | | Tester |
| # | | | | holds Token |
| # | | | | and sends |
| # | | | | Idle symbol |
| # | | | | s until TVX |
| # | | | | expires. |
| !IDLE | | IDLE_I | | |
| [a=0] | | | | |
| GOTO D | | | | |
| START TRT, TRTclaim, T_Max | | | | |
| ?CLAIM | | CLAIM_R1 | | IUT issues |
| # | | | | Claim frame |
| # | | | | . |
| ?TIMEOUT TRT | | | | Tester |
| # | | | | holds IUT's |
| # | | | | Claim frame |
| # | | | | to cause |
| # | | | | that the |
| # | | | | Claim frame |
| # | | | | fails. |
| START TRT, TRTbeacon, T_Max | | | | |
| ?BEACON | | BEACON_R1 | | IUT issues |
| # | | | | Beacon |
| # | | | | frame. |
| !BEACON START TRT, TRTclaim, | | BEACON_T1 | | Tester |
| #T_Max | | | | repeats |
| # | | | | IUT's Beaco |
| # | | | | n frame. |
| ?CLAIM | | CLAIM_R1 | | After IUT |
| # | | | | receives |
| # | | | | its own |
| # | | | | Beacon |
| # | | | | frame, it |
| # | | | | issues |
| # | | | | Claim frame |
| # | | | | . |
| !CLAIM START TRT | | CLAIM_T1 | | Tester |
| # | | | | repeats |
| # | | | | IUT's Beaco |
| # | | | | n frame. |
| ?TOKEN | | TOKEN_R1 | P | After IUT |
| # | | | | wins Claim |

Continued on next page .....

..... Continued from previous page.

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| # | | | | Token proce |
| # | | | | ss, it |
| # | | | | issues |
| # | | | | Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | I | |
| ?TIMEOUT TRT | | | I | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/TTRP/EarlyToken
Identifier: EarlyToken
Purpose: Test MAC Timed Token Rotation protocol(TTRP) 1: Tester receives Early Token
Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| EarlyToken | | | | |
| +FDDI/INITCONNECT | | | | |
| !ECHO_Req | | ECHO_Req_T1 | | Tester |
| # | | | | sends SMT |
| # | | | | ECHO Reques |
| # | | | | t frame. |
| !TOKEN | | TOKEN_T1 | | Tester |
| # | | | | issues |
| # | | | | Token. |
| ?ECHO_Resp | | ECHO_Resp_R1 | | IUT returns |
| # | | | | ECHO Respon |
| # | | | | se frame. |
| ?TOKEN | | TOKEN_R1 | P | IUT returns |
| # | | | | Token |
| # | | | | (TRT<TTRT). |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/TTRP/LateToken
Identifier: LateToken
Purpose: Test MAC Timed Token Rotation Protocol(TTRP) 2: Tester receives Late Token
Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| LateToken | | | | |
|  +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
|   !ECHO_Req | | ECHO_Req_T1 | | Tester |
| # | | | | sends SMT |
| # | | | | ECHO Reques |
| # | | | | t frame, |
|    ?TIMEOUT TRT | | | | |
|     START TRT | | | | |
|      !TOKEN | | TOKEN_T1 | | and issues |
| # | | | | Late Token |
| # | | | | when TTRT<T |
| # | | | | RT<2xTTRT. |
|       ?TOKEN START TRT | | TOKEN_R1 | | IUT returns |
| # | | | | Token. |
|        !ECHO_Req | | ECHO_Req_T1 | | Tester |
| # | | | | sends SMT |
| # | | | | ECHO Reques |
| # | | | | t frame |
| # | | | | again, |
|         ?TIMEOUT TRT | | | | |
|          START TRT | | | | |
|           !TOKEN | | TOKEN_T1 | | and issues |
| # | | | | Late Token |
| # | | | | again when |
| # | | | | TTRT<TRT<2x |
| # | | | | TTRT. |
|            ?TOKEN | | TOKEN_R1 | P | IUT returns |
| # | | | | Token. |
|            ?OTHERWISE | | | F | |
|            ?TIMEOUT TRT | | | F | |
|       ?OTHERWISE | | | F | |
|       ?TIMEOUT TRT | | | F | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/MONITORING/TRTtesting
Identifier: TRTtesting
Purpose: Test MAC Monitoring function 1: TRT Monitoring Function
Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| TRTtesting | | | | |
|  +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
|   !ECHO_Req | | ECHO_Req_T1 | | Tester |
| # | | | | sends SMT |
| # | | | | ECHO Reques |
| # | | | | t frame. |
|    ?TIMEOUT TRT | | | | Causing TRT |
| # | | | | expired. |
|     START TRT | | | | |
|      ?TIMEOUT TRT | | | | Causing TRT |
| # | | | | expired |
| # | | | | twice and |
| # | | | | Token never |
| # | | | | has been |
| # | | | | seen. |
|       START TRT, TRTclaim, T_Max | | | | |
|        ?CLAIM | | CLAIM_R1 | | IUT initiat |
| # | | | | es Claim |
| # | | | | Token proce |
| # | | | | ss. |
|         !CLAIM START TRT | | CLAIM_T1 | | Tester |
| # | | | | repeats |
| # | | | | Claim frame |
| # | | | | . |
|          ?TOKEN | | TOKEN_R1 | P | IUT issues |
| # | | | | Token. |
|          ?OTHERWISE | | | F | |
|          ?TIMEOUT TRT | | | F | |
|        ?OTHERWISE | | | F | |
|        ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/MONITORING/TVXtesting | | | | |
| Identifier: TVXtesting | | | | |
| Purpose: Test MAC Monitoring Function 2: TVX Monitoring Function | | | | |
| Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| TVXtesting | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n(test . |
| # | | | | group) |
| +TVX_SUBTREE1 | | | | Subtree 1 |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n(test |
| # | | | | group) |
| +TVX_SUBTREE2 | | | | Subtree 2 |
| TVX_SUBTREE1 | | | | |
| !DATA_OverMaxLen | | DATA_OverMaxL | | Tester |
| # | | en_T1 | | sends teste |
| # | | | | r frame |
| # | | | | (DA=IUT |
| # | | | | Address; |
| # | | | | Frame symbo |
| # | | | | l time>TVX) |
| # | | | | , to cause |
| # | | | | TVX expire. |
| START TRT, TRTclaim, T_Max | | | | |
| ?CLAIM | | CLAIM_R1 | | IUT initiat |
| # | | | | es Claim |
| # | | | | Token proce |
| # | | | | ss. |
| !CLAIM START TRT | | CLAIM_T1 | | Tester |
| # | | | | repeats the |
| # | | | | Claim frame |
| # | | | | sent by |
| # | | | | IUT. |
| ?TOKEN | | TOKEN_R1 | (P) | IUT issues |
| # | | | | Token. |
| ?OTHERWISE | | | (F) | |
| ?TIMEOUT TRT | | | (F) | |
| ?OTHERWISE | | | (F) | |
| ?TIMEOUT TRT | | | (F) | |
| TVX_SUBTREE2 | | | | |
| ?TIMEOUT TVX(a:=1) | E | | | Holds Token |
| # | | | | and sends |
| # | | | | Idle symbol |
| # | | | | s until TVX |
| # | | | | expires. |
| !IDLE | | IDLE_I | | |
| [a=0] | | | | |
| GOTO E | | | | |

Continued on next page .....

..... Continued from previous page.

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| START TRT, TRTclaim, T_Max | | | | |
| ?CLAIM | | CLAIM_R1 | | IUT starts |
| # | | | | Claim Token |
| # | | | | process. |
| !CLAIM START TRT | | CLAIM_T1 | | |
| ?TOKEN | | TOKEN_R1 | R | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/FED/PHInvalidR10b

Identifier: PHInvalidR10b

Purpose: Test MAC Frame Error Detection 1: R(10b) -- a transition at MAC receive state machine

Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| PHInvalidR10b | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !DATA | | DATA_T3 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | A Invalid |
| # | | | | Symbol in |
| # | | | | PA) |
| !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
| ?DATA_Strip | | DATA_Strip_R1 | | Idles retur |
| # | | | | ned |
| ?TOKEN | | TOKEN_R1 | P | followed by |
| # | | | | Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/FED/PHInvalidR20b | | | | |
| Identifier: PHInvalidR20b | | | | |
| Purpose: Test MAC Frame Error Detection 2: R(20b) -- a transition at MAC receive state machine | | | | |
| Defaults Reference: | | | | |
| **Behaviour Description** | **Label** | **Constraint Reference** | **V** | **Comments** |
| PHInvalidR20b | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !DATA | | DATA_T4 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | a Invalid |
| # | | | | Symbol |
| # | | | | after J in |
| # | | | | SD) |
| !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
| ?DATA_Strip | | DATA_Strip_R1 | | PA returned |
| # | | | | , |
| ?TOKEN | | TOKEN_R1 | P | followed by |
| # | | | | Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/FED/PHInvalidR30b
Identifier: PHInvalidR30b
Purpose: Test MAC Frame Error Detection 3: R(30b) -- a transition at MAC receive state machine
Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| PHInvalidR30b | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !DATA | | DATA_T5 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | a Invalid |
| # | | | | Symbol in |
| # | | | | INFO) |
| !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
| ?DATA_Strip | | DATA_Strip_R2 | | PA,SD,FC,DA |
| # | | | | ,SA and |
| # | | | | Idles retur |
| # | | | | ned, |
| ?TOKEN | | TOKEN_R1 | P | followed by |
| # | | | | Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/FED/PHInvalidR40b | | | | |
| Identifier: PHInvalidR40b | | | | |
| Purpose: Test MAC Frame Error Detection 4: R(40b) -- a transition at MAC receive state machine | | | | |
| Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| PHInvalidR40b | | | | |
| +FDDI/INITCONNECT | | | | |
| !DATA | | DATA_T6 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | a Invalid |
| # | | | | Symbol in |
| # | | | | FS) |
| !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
| ?DATA_Strip | | DATA_Strip_R4 | | PA, SD, FC, |
| # | | | | DA, SA, |
| # | | | | INFO, FCS, |
| # | | | | ED and FS |
| # | | | | returned |
| # | | | | (E=S, A=S, |
| # | | | | C=S) |
| ?TOKEN | | TOKEN_R1 | P | followed by |
| # | | | | Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour |
|---|
| Reference: FDDI/MAC/FED/DetectSD |
| Identifier: DetectSD |
| Purpose: Test MAC Frame Error Detection 5: Detect SD |
| Defaults Reference: |

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| DetectSD | | | | |
|  +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
|   !DATA | | DATA_T7_0 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | an Idle |
| # | | | | Symbol |
| # | | | | after J) |
|    !TOKEN | | TOKEN_T1 | | Followed by |
| # | | | | Token. |
|     ?DATA_Strip | | DATA_Strip_R1 | | PA and |
| # | | | | Idles retur |
| # | | | | ned. |
|      ?TOKEN | | TOKEN_R1 | (P) | Followed by |
| # | | | | Token. |
|      ?OTHERWISE | | | (F) | |
|      ?TIMEOUT TRT | | | (F) | |
|     ?OTHERWISE | | | (F) | |
|     ?TIMEOUT TRT | | | (F) | |
|      !DATA START TRT | | DATA_T7_1 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | other symbo |
| # | | | | l(not K |
| # | | | | symbol) |
| # | | | | before K in |
| # | | | | SD) |
|       !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
|        ?DATA_Strip | | DATA_Strip_R1 | | PA and Idle |
| # | | | | returned, |
|         ?TOKEN | | TOKEN_R1 | R | followed by |
| # | | | | Token. |
|         ?OTHERWISE | | | F | |
|         ?TIMEOUT TRT | | | F | |
|        ?OTHERWISE | | | F | |
|        ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/FED/DetectFC<br>Identifier: DetectFC<br>Purpose: Test MAC Frame Error Detection 6: Detect FC<br>Defaults Reference: | | | | |
| **Behaviour Description** | **Label** | **Constraint Reference** | **V** | **Comments** |
| DetectFC | | | | |
| +FDDI/INITCONNECT | | | | Initializing Connection |
| !DATA | | DATA_T8 | | Tester sends the frame(DA=IUT Address; a symbol sequence (not Idle (s) or nn) after K in FC), |
| !TOKEN | | TOKEN_T1 | | followed by Token. |
| ?DATA_Strip | | DATA_Strip_R5 | | PA, SD and Idles returned, |
| ?TOKEN | | TOKEN_R1 | P | followed by Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/FED/DetectFrameBody1<br>Identifier: DetectFrameBody1<br>Purpose: Test MAC Frame Error Detection 7: Detect Frame Body 1<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| DetectFrameBody1 | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !DATA | | DATA_T9 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | an Idle |
| # | | | | symbol in |
| # | | | | INFO), |
| !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
| ?DATA_Strip | | DATA_Strip_R2 | | PA, SD, FC, |
| # | | | | DA, SA and |
| # | | | | Idles retur |
| # | | | | ned, |
| ?TOKEN | | TOKEN_R1 | P | followed by |
| # | | | | Token. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/FED/DetectFrameBody2<br>Identifier: DetectFrameBody2<br>Purpose: Test MAC Frame Error Detection 8: Detect Frame Body 2<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| DetectFrameBody2 | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !DATA | | DATA_T10 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | a symbol |
| # | | | | (not Idle |
| # | | | | or data) in |
| # | | | | INFO), |
| !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
| ?DATA_Strip | | DATA_Strip_R2 | | PA, SD, FC, |
| # | | | | DA, SA and |
| # | | | | Idles retur |
| # | | | | ned, |
| ?TOKEN | | TOKEN_R1 | P | followed by |
| # | | | | Token |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/FED/DetectInvalidLength | | | | |
| Identifier: DetectInvalidLength | | | | |
| Purpose: Test MAC Frame Error Detection 9: Detect Invalid Data Length Frame | | | | |
| Defaults Reference: | | | | |
| **Behaviour Description** | **Label** | **Constraint Reference** | **V** | **Comments** |
| DetectInvalidLength | | | | |
|  +FDDI/INITCONNECT | | | | |
|   !DATA_InvLen | | DATA_InvLen_T | | Tester |
| # | | 1 | | sends the |
| # | | | | frame |
| # | | | | (DA=IUT |
| # | | | | Address; a |
| # | | | | Invalid |
| # | | | | Data Length |
| # | | | | Frame), |
|    !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
|     ?DATA_Strip | | DATA_Strip_R3 | | PA, SD, FC, |
| # | | | | DA, SA, |
| # | | | | INFO, FCS, |
| # | | | | ED and FS |
| # | | | | returned |
| # | | | | (A=S, E=S, |
| # | | | | C=R), |
|      ?TOKEN | | TOKEN_R1 | P | followed by |
| # | | | | Token. |
|      ?OTHERWISE | | | F | |
|      ?TIMEOUT TRT | | | F | |
|     ?OTHERWISE | | | F | |
|     ?TIMEOUT TRT | | | F | |

Test Case Dynamic Behaviour

Reference: FDDI/MAC/FED/DetectFCS
Identifier: DetectFCS
Purpose: Test MAC Frame Error Detection 10: Detect FCS Errors
Defaults Reference:

| Behaviour Description | Label | Constraint Reference | V | Comments |
|---|---|---|---|---|
| DetectFCS | | | | |
|  +FDDI/INITCONNECT | | | | Intializing |
| # | | | | Connection |
|   !DATA | | DATA_T11_0 | | Tester . |
| # | | | | sends the |
| # | | | | frame(DA=IU |
| # | | | | T Address; |
| # | | | | FCS errors) |
| # | | | | , |
|    !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
|     ?DATA | | DATA_R11_0 | | IUT repeats |
| # | | | | Tester’s |
| # | | | | frame(A=S, |
| # | | | | E=S,C=R), |
|      ?TOKEN | | TOKEN_R1 | | Token retur |
| # | | | | ned. |
|       !DATA | | DATA_T11_1 | | Tester |
| # | | | | sends the |
| # | | | | frame(DA<>I |
| # | | | | UT Address; |
| # | | | | FCS error), |
|        !TOKEN | | TOKEN_T1 | | followed by |
| # | | | | Token. |
|         ?DATA | | DATA_R11_1 | | IUT repeats |
| # | | | | Tester’s |
| # | | | | frame(A=R, |
| # | | | | E=S, C=R) |
|          ?TOKEN | | TOKEN_R1 | P | IUT returns |
| # | | | | Token. |
|          ?OTHERWISE | | | F | |
|          ?TIMEOUT TRT | | | F | |
|         ?OTHERWISE | | | F | |
|         ?TIMEOUT TRT | | | F | |
|      ?OTHERWISE | | | F | |
|      ?TIMEOUT TRT | | | F | |
|     ?OTHERWISE | | | F | |
|     ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/TED/PHInvalidR50b<br>Identifier: PHInvalidR50b<br>Purpose: Test MAC Token Error Detection 1: R(50b) -- a transition at MAC receive state machine<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| PHInvalidR50b | | | | |
| +FDDI/INITCONNECT | | | | Initializin |
| # | | | | g Connectio |
| # | | | | n |
| !TOKEN | | TOKEN_T2 | | Tester |
| # | | | | sends the |
| # | | | | Token (a |
| # | | | | Invalid |
| # | | | | Symbol in |
| # | | | | ED), |
| ?TOKEN_Strip | | TOKEN_Strip_R | P | PA,SD,FC |
| # | | 1 | | and Idles |
| # | | | | returned. |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/TED/DetectTokenED1<br>Identifier: DetectTokenED1<br>Purpose: Test MAC Token Error Detection 2: Test 1 for ED of Token<br>Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| DetectTokenED1 | | | | |
| +FDDI/INITCONNECT | | | | |
| !TOKEN | | TOKEN_T3 | | Tester |
| # | | | | sends Token |
| # | | | | (an Idle |
| # | | | | Symbol in |
| # | | | | ED) |
| ?TOKEN_Strip | | TOKEN_Strip_R | (P) | PA, SD, FC |
| # | | 1 | | and Idles |
| # | | | | returned. |
| ?OTHERWISE | | | (F) | |
| ?TIMEOUT TRT | | | (F) | |

| Test Case Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/MAC/TED/DetectTokenED2 | | | | |
| Identifier: DetectTokenED2 | | | | |
| Purpose: Test MAC Token Error Detection 3: Test 2 for the ED of Token | | | | |
| Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| DetectTokenED2 | | | | |
| +FDDI/INITCONNECT | | | | |
| !TOKEN | | TOKEN_T4 | | Tester |
| # | | | | sends the |
| # | | | | Token(any |
| # | | | | symbol(not |
| # | | | | Idles or T) |
| # | | | | in ED) |
| ?TOKEN_Strip | | TOKEN_Strip_R | P | PA, SD, FC |
| # | | 1 | | and Idles |
| # | | | | returned |
| ?OTHERWISE | | | F | |
| ?TIMEOUT TRT | | | F | |

| Test Step Dynamic Behaviour | | | | |
|---|---|---|---|---|
| Reference: FDDI/INITCONNECT | | | | |
| Identifier: INITCONNECT | | | | |
| Objective: To complete ring initialization and form a Token path for test cases | | | | |
| Defaults Reference: | | | | |
| Behaviour Description | Label | Constraint Reference | V | Comments |
| INITCONNECT | | | | To be speci |
| # | | | | fied. |
| +CMTandTokenPath | | | | |

PDU Constraint List

PDU Name: DATA

| Constraint Name | Field Name | | | | | | | | Comments |
|---|---|---|---|---|---|---|---|---|---|
| | PA | SD | FC | A_data | INFO | FCS | ED | FS | |
| DATA_T1<br>#<br>#<br>#<br>#<br>#<br>#<br>#<br>#<br>#<br># | P_16 | SJK | '11000001'B | A1 | S01234567 | M_FCS_T<br>ramt | ST | SRRR | Note: unless FCS itself is tested, the contents in FCS will be automatically generated by Tester or IUT. It is assumed that M_FCS_Tramt is the FCS transmitted by Tester and M_FCS_Rev is the FCS transmitted by IUT. |
| DATA_R1<br># | ? | SJK | '11000001'B | A1 | S01234567 | M_FCS_R<br>ev | ST | SRRR | |
| DATA_T2<br># | P_16 | SJK | '11000001'B | A2 | S01234567 | M_FCS_T<br>ramt | ST | SRRR | |
| DATA_R2<br># | ? | SJK | '11000001'B | A2 | S01234567 | M_FCS_R<br>ev | ST | SRSS | |
| DATA_T3<br>#<br>#<br>#<br># | SIIQ<br>IIII<br>IIII<br>IIII<br>I | SJK | '11000001'B | A2 | S01234567 | M_FCS_T<br>ramt | ST | SRRR | |
| DATA_T4<br># | P_16 | SJQ | '11000001'B | A2 | S01234567 | M_FCS_T<br>ramt | ST | SRRR | |
| DATA_T5<br># | P_16 | SJK | '11000001'B | A2 | S0Q234567 | M_FCS_T<br>ramt | ST | SRRR | |
| DATA_T6<br># | P_16 | SJK | '11000001'B | A2 | S01234567 | M_FCS_T<br>ramt | ST | SRQR | |
| DATA_T7_0<br># | P_16 | SJI | '11000001'B | A2 | S01234567 | M_FCS_T<br>ramt | ST | SRRR | |
| DATA_T7_1<br># | P_16 | SIK | '11000001'B | A2 | S01234567 | M_FCS_T<br>ramt | ST | SRRR | |

Continued on next page .....

..... Continued from previous page.

| Constraint | Field Name | | | | | | | | Comments |
|---|---|---|---|---|---|---|---|---|---|
| Name | PA | SD | FC | A_data | INFO | FCS | ED | FS | |
| DATA_T8 # | P_16 | SJK | SQH | A2 | S01234567 | M_FCS_T ramt | ST | SRRR | |
| DATA_T9 # | P_16 | SJK | '11000001'B | A2 | S0I234567 | M_FCS_T ramt | ST | SRRR | |
| DATA_T10 # | P_16 | SJK | '11000001'B | A2 | S0T234567 | M_FCS_T ramt | ST | SRRR | |
| DATA_T11_0 # | P_16 | SJK | '11000001'B | A2 | S01234567 | S000000 00 | ST | SRRR | |
| DATA_R11_0 # | ? | SJK | '11000001'B | A2 | S01234567 | S000000 00 | ST | SSSR | |
| DATA_T11_1 # | P_16 | SJK | '11000001'B | A1 | S01234567 | S000000 00 | ST | SRRR | |
| DATA_R11_1 # | ? | SJK | '11000001'B | A1 | S01234567 | S000000 00 | ST | SSRR | |

53

| PDU Constraint List | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PDU Name: DATA_InvLen | | | | | | | | | |
| Constraint | Field Name | | | | | | | | Comments |
| Name | PA | SD | FC | A_data | INFO | FCS | ED | FS | |
| DATA_InvLen_T1 # | P_16 | SJK | '11000001'B | A2 | S01234567 | M_FCS_ Tramt | ST | SRRR | |

| PDU Constraint List | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PDU Name: DATA_OverMaxLen | | | | | | | | | |
| Constraint | Field Name | | | | | | | | Comments |
| Name | PA | SD | FC | A_data | INFO | FCS | ED | FS | |
| DATA_OverMaxLen #_T1 | P_16 | SJK | '11000001'B | A2 | S012345 67 | M_FCS_ Tramt | ST | SRRR | |

| PDU Constraint List | | | | | |
|---|---|---|---|---|---|
| PDU Name: TOKEN | | | | | |
| Constraint | Field Name | | | | Comments |
| Name | PA | SD | FC | ED | |
| TOKEN_T1 | P_16 | SJK | '10000000'B | STT | |
| TOKEN_R1 | ? | SJK | '10000000'B | STT | |
| TOKEN_T2 | P_16 | SJK | '10000000'B | SQT | |
| TOKEN_T3 | P_16 | SJK | '10000000'B | SIT | |
| TOKEN_T4 | P_16 | SJK | '10000000'B | SOT | |

| PDU Constraint List | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PDU Name: CLAIM | | | | | | | | | |
| Constraint Name | Field Name | | | | | | | | Comments |
| | PA | SD | FC | A_claim | INFO | FCS | ED | FS | |
| CLAIM_T1 # | P_16 | SJK | '11000011'B | B1 | T_Req_IUT | M_FCS_T ramt | ST | SRRR | |
| CLAIM_R1 # | ? | SJK | '11000011'B | B1 | T_Req_IUT | M_FCS_R ev | ST | SRRR | |
| CLAIM_T2 # | P_16 | SJK | '11000011'B | B2 | T_Bid_Max | M_FCS_T ramt | ST | SRRR | |
| CLAIM_R2 # | ? | SJK | '11000011'B | B2 | T_Bid_Max | M_FCS_R ev | ST | SRRR | |
| CLAIM_T3 # | P_16 | SJK | '11000011'B | B2 | T_Bid_Min | M_FCS_T ramt | ST | SRRR | |
| CLAIM_T4 # | P_16 | SJK | '11000011'B | B2 | T_Req_Tester | M_FCS_T ramt | ST | SRRR | |
| CLAIM_R4 # | ? | SJK | '11000011'B | B2 | T_Req_Tester | M_FCS_R ev | ST | SRRR | |

| PDU Constraint List | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PDU Name: BEACON | | | | | | | | | |
| Constraint | Field Name | | | | | | | | Comments |
| Name | PA | SD | FC | A_beacon | INFO | FCS | ED | FS | |
| BEACON_T1 # | P_16 | SJK | '11000010'B | C1 | '00000000'B | M_FCS_T ramt | ST | SRRR | |
| BEACON_R1 # | ? | SJK | '11000010'B | C1 | '00000000'B | M_FCS_R ev | ST | SRRR | |
| BEACON_T2 # | P_16 | SJK | '11000010'B | C2 | '00000000'B | M_FCS_T ramt | ST | SRRR | |
| BEACON_R2 # | ? | SJK | '11000010'B | C2 | '00000000'B | M_FCS_R ev | ST | SRRR | |

| PDU Constraint List | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PDU Name: ECHO_Req | | | | | | | | | | |
| Constraint | Field Name | | | | | | | | | Comments |
| Name | PA | SD | FC | DA | SA | SMT_HeaderandInfo | FCS | ED | FS | |
| ECHO_Req_T1 # # | P_16 | SJK | '81'H | IUT_Ad dress | Tester _Addre ss | F1 | M_FCS_T ramt | ST | SRRR | |

| PDU Constraint List | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PDU Name: ECHO_Resp | | | | | | | | | |
| Constraint | Field Name | | | | | | | | Comments |
| Name | PA | SD | FC | DA | SA | SMT_HeaderandInfo | FCS | ED | FS | |
| ECHO_Resp_R1 | ? | SJK | '81'H | Tester | IUT_Ad | F2 | M_FCS_R | ST | SRRR | |
| # | | | | _Addre | dress | | ev | | | |
| # | | | | ss | | | | | | |
| ECHO_Resp_T1 | P_16 | SJK | '81'H | Tester | IUT_Ad | F2 | M_FCS_T | ST | SRSS | |
| # | | | | _Addre | dress | | ramt | | | |
| # | | | | ss | | | | | | |

| PDU Constraint List | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PDU Name: DATA_Strip | | | | | | | | | |
| Constraint | Field Name | | | | | | | | Comments |
| Name | PA | SD | FC | A_data_strip | INFO | FCS | ED | FS | |
| DATA_Strip_R1 | ? | - | - | - | - | - | - | - | |
| DATA_Strip_R2 | ? | SJK | '11000001'B | D1 | - | - | - | - | |
| DATA_Strip_R3 | ? | SJK | '11000001'B | D1 | S01234567 | M_FCS_R | ST | SSSR | |
| # | | | | | | ev | | | |
| DATA_Strip_R4 | ? | SJK | '11000001'B | D1 | S01234567 | M_FCS_R | ST | SSSS | |
| # | | | | | | ev | | | |
| DATA_Strip_R5 | ? | SJK | - | - | - | - | - | - | |
| DATA_Strip_R6 | ? | SJK | '11000001'B | D2 | - | - | - | - | |

| PDU Constraint List | | | | | |
|---|---|---|---|---|---|
| PDU Name: TOKEN_Strip | | | | | |
| Constraint | Field Name | | | | Comments |
| Name | PA | SD | FC | ED | |
| TOKEN_Strip_R1 | ? | SJK | '10000000'B | - | |

| PDU Constraint List | | |
|---|---|---|
| PDU Name: IDLE | | |
| Constraint | Field Name | Comments |
| Name | SingleIdle | |
| IDLE_I | SI | |

| Generic Field List | | | |
|---|---|---|---|
| Generic Field Name: A_data | | | |
| Constraint | Field Name | | Comments |
| Name | DA | SA | |
| A1 | Tester_Address | Tester_Address | |
| A2 | IUT_Address | Tester_Address | |

| Generic Field List | | | |
|---|---|---|---|
| Generic Field Name: A_claim | | | |
| Constraint | Field Name | | Comments |
| Name | DA | SA | |
| B1 | IUT_Address | IUT_Address | |
| B2 | Tester_Address | Tester_Address | |

| Generic Field List | | | |
|---|---|---|---|
| Generic Field Name: A_beacon | | | |
| Constraint | Field Name | | Comments |
| Name | DA | SA | |
| C1 | '000000000000'H | IUT_Address | |
| C2 | '000000000000'H | Tester_Address | |

| Generic Field List | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Generic Field Name: SMT_HeaderandInfo | | | | | | | | | | | |
| Constraint Name | Field Name | | | | | | | | | | Comments |
| | Frame_Class | Frame_Type | Version_ID | Transaction_ID | Station_ID | Pad | InfoField_Length | Parameter_Type | Parameter_Length | Echo_data | |
| F1 | '04'H | '02'H | '0000 | '0000 | '0000 | '0000 | '116A | '0011 | '1168 | S0123 | |
| # | | | 'H | 0000' | 00000 | 'H | 'H | 'H | 'H | 4567 | |
| # | | | | H | 00000 | | | | | | |
| # | | | | | 01'H | | | | | | |
| F2 | '04'H | '03'H | '0000 | '0000 | '0000 | '0000 | '116A | '0011 | '1168 | S0123 | |
| # | | | 'H | 0000' | 00000 | 'H | 'H | 'H | 'H | 4567 | |
| # | | | | H | 00000 | | | | | | |
| # | | | | | 01'H | | | | | | |

| Generic Field List | | | |
|---|---|---|---|
| Generic Field Name: A_data_strip | | | |
| Constraint Name | Field Name | | Comments |
| | DA | SA | |
| D1 | IUT_Address | Tester_Address | |
| D2 | Tester_Address | IUT_Address | |

# Appendix B

## FDDI MAC Conformance Test

## TTCN Machine Processable Form

```
$Suite $SuiteId FDDI
$Begin_SuiteOverview
$SuiteId FDDI
$StandardsRef FDDI MAC (X3T9.5/88-139);
$PICSRef Test Suite for FDDI MAC Conformance Verification
$PIXITRef Test Suite for FDDI MAC Conformance Verification
$HowUsed Test Suite for FDDI MAC Conformance Verification
$TestMethods The Remote Test Method
$Comment Test Suite for FDDI MAC Conformance Verification $SuiteIndex
$TestCaseIndex $End_TestCaseIndex
$TestStepIndex $End_TestStepIndex
$DefaultIndex $End_DefaultIndex
$End_SuiteIndex $End_SuiteOverview
$Declarations
$UserTYPEdefs
$Begin_TTCN_TYPEdefs
$TTCN_TYPEdef
$UserTypeId Symbol
$Base BITSTRING
$TypeDef BITSTRING[5]
$Comment The smallest signaling element used by MAC. $End_TTCN_TYPEdef
$TTCN_TYPEdef
$UserTypeId Sstring
$Base BITSTRING
$TypeDef
('11110'B, '01001'B, '10100'B, '10101'B, '01010'B, '01011'B, '01110'B, '01111'B, '1(
$Comment SymbolString(Sstring) is the TTCN new type defined by user. $End_TTCN_TYPE
$End_TTCN_TYPEdefs $End_UserTYPEdefs
$Begin_TS_PARdcls
$TS_PARdcl
$TS_PARid IUT_Address
$TS_PARtype INTEGER
$PICS_PIXIT
$Comment The address of Station Under Test. $End_TS_PARdcl
$TS_PARdcl
$TS_PARid Tester_Address
$TS_PARtype INTEGER
$PICS_PIXIT
$Comment The address of Tester. Get this value from Tester. $End_TS_PARdcl
$TS_PARdcl
$TS_PARid TTRT
$TS_PARtype INTEGER (4000 TO 167772)
$PICS_PIXIT
$Comment The operative Target Token Rotation Time. $End_TS_PARdcl
$TS_PARdcl
$TS_PARid T_Req_Tester
$TS_PARtype INTEGER (4000 TO 167772)
$PICS_PIXIT
$Comment Tester's Requested TTRT. $End_TS_PARdcl
$TS_PARdcl
$TS_PARid T_Req_IUT
$TS_PARtype INTEGER (4000 TO 167772)
$PICS_PIXIT
$Comment IUT's Requested TTRT. $End_TS_PARdcl
$TS_PARdcl
$TS_PARid T_Bid_Max
$TS_PARtype INTEGER (4000 TO 167772)
$PICS_PIXIT
$Comment Highest Bidding Value of Tester in Claim Token Process. $End_TS_PARdcl
$TS_PARdcl
$TS_PARid T_Bid_Min
$TS_PARtype INTEGER (4000 TO 167772)
$PICS_PIXIT
$Comment Lowest Bidding Value of Tester in Claim Token process. $End_TS_PARdcl
$TS_PARdcl
$TS_PARid M_FCS_Rev
```

```
$TS_PARtype BITSTRING
$PICS_PIXIT
$End_TS_PARdcl
$TS_PARdcl
$TS_PARid M_FCS_Tramt
$TS_PARtype BITSTRING
$PICS_PIXIT
$End_TS_PARdcl
$End_TS_PARdcls
$Begin_CONSTdcls
$CONSTdcl
$CONSTid T_Max
$CONSTtype INTEGER
$CONSTvalue 167772
$Comment The maximum value (default value) of TRT. $End_CONSTdcl
$CONSTdcl
$CONSTid T_Min
$CONSTtype INTEGER
$CONSTvalue 4000
$Comment The minimum value (default value) of TRT. $End_CONSTdcl
$CONSTdcl
$CONSTid S0
$CONSTtype Sstring
$CONSTvalue '11110'B
$Comment Data Symbol 0 $End_CONSTdcl
$CONSTdcl
$CONSTid S1
$CONSTtype Sstring
$CONSTvalue '01001'B
$Comment Data Symbol 1 $End_CONSTdcl
$CONSTdcl
$CONSTid S2
$CONSTtype Sstring
$CONSTvalue '10100'B
$Comment Data Symbol 2 $End_CONSTdcl
$CONSTdcl
$CONSTid S3
$CONSTtype Sstring
$CONSTvalue '10101'B
$Comment Data Symbol 3 $End_CONSTdcl
$CONSTdcl
$CONSTid S4
$CONSTtype Sstring
$CONSTvalue '01010'B
$Comment Data Symbol 4 $End_CONSTdcl
$CONSTdcl
$CONSTid S5
$CONSTtype Sstring
$CONSTvalue '01011'B
$Comment Data Symbol 5 $End_CONSTdcl
$CONSTdcl
$CONSTid S6
$CONSTtype Sstring
$CONSTvalue '01110'B
$Comment Data Symbol 6 $End_CONSTdcl
$CONSTdcl
$CONSTid S7
$CONSTtype Sstring
$CONSTvalue '01111'B
$Comment Data Symbol 7 $End_CONSTdcl
$CONSTdcl
$CONSTid S8
$CONSTtype Sstring
$CONSTvalue '10010'B
$Comment Data Symbol 8 $End_CONSTdcl
$CONSTdcl
```

```
$CONSTid S9
$CONSTtype Sstring
$CONSTvalue '10011'B
$Comment Data Symbol 9 $End_CONSTdcl
$CONSTdcl
$CONSTid sA
$CONSTtype Sstring
$CONSTvalue '10110'B
$Comment Data Symbol A $End_CONSTdcl
$CONSTdcl
$CONSTid SB
$CONSTtype Sstring
$CONSTvalue '10111'B
$Comment Data Symbol B $End_CONSTdcl
$CONSTdcl
$CONSTid SC
$CONSTtype Sstring
$CONSTvalue '11010'B
$Comment Data Symbol C $End_CONSTdcl
$CONSTdcl
$CONSTid sD
$CONSTtype Sstring
$CONSTvalue '11011'B
$Comment Data Symbol D $End_CONSTdcl
$CONSTdcl
$CONSTid SE
$CONSTtype Sstring
$CONSTvalue '11100'B
$Comment Data Symbol E $End_CONSTdcl
$CONSTdcl
$CONSTid SF
$CONSTtype Sstring
$CONSTvalue '11101'B
$Comment Data Symbol F $End_CONSTdcl
$CONSTdcl
$CONSTid SQ
$CONSTtype Sstring
$CONSTvalue '00000'B
$Comment Line State Symbol Q $End_CONSTdcl
$CONSTdcl
$CONSTid SI
$CONSTtype Sstring
$CONSTvalue '11111'B
$Comment Line State Symbol I $End_CONSTdcl
$CONSTdcl
$CONSTid SH
$CONSTtype Sstring
$CONSTvalue '00100'B
$Comment Line State Symbol H $End_CONSTdcl
$CONSTdcl
$CONSTid SJ
$CONSTtype Sstring
$CONSTvalue '11000'B
$Comment Starting Delimiter J $End_CONSTdcl
$CONSTdcl
$CONSTid SK
$CONSTtype Sstring
$CONSTvalue '10001'B
$Comment Starting Delimiter K $End_CONSTdcl
$CONSTdcl
$CONSTid ST
$CONSTtype Sstring
$CONSTvalue '10101'B
$Comment Ending Delimiter T $End_CONSTdcl
$CONSTdcl
$CONSTid SR
```

```
$CONSTtype Sstring
$CONSTvalue '01010'B
$Comment Reset Symbol R $End_CONSTdcl
$CONSTdcl
$CONSTid SS
$CONSTtype Sstring
$CONSTvalue '01011'B
$Comment Set Symbol S $End_CONSTdcl
$CONSTdcl
$CONSTid S0S1S2S3S4S5S6S7
$CONSTtype HEXSTRING
$CONSTvalue 'F269552DCF'H $End_CONSTdcl
$CONSTdcl
$CONSTid SISISQSISISISISISISISISISISISI
$CONSTtype HEXSTRING
$CONSTvalue 'FFC1FFFFFFFFFFFFFFFF'H $End_CONSTdcl
$CONSTdcl
$CONSTid S0SQS2S3S4S5S6S7
$CONSTtype HEXSTRING
$CONSTvalue 'F029552DCF'H $End_CONSTdcl
$CONSTdcl
$CONSTid S0SIS2S3S4S5S6S7
$CONSTtype HEXSTRING
$CONSTvalue 'F7E9552DCF'H $End_CONSTdcl
$CONSTdcl
$CONSTid S0STS2S3S4S5S6S7
$CONSTtype HEXSTRING
$CONSTvalue 'F569552DCF'H $End_CONSTdcl
$CONSTdcl
$CONSTid SQSH
$CONSTtype BITSTRING
$CONSTvalue '0000000100'B $End_CONSTdcl
$CONSTdcl
$CONSTid SJSK
$CONSTtype BITSTRING
$CONSTvalue '1100010001'B $End_CONSTdcl
$CONSTdcl
$CONSTid SJSQ
$CONSTtype BITSTRING
$CONSTvalue '1100000000'B $End_CONSTdcl
$CONSTdcl
$CONSTid SJSI
$CONSTtype BITSTRING
$CONSTvalue '1100011111'B $End_CONSTdcl
$CONSTdcl
$CONSTid SISK
$CONSTtype BITSTRING
$CONSTvalue '1111110001'B $End_CONSTdcl
$CONSTdcl
$CONSTid SRSRSR
$CONSTtype BITSTRING
$CONSTvalue '010100101001010'B $End_CONSTdcl
$CONSTdcl
$CONSTid SRSSSS
$CONSTtype BITSTRING
$CONSTvalue '010100101101011'B $End_CONSTdcl
$CONSTdcl
$CONSTid SRSQSR
$CONSTtype BITSTRING
$CONSTvalue '010100000001010'B $End_CONSTdcl
$CONSTdcl
$CONSTid S0S0S0S0S0S0S0S0
$CONSTtype HEXSTRING
$CONSTvalue 'F7BDEF7BDE'H $End_CONSTdcl
$CONSTdcl
$CONSTid SSSSSR
```

```
$CONSTtype BITSTRING
$CONSTvalue '010110101101010'B $End_CONSTdcl
$CONSTdcl
$CONSTid SSSRSR
$CONSTtype BITSTRING
$CONSTvalue '010110101001010'B $End_CONSTdcl
$CONSTdcl
$CONSTid SQST
$CONSTtype BITSTRING
$CONSTvalue '0000010101'B $End_CONSTdcl
$CONSTdcl
$CONSTid STST
$CONSTtype BITSTRING
$CONSTvalue '1010110101'B $End_CONSTdcl
$CONSTdcl
$CONSTid SIST
$CONSTtype BITSTRING
$CONSTvalue '1111110101'B $End_CONSTdcl
$CONSTdcl
$CONSTid SOST
$CONSTtype BITSTRING
$CONSTvalue '1111010101'B $End_CONSTdcl
$CONSTdcl
$CONSTid SSSSSS
$CONSTtype BITSTRING
$CONSTvalue '010110101101011'B $End_CONSTdcl
$CONSTdcl
$CONSTid P_16
$CONSTtype OCTETSTRING
$CONSTvalue 'FFFFFFFFFFFFFFFFFFFF'H
$Comment The Preamble field in frame transmitted by Tester. $End_CONSTdcl $End_CONST
$Begin_TS_VARdcls
$TS_VARdcl
$TS_VARid a
$TS_VARtype INTEGER
$TS_VARvalue 0
$Comment A variable for the test suite. $End_TS_VARdcl $End_TS_VARdcls
$Begin_PCOdcls
$PCOdcl
$PCOid L
$Role The Point of Control and Observation of lower tester for MAC, CMT or SMT frame
$PDUdcls
$Begin_TTCN_PDUdcl
$PDUid DATA
$Comment General MAC Data Frame
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid DA
$FIELDtype Sstring[12]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype Sstring[12]
$Comment Source Address $End_FIELDdcl
```

```
$FIELDdcl
$FIELDid INFO
$FIELDtype Sstring[8956]
$Comment INFOrmation $End_FIELDdcl
$FIELDdcl
$FIELDid FCS
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[1]
$Comment End Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[3]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid TOKEN
$Comment MAC Nonrestricted Token
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[2]
$Comment End Delimiter $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid DATA_InvLen
$Comment The MAC frame with the Invalid Data Length
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid DA
$FIELDtype Sstring[12]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype Sstring[12]
$Comment Source Address $End_FIELDdcl
$FIELDdcl
$FIELDid INFO
$FIELDtype Sstring[8955]
$Comment INFOrmation field with the Invalid Data Length. $End_FIELDdcl
$FIELDdcl
$FIELDid FCS
```

```
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[2]
$Comment End Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[8]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid DATA_OverMaxLen
$Comment The MAC frame whose symbol times is greater than TVX
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid DA
$FIELDtype Sstring[12]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype Sstring[12]
$Comment Source Address $End_FIELDdcl
$FIELDdcl
$FIELDid INFO
$FIELDtype Sstring[8958]
$Comment INFOrmation field. This frame's symbol time is greater than TVX. $End_FIELI
$FIELDdcl
$FIELDid FCS
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[1]
$Comment End Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[3]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid CLAIM
$Comment MAC Claim Frame
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
```

```
$FIELDdcl
$FIELDid DA
$FIELDtype Sstring[12]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype Sstring[12]
$Comment Source Address $End_FIELDdcl
$FIELDdcl
$FIELDid INFO
$FIELDtype Sstring[8956]
$Comment INFOrmation $End_FIELDdcl
$FIELDdcl
$FIELDid FCS
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[1]
$Comment End Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[3]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid BEACON
$Comment MAC Beacon Frame
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid DA
$FIELDtype Sstring[12]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype Sstring[12]
$Comment Source Address $End_FIELDdcl
$FIELDdcl
$FIELDid INFO
$FIELDtype Sstring[8956]
$Comment INFOrmation $End_FIELDdcl
$FIELDdcl
$FIELDid FCS
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[1]
$Comment End Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[3]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid ECHO_Req
```

```
$Comment SMT ECHO request Frame
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype OCTETSTRING[1]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid DA
$FIELDtype OCTETSTRING[6]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype OCTETSTRING[6]
$Comment Source Address $End_FIELDdcl
$FIELDdcl
$FIELDid Frame_Class
$FIELDtype OCTETSTRING[1]
$Comment To identify the function of the frame. $End_FIELDdcl
$FIELDdcl
$FIELDid Frame_Type
$FIELDtype OCTETSTRING[1]
$Comment To designate the type of the frame $End_FIELDdcl
$FIELDdcl
$FIELDid Version_ID
$FIELDtype OCTETSTRING[2]
$Comment The value will not change when upward compatible changes are made to the S
$FIELDdcl
$FIELDid Transaction_ID
$FIELDtype OCTETSTRING[4]
$Comment To be used to pair SMT responses with their requests. $End_FIELDdcl
$FIELDdcl
$FIELDid Station_ID
$FIELDtype OCTETSTRING[8]
$Comment The unique identifier for an FDDI station (or concentrator). $End_FIELDdcl
$FIELDdcl
$FIELDid Pad
$FIELDtype  OCTETSTRING[2] $End_FIELDdcl
$FIELDdcl
$FIELDid InfoField_Length
$FIELDtype OCTETSTRING[2]
$Comment The length of the SMT Information field. $End_FIELDdcl
$FIELDdcl
$FIELDid Parameter_Type
$FIELDtype  OCTETSTRING[2] $End_FIELDdcl
$FIELDdcl
$FIELDid Parameter_Length
$FIELDtype OCTETSTRING[2]
$Comment The length of the Echo_data. $End_FIELDdcl
$FIELDdcl
$FIELDid Echo_data
$FIELDtype OCTETSTRING[1168]
$Comment SMT InfoField $End_FIELDdcl
$FIELDdcl
$FIELDid FCS
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
```

```
$FIELDtype Sstring[1]
$Comment Ending Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[3]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid ECHO_Resp
$Comment SMT ECHO Response Frame
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype OCTETSTRING[1]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid DA
$FIELDtype OCTETSTRING[6]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype OCTETSTRING[6]
$Comment Source Address $End_FIELDdcl
$FIELDdcl
$FIELDid Frame_Class
$FIELDtype OCTETSTRING[1]
$Comment To identify the function of the frame. $End_FIELDdcl
$FIELDdcl
$FIELDid Frame_Type
$FIELDtype OCTETSTRING[1]
$Comment To designate the type of the frame. $End_FIELDdcl
$FIELDdcl
$FIELDid Version_ID
$FIELDtype OCTETSTRING[2]
$Comment The value will not change when upward compatible changes are made to the SM
$FIELDdcl
$FIELDid Transaction_ID
$FIELDtype OCTETSTRING[4]
$Comment To be used to pair SMT responses with their requests. $End_FIELDdcl
$FIELDdcl
$FIELDid Station_ID
$FIELDtype OCTETSTRING[8]
$Comment The unique identifier for an FDDI station (or concentrator). $End_FIELDdcl
$FIELDdcl
$FIELDid Pad
$FIELDtype OCTETSTRING[2] $End_FIELDdcl
$FIELDdcl
$FIELDid InfoField_Length
$FIELDtype OCTETSTRING[2]
$Comment The length of the SMT Information field. $End_FIELDdcl
$FIELDdcl
$FIELDid Parameter_Type
$FIELDtype OCTETSTRING[2] $End_FIELDdcl
$FIELDdcl
$FIELDid Parameter_Length
$FIELDtype OCTETSTRING[2]
$Comment The length of the Echo_data. $End_FIELDdcl
$FIELDdcl
$FIELDid Echo_data
```

```
$FIELDtype OCTETSTRING[1168]
$Comment SMT InfoField $End_FIELDdcl
$FIELDdcl
$FIELDid FCS
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[1]
$Comment Ending Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[3]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid DATA_Strip
$Comment MAC Data Frame Stripped
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid DA
$FIELDtype Sstring[12]
$Comment Destination Address $End_FIELDdcl
$FIELDdcl
$FIELDid SA
$FIELDtype Sstring[12]
$Comment Source Address $End_FIELDdcl
$FIELDdcl
$FIELDid INFO
$FIELDtype Sstring[8956]
$Comment INFOrmation $End_FIELDdcl
$FIELDdcl
$FIELDid FCS
$FIELDtype Sstring[8]
$Comment Frame Check Sequence $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[1]
$Comment Ending Delimiter $End_FIELDdcl
$FIELDdcl
$FIELDid FS
$FIELDtype Sstring[3]
$Comment Frame Status $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid TOKEN_Strip
$Comment MAC Nonrestricted Token Stripped
$PCI
$FIELDdcl
$FIELDid PA
$FIELDtype Sstring[16]
$Comment Preamble $End_FIELDdcl
$FIELDdcl
$FIELDid SD
$FIELDtype Sstring[2]
$Comment Starting Delimiter $End_FIELDdcl
```

```
$FIELDdcl
$FIELDid FC
$FIELDtype Sstring[2]
$Comment Frame Control $End_FIELDdcl
$FIELDdcl
$FIELDid ED
$FIELDtype Sstring[2]
$Comment Ending Delimiter $End_FIELDdcl $End_PCI $End_TTCN_PDUdcl
$Begin_TTCN_PDUdcl
$PDUid IDLE
$PCI
$FIELDdcl
$FIELDid SingleIdle
$FIELDtype Sstring[1]
$Comment A single Idle symbol is sent by Tester. $End_FIELDdcl $End_PCI $End_TTCN_PI
$Begin_TIMERdcls
$TIMERdcl
$TimerTypeId TVX
$Duration 2621 us
$Comment Valid Transmission Timer. This is the default value of timeout. $End_TIMERc
$TIMERdcl
$TimerTypeId TRT
$Duration TTRT us
$Comment Token-Rotation Timer. $End_TIMERdcl $End_TIMERdcls $End_Declarations
$DynamicPart
$TestCases
$TestGroup $TestGroupID MAC
$TestGroup $TestGroupID BASIC
$Begin_TestCase
$TestCaseRef FDDI/MAC/BASIC/FrameTransmit
$TestCaseId FrameTransmit
$TestPurpose Testing MAC Frame Transmission
$BehaviourDescription
$TreeHeader FrameTransmit
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!ECHO_Req
$CRef ECHO_Req_T1
$Comment Tester's ECHO Request Frame(DA=IUT Address). $End_BehaviourLine
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment Tester releases Token to IUT. $End_BehaviourLine
$BehaviourLine
$Line [4]?ECHO_Resp
$CRef ECHO_Resp_R1
$Comment IUT sends ECHO Response back. $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment Tester receives Token from IUT. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription
```

```
$ExtComments $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/BASIC/FrameRepeat
$TestCaseId FrameRepeat
$TestPurpose Testing MAC Frame Repeating
$BehaviourDescription
$TreeHeader FrameRepeat
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T1
$Comment Sending Tester Frame(DA<>IUT Address). $End_BehaviourLine
$BehaviourLine
$Line [3]?DATA
$CRef DATA_R1
$Verdict P
$Comment Tester gets the Tester frame repeated (A=R; C=R; E=R). $End_BehaviourLine
$BehaviourLine
$Line [3]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Verdict I $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/BASIC/FrameReceive
$TestCaseId FrameReceive
$TestPurpose Testing MAC Frame Receiving
$BehaviourDescription
$TreeHeader FrameReceive
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T2
$Comment Tester sends the frame(DA=IUT Address). $End_BehaviourLine
$BehaviourLine
$Line [3]?DATA
$CRef DATA_R2
$Verdict P
$Comment Tester gets the frame (A=S; C=S; E=R) received by IUT. $End_BehaviourLine
$BehaviourLine
$Line [3]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/BASIC/FrameStrip
$TestCaseId FrameStrip
$TestPurpose Testing MAC Frame Stripping
$BehaviourDescription
$TreeHeader FrameStrip
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection. $End_BehaviourLine
$BehaviourLine
$Line [2]!ECHO_Req
$CRef ECHO_Req_T1
$Comment Tester sends SMT ECHO Request frame, $End_BehaviourLine
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment and then issues Token. $End_BehaviourLine
```

```
$BehaviourLine
$Line [4]?ECHO_Resp
$CRef ECHO_Resp_R1
$Comment IUT sends SMT ECHO Response frame, $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN START TRT
$CRef TOKEN_R1
$Comment and issues Token to Tester. $End_BehaviourLine
$BehaviourLine
$Line [6]!ECHO_Resp
$CRef ECHO_Resp_T1
$Comment Tester received ECHO Response and returns this frame to IUT. $End_Behaviou
$BehaviourLine
$Line [7]?DATA_Strip
$CRef DATA_Strip_R6
$Verdict P
$Comment IUT strips this frame. $End_BehaviourLine
$BehaviourLine
$Line [7]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict I $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict I $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict I $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict I $End_BehaviourLine $End_BehaviourDescription $End_TestCase $End_TestGrou
$TestGroup $TestGroupID CLAIMTOKEN
$Begin_TestCase
$TestCaseRef FDDI/MAC/CLAIMTOKEN/TesterWinClaim
$TestCaseId TesterWinClaim
$TestPurpose Testing MAC Claim Token Process 1: Tester wins Claim Token
$BehaviourDescription
$TreeHeader TesterWinClaim
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]?TIMEOUT TVX(a:=1)
$Label A
$Comment Tester holds Token and sends Idle symbols until TVX expires. $End_Behaviou
$BehaviourLine
$Line [2]!IDLE
$CRef IDLE_I $End_BehaviourLine
$BehaviourLine
$Line [3][a=0] $End_BehaviourLine
$BehaviourLine
$Line [4]GOTO A $End_BehaviourLine
$BehaviourLine
$Line [3]START TRT, TRTclaim, T_Max $End_BehaviourLine
$BehaviourLine
$Line [4]?CLAIM START TRT, TRTclaim, T_Max
$CRef CLAIM_R1
$Comment IUT issues Claim frame. $End_BehaviourLine
$BehaviourLine
$Line [5]!CLAIM
$CRef CLAIM_T2
$Comment Tester issues its Claim frame with highest T_Bid. $End_BehaviourLine
```

```
$BehaviourLine
$Line [6]?CLAIM START TRT
$CRef CLAIM_R2
$Comment IUT repeats Tester Claim Frame. $End_BehaviourLine
$BehaviourLine
$Line [7]!TOKEN
$CRef TOKEN_T1
$Comment Tester issues Token. $End_BehaviourLine
$BehaviourLine
$Line [8]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment IUT repeats Token. $End_BehaviourLine
$BehaviourLine
$Line [8]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [8]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [6]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [6]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/CLAIMTOKEN/IUTWinClaim
$TestCaseId IUTWinClaim
$TestPurpose Testing MAC Claim Token process 2: IUT wins Claim Token
$BehaviourDescription
$TreeHeader IUTWinClaim
$BehaviourLine
$Line [1]+FDDI/INITCONNECT $End_BehaviourLine
$BehaviourLine
$Line [2]?TIMEOUT TVX(a:=1)
$Label B $End_BehaviourLine
$BehaviourLine
$Line [2]!IDLE
$CRef IDLE_I
$Comment Tester holds Token and sends Idle symbols until TVX expires. $End_Behaviou
$BehaviourLine
$Line [3][a=0] $End_BehaviourLine
$BehaviourLine
$Line [4]GOTO B $End_BehaviourLine
$BehaviourLine
$Line [3]START TRT, TRTclaim, T_Max $End_BehaviourLine
$BehaviourLine
$Line [4]?CLAIM START TRT, TRTclaim, T_Max
$CRef CLAIM_R1
$Comment IUT issues Claim frame. $End_BehaviourLine
$BehaviourLine
$Line [5]!CLAIM
$CRef CLAIM_T3
$Comment Tester issues its Claim frame with lowest T_Bid. $End_BehaviourLine
$BehaviourLine
$Line [6]?CLAIM
$CRef CLAIM_R1
$Comment IUT issues its Claim frame again. $End_BehaviourLine
$BehaviourLine
$Line [7]!CLAIM START TRT
```

```
$CRef CLAIM_T1
$Comment Tester repeats IUT's Claim frame. $End_BehaviourLine
$BehaviourLine
$Line [8]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment IUT issues Token. $End_BehaviourLine
$BehaviourLine
$Line [8]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [8]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [6]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [6]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase $End_TestGrou
$TestGroup $TestGroupID BEACONING
$Begin_TestCase
$TestCaseRef FDDI/MAC/BEACONING/TesterGotOwnBeacon
$TestCaseId TesterGotOwnBeacon
$TestPurpose Testing MAC Beacon process 1: Tester receives its own Beacon
$BehaviourDescription
$TreeHeader TesterGotOwnBeacon
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]?TIMEOUT TVX(a:=1)
$Label C
$Comment Tester holds Token and sends Idle symbols until TVX expires. $End_Behaviou
$BehaviourLine
$Line [2]!IDLE
$CRef IDLE_I  $End_BehaviourLine
$BehaviourLine
$Line [3][a=0] $End_BehaviourLine
$BehaviourLine
$Line [4]GOTO C $End_BehaviourLine
$BehaviourLine
$Line [3]START TRT, TRTclaim, T_Max $End_BehaviourLine
$BehaviourLine
$Line [4]?CLAIM
$CRef CLAIM_R1
$Comment IUT issues Claim frame. $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Comment Tester holds IUT's Claim frame to cause that the Claim Token process fails.
$BehaviourLine
$Line [6]START TRT, TRTbeacon, T_Max $End_BehaviourLine
$BehaviourLine
$Line [7]?BEACON
$CRef BEACON_R1
$Comment IUT issues Beacon frame. $End_BehaviourLine
$BehaviourLine
$Line [8]!BEACON
$CRef BEACON_T2
$Comment Tester holds IUT's Beacon frame and issues its own Beacon frame. $End_Beha
$BehaviourLine
```

```
$Line [9]?BEACON START TRT, TRTclaim, T_Max
$CRef BEACON_R2
$Comment IUT repeats Tester's Beacon frame. $End_BehaviourLine
$BehaviourLine
$Line [10]!CLAIM
$CRef CLAIM_T4
$Comment After Tester receives its own Beacon frame, it sends Claim frame. $End_Behа
$BehaviourLine
$Line [11]?CLAIM START TRT
$CRef CLAIM_R4
$Comment IUT repeats Tester's Claim frame. $End_BehaviourLine
$BehaviourLine
$Line [12]!TOKEN
$CRef TOKEN_T1
$Comment Tester issues Token after it receives its own Claim frame. $End_BehaviourLi
$BehaviourLine
$Line [13]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment IUT repeats Token. $End_BehaviourLine
$BehaviourLine
$Line [13]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [13]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [11]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [11]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [9]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [9]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict I $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict I $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/BEACONING/IUTGotOwnBeacon
$TestCaseId IUTGotOwnBeacon
$TestPurpose Testing MAC Beacon process 2: IUT receives its own Beacon
$BehaviourDescription
$TreeHeader IUTGotOwnBeacon
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]?TIMEOUT TVX(a:=1)
$Label D
$Comment Tester holds Token and sends Idle symbols until TVX expires. $End_Behaviour
$BehaviourLine
$Line [2]!IDLE
$CRef IDLE_I $End_BehaviourLine
```

```
$BehaviourLine
$Line [3][a=0] $End_BehaviourLine
$BehaviourLine
$Line [4]GOTO D $End_BehaviourLine
$BehaviourLine
$Line [3]START TRT, TRTclaim, T_Max $End_BehaviourLine
$BehaviourLine
$Line [4]?CLAIM
$CRef CLAIM_R1
$Comment IUT issues Claim frame. $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Comment Tester holds IUT's Claim frame to cause that the Claim frame fails. $End_Be
$BehaviourLine
$Line [6]START TRT, TRTbeacon, T_Max $End_BehaviourLine
$BehaviourLine
$Line [7]?BEACON
$CRef BEACON_R1
$Comment IUT issues Beacon frame. $End_BehaviourLine
$BehaviourLine
$Line [8]!BEACON START TRT, TRTclaim, T_Max
$CRef BEACON_T1
$Comment Tester repeats IUT's Beacon frame. $End_BehaviourLine
$BehaviourLine
$Line [9]?CLAIM
$CRef CLAIM_R1
$Comment After IUT receives its own Beacon frame, it issues Claim frame. $End_Behavi
$BehaviourLine
$Line [10]!CLAIM START TRT
$CRef CLAIM_T1
$Comment Tester repeats IUT's Beacon frame. $End_BehaviourLine
$BehaviourLine
$Line [11]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment After IUT wins Claim Token process, it issues Token. $End_BehaviourLine
$BehaviourLine
$Line [11]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [11]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [9]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [9]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict I $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict I $End_BehaviourLine $End_BehaviourDescription $End_TestCase $End_TestGrour
$TestGroup $TestGroupID TTRP
$Begin_TestCase
$TestCaseRef FDDI/MAC/TTRP/EarlyToken
$TestCaseId EarlyToken
$TestPurpose Testing MAC Timed Token Rotation protocol(TTRP) 1: Tester receives Earl
$BehaviourDescription
```

```
$TreeHeader EarlyToken
$BehaviourLine
$Line [1]+FDDI/INITCONNECT $End_BehaviourLine
$BehaviourLine
$Line [2]!ECHO_Req
$CRef ECHO_Req_T1
$Comment Tester sends SMT ECHO Request frame. $End_BehaviourLine
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment Tester issues Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?ECHO_Resp
$CRef ECHO_Resp_R1
$Comment IUT returns ECHO Response frame. $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment IUT returns Token (TRT<TTRT). $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/TTRP/LateToken
$TestCaseId LateToken
$TestPurpose Testing MAC Timed Token Rotation Protocol(TTRP) 2: Tester receives Late
$BehaviourDescription
$TreeHeader LateToken
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!ECHO_Req
$CRef ECHO_Req_T1
$Comment Tester sends SMT ECHO Request frame, $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT $End_BehaviourLine
$BehaviourLine
$Line [4]START TRT $End_BehaviourLine
$BehaviourLine
$Line [5]!TOKEN
$CRef TOKEN_T1
$Comment and issues Late Token when TTRT<TRT<2xTTRT. $End_BehaviourLine
$BehaviourLine
$Line [6]?TOKEN START TRT
$CRef TOKEN_R1
$Comment IUT returns Token. $End_BehaviourLine
$BehaviourLine
$Line [7]!ECHO_Req
$CRef ECHO_Req_T1
$Comment Tester sends SMT ECHO Request frame again, $End_BehaviourLine
$BehaviourLine
$Line [8]?TIMEOUT TRT $End_BehaviourLine
$BehaviourLine
$Line [9]START TRT $End_BehaviourLine
$BehaviourLine
```

```
$Line [10]!TOKEN
$CRef TOKEN_T1
$Comment and issues Late Token again when TTRT<TRT<2xTTRT. $End_BehaviourLine
$BehaviourLine
$Line [11]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment IUT returns Token. $End_BehaviourLine
$BehaviourLine
$Line [11]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [11]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [6]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [6]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase $End_TestGrou
$TestGroup $TestGroupID MONITORING
$Begin_TestCase
$TestCaseRef FDDI/MAC/MONITORING/TRTtesting
$TestCaseId TRTtesting
$TestPurpose Testing MAC Monitoring function 1: TRT Monitoring Function
$BehaviourDescription
$TreeHeader TRTtesting
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!ECHO_Req
$CRef ECHO_Req_T1
$Comment Tester sends SMT ECHO Request frame. $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Comment Causing TRT expired. $End_BehaviourLine
$BehaviourLine
$Line [4]START TRT $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Comment Causing TRT expired twice and Token never has been seen. $End_BehaviourLin
$BehaviourLine
$Line [6]START TRT, TRTclaim, T_Max $End_BehaviourLine
$BehaviourLine
$Line [7]?CLAIM
$CRef CLAIM_R1
$Comment IUT initiates Claim Token process. $End_BehaviourLine
$BehaviourLine
$Line [8]!CLAIM START TRT
$CRef CLAIM_T1
$Comment Tester repeats Claim frame. $End_BehaviourLine
$BehaviourLine
$Line [9]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment IUT issues Token. $End_BehaviourLine
$BehaviourLine
$Line [9]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [9]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?OTHERWISE
$Verdict F $End_BehaviourLine
```

```
$BehaviourLine
$Line [7]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/MONITORING/TVXtesting
$TestCaseId TVXtesting
$TestPurpose Testing MAC Monitoring function 2: TVX Monitoring
$BehaviourDescription
$TreeHeader TVXtesting
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection(test group) $End_BehaviourLine
$BehaviourLine
$Line [2]+TVX_SUBTREE1
$Comment Subtree 1 $End_BehaviourLine
$BehaviourLine
$Line [3]+FDDI/INITCONNECT
$Comment Initializing Connection(test group) $End_BehaviourLine
$BehaviourLine
$Line [4]+TVX_SUBTREE2
$Comment Subtree 2 $End_BehaviourLine
$TreeHeader TVX_SUBTREE1
$BehaviourLine
$Line [1]!DATA_OverMaxLen
$CRef DATA_OverMaxLen_T1
$Comment Tester sends tester frame(DA=IUT Address; Frame symbol time>TVX), to cause
$BehaviourLine
$Line [2]START TRT, TRTclaim, T_Max $End_BehaviourLine
$BehaviourLine
$Line [3]?CLAIM
$CRef CLAIM_R1
$Comment IUT initiates Claim Token process. $End_BehaviourLine
$BehaviourLine
$Line [4]!CLAIM START TRT
$CRef CLAIM_T1
$Comment Tester repeats the Claim frame sent by IUT. $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict (P)
$Comment IUT issues Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [3]?OTHERWISE
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Verdict (F) $End_BehaviourLine
$TreeHeader TVX_SUBTREE2
$BehaviourLine
$Line [1]?TIMEOUT TVX(a:=1)
$Label E
$Comment Holds Token and sends Idle symbols until TVX expires. $End_BehaviourLine
$BehaviourLine
$Line [1]!IDLE
$CRef IDLE_I $End_BehaviourLine
$BehaviourLine
$Line [2][a=0] $End_BehaviourLine
$BehaviourLine
$Line [3]GOTO E $End_BehaviourLine
$BehaviourLine
```

```
$Line [2]START TRT, TRTclaim, T_Max $End_BehaviourLine
$BehaviourLine
$Line [3]?CLAIM
$CRef CLAIM_R1
$Comment IUT starts Claim Token process. $End_BehaviourLine
$BehaviourLine
$Line [4]!CLAIM START TRT
$CRef CLAIM_T1 $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict R $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [3]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase $End_TestGrou
$TestGroup $TestGroupID FED
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/PHInvalidR10b
$TestCaseId PHInvalidR10b
$TestPurpose Testing MAC Frame Error Detection 1: R(10b) -- a transition at MAC rece
$BehaviourDescription
$TreeHeader PHInvalidR10b
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T3
$Comment Tester sends the frame(DA=IUT Address; A Invalid Symbol in PA) $End_Behavic
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R1
$Comment Idles returned $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/PHInvalidR20b
$TestCaseId PHInvalidR20b
```

```
$TestPurpose Testing MAC Frame Error Detection 2: R(20b) -- a transition at MAC rece
$BehaviourDescription
$TreeHeader PHInvalidR20b
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T4
$Comment Tester sends the frame(DA=IUT Address; a Invalid Symbol after J in SD) $Enc
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R1
$Comment PA returned, $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/PHInvalidR30b
$TestCaseId PHInvalidR30b
$TestPurpose Testing MAC Frame Error Detection 3: R(30b) -- a transition at MAC rece
$BehaviourDescription
$TreeHeader PHInvalidR30b
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T5
$Comment Tester sends the frame(DA=IUT Address; a Invalid Symbol in INFO) $End_Beha
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R2
$Comment PA,SD,FC,DA,SA and Idles returned, $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
```

```
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/PHInvalidR40b
$TestCaseId PHInvalidR40b
$TestPurpose Testing MAC Frame Error Detection 4: R(40b) -- a transition at MAC rece
$BehaviourDescription
$TreeHeader PHInvalidR40b
$BehaviourLine
$Line [1]+FDDI/INITCONNECT $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T6
$Comment Tester sends the frame(DA=IUT Address; a Invalid Symbol in FS) $End_Behavic
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R4
$Comment PA, SD, FC, DA, SA, INFO, FCS, ED and FS returned(E=S, A=S, C=S) $End_Beha\
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/DetectSD
$TestCaseId DetectSD
$TestPurpose Testing MAC Frame Error Detection 5: Detecting SD
$BehaviourDescription
$TreeHeader DetectSD
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T7_0
$Comment Tester sends the frame(DA=IUT Address; an Idle Symbol after J) $End_Behavic
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment Followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R1
$Comment PA and Idles returned. $End_BehaviourLine
$BehaviourLine
```

```
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict (P)
$Comment Followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [5]!DATA START TRT
$CRef DATA_T7_1
$Comment Tester sends the frame(DA=IUT Address; other symbol(not K symbol) before K
$BehaviourLine
$Line [6]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [7]?DATA_Strip
$CRef DATA_Strip_R1
$Comment PA and Idle returned, $End_BehaviourLine
$BehaviourLine
$Line [8]?TOKEN
$CRef TOKEN_R1
$Verdict R
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [8]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [8]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [7]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/DetectFC
$TestCaseId DetectFC
$TestPurpose Testing MAC Frame Error Detection 6: Detecting FC
$BehaviourDescription
$TreeHeader DetectFC
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T8
$Comment Tester sends the frame(DA=IUT Address; a symbol sequence(not Idle(s) or nn)
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R5
$Comment PA, SD and Idles returned, $End_BehaviourLine
```

```
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/DetectFrameBody1
$TestCaseId DetectFrameBody1
$TestPurpose Testing MAC Frame Error DEtection 7: Detecting Frame Body 1
$BehaviourDescription
$TreeHeader DetectFrameBody1
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T9
$Comment Tester sends the frame(DA=IUT Address; an Idle symbol in INFO), $End_Behavi
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R2
$Comment PA, SD, FC, DA, SA and Idles returned, $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/DetectFrameBody2
$TestCaseId DetectFrameBody2
$TestPurpose Testing MAC Frame Error Detection 8: Detecting Frame Body 2
$BehaviourDescription
$TreeHeader DetectFrameBody2
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
```

```
$CRef DATA_T10
$Comment Tester sends the frame(DA=IUT Address; a symbol(not Idle or data) in INFO),
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R2
$Comment PA, SD, FC, DA, SA and Idles returned, $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/DetectInvalidLength
$TestCaseId DetectInvalidLength
$TestPurpose Testing Frame Error Detection 9: Detecting Invalid Data Length Frame
$BehaviourDescription
$TreeHeader DetectInvalidLength
$BehaviourLine
$Line [1]+FDDI/INITCONNECT $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA_InvLen
$CRef DATA_InvLen_T1
$Comment Tester sends the frame (DA=IUT Address; a Invalid Data Length Frame), $End_
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA_Strip
$CRef DATA_Strip_R3
$Comment PA, SD, FC, DA, SA, INFO, FCS, ED and FS returned(A=S, E=S, C=R), $End_Beha
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [5]?CTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/FED/DetectFCS
```

```
$TestCaseId DetectFCS
$TestPurpose Testing Frame Error Detection 10: Detecting FCS Errors
$BehaviourDescription
$TreeHeader DetectFCS
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Intializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!DATA
$CRef DATA_T11_0
$Comment Tester sends the frame(DA=IUT Address; FCS errors), $End_BehaviourLine
$BehaviourLine
$Line [3]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [4]?DATA
$CRef DATA_R11_0
$Comment IUT repeats Tester's frame(A=S,E=S,C=R), $End_BehaviourLine
$BehaviourLine
$Line [5]?TOKEN
$CRef TOKEN_R1
$Comment Token returned. $End_BehaviourLine
$BehaviourLine
$Line [6]!DATA
$CRef DATA_T11_1
$Comment Tester sends the frame(DA<>IUT Address; FCS error), $End_BehaviourLine
$BehaviourLine
$Line [7]!TOKEN
$CRef TOKEN_T1
$Comment followed by Token. $End_BehaviourLine
$BehaviourLine
$Line [8]?DATA
$CRef DATA_R11_1
$Comment IUT repeats Tester's frame(A=R, E=S, C=R) $End_BehaviourLine
$BehaviourLine
$Line [9]?TOKEN
$CRef TOKEN_R1
$Verdict P
$Comment IUT returns Token. $End_BehaviourLine
$BehaviourLine
$Line [9]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [9]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [8]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [8]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [5]?TIMEOUT TRT
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [4]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase $End_TestGrou
$TestGroup $TestGroupID TED
$Begin_TestCase
```

```
$TestCaseRef FDDI/MAC/TED/PHInvalidR50b
$TestCaseId PHInvalidR50b
$TestPurpose Testing MAC Token Error Detection 1: R(50b) -- a transition at MAC rece
$BehaviourDescription
$TreeHeader PHInvalidR50b
$BehaviourLine
$Line [1]+FDDI/INITCONNECT
$Comment Initializing Connection $End_BehaviourLine
$BehaviourLine
$Line [2]!TOKEN
$CRef TOKEN_T2
$Comment Tester sends the Token (a Invalid Symbol in ED), $End_BehaviourLine
$BehaviourLine
$Line [3]?TOKEN_Strip
$CRef TOKEN_Strip_R1
$Verdict P
$Comment PA,SD,FC and Idles returned. $End_BehaviourLine
$BehaviourLine
$Line [3]?OTHERWISE
$Verdict F $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/TED/DetectTokenED1
$TestCaseId DetectTokenED1
$TestPurpose Testing 1 for ED of Token
$BehaviourDescription
$TreeHeader DetectTokenED1
$BehaviourLine
$Line [1]+FDDI/INITCONNECT $End_BehaviourLine
$BehaviourLine
$Line [2]!TOKEN
$CRef TOKEN_T3
$Comment Tester sends Token(an Idle Symbol in ED) $End_BehaviourLine
$BehaviourLine
$Line [3]?TOKEN_Strip
$CRef TOKEN_Strip_R1
$Verdict (P)
$Comment PA, SD, FC and Idles returned. $End_BehaviourLine
$BehaviourLine
$Line [3]?OTHERWISE
$Verdict (F) $End_BehaviourLine
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Verdict (F) $End_BehaviourLine $End_BehaviourDescription $End_TestCase
$Begin_TestCase
$TestCaseRef FDDI/MAC/TED/DetectTokenED2
$TestCaseId DetectTokenED2
$TestPurpose Testing 2 for the ED of Token
$BehaviourDescription
$TreeHeader DetectTokenED2
$BehaviourLine
$Line [1]+FDDI/INITCONNECT $End_BehaviourLine
$BehaviourLine
$Line [2]!TOKEN
$CRef TOKEN_T4
$Comment Tester sends the Token(any symbol(not Idles or T) in ED) $End_BehaviourLine
$BehaviourLine
$Line [3]?TOKEN_Strip
$CRef TOKEN_Strip_R1
$Verdict P
$Comment PA, SD, FC and Idles returned $End_BehaviourLine
$BehaviourLine
$Line [3]?OTHERWISE
$Verdict F $End_BehaviourLine
```

```
$BehaviourLine
$Line [3]?TIMEOUT TRT
$Verdict F $End_BehaviourLine $End_BehaviourDescription $End_TestCase $End_TestGrour
$End_TestCases
$TestStepLibrary
$Begin_TestStep
$TestStepRef FDDI/INITCONNECT
$TestStepId INITCONNECT
$Objective to complete ring initialization and form a Token path for test cases
$BehaviourDescription
$TreeHeader INITCONNECT
$BehaviourLine
$Line [1] (a:=0)
$End_BehaviourLine $End_BehaviourDescription
$End_TestStep
$End_TestStepLibrary
$End_DynamicPart
$ConstraintsPart
$PDUConstraints
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T1
$FVI
   $PDU_VALdcl $FIELDid PA
               $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
               $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
               $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
               $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
               $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
               $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
               $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_R1
$FVI
   $PDU_VALdcl $FIELDid PA
               $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
               $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
               $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Rev
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T2
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_R2
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
              $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
              $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
              $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
              $CONSvalue M_FCS_Rev
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
              $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
              $CONSvalue '010100101101011'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T3
$FVI
   $PDU_VALdcl $FIELDid PA
              $CONSvalue 'FFC1FFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
              $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
              $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
              $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
              $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
              $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
              $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
              $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
              $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T4
$FVI
   $PDU_VALdcl $FIELDid PA
              $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
              $CONSvalue '1100000000'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
              $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
                $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T5
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F029552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T6
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100000001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T7_0
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100011111'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T7_1
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1111110001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T8
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '0000000100'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T9
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
               $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
               $CONSvalue 'F7E9552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
               $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
               $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
               $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T10
$FVI
   $PDU_VALdcl $FIELDid PA
               $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
               $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
               $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
               $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
               $CONSvalue 'F569552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
               $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
               $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
               $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T11_0
$FVI
   $PDU_VALdcl $FIELDid PA
               $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
               $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
               $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
                $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
                $CONSvalue 'F7BDEF7BDE'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_R11_0
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
                $CONSvalue 'F7BDEF7BDE'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue '010110101101010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_T11_1
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
```

```
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F269552DCF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue 'F7BDEF7BDE'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA
$CONSid DATA_R11_1
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F269552DCF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue 'F7BDEF7BDE'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010110101001010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_InvLen
$CONSid DATA_InvLen_T1
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue P_16
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
```

```
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_OverMaxLen
$CONSid DATA_OverMaxLen_T1
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid TOKEN
$CONSid TOKEN_T1
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '10000000'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
```

```
                $CONSvalue '1010110101'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid TOKEN
$CONSid TOKEN_R1
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '10000000'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '1010110101'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid TOKEN
$CONSid TOKEN_T2
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '10000000'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '0000010101'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid TOKEN
$CONSid TOKEN_T3
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '10000000'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '1111110101'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid TOKEN
$CONSid TOKEN_T4
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
```

```
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '10000000'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '1111010101'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid CLAIM
$CONSid CLAIM_T1
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000011'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue T_Req_IUT
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Tramt
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid CLAIM
$CONSid CLAIM_R1
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000011'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue T_Req_IUT
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
```

```
                $CONSvalue M_FCS_Rev
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid CLAIM
$CONSid CLAIM_T2
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000011'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue T_Bid_Max
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Tramt
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid CLAIM
$CONSid CLAIM_R2
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000011'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue T_Bid_Max
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
```

```
               $CONSvalue M_FCS_Rev
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
               $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
               $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid CLAIM
$CONSid CLAIM_T3
$FVI
   $PDU_VALdcl $FIELDid PA
               $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
               $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
               $CONSvalue '11000011'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
               $CONSvalue T_Bid_Min
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
               $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
               $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
               $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid CLAIM
$CONSid CLAIM_T4
$FVI
   $PDU_VALdcl $FIELDid PA
               $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
               $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
               $CONSvalue '11000011'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
               $CONSvalue T_Req_Tester
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
```

```
                $CONSvalue M_FCS_Tramt
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid CLAIM
$CONSid CLAIM_R4
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000011'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue T_Req_Tester
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Rev
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid BEACON
$CONSid BEACON_T1
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000010'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue '000000000000'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue '00000000'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
```

```
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid BEACON
$CONSid BEACON_R1
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000010'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
            $CONSvalue '000000000000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue '00000000'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Rev
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid BEACON
$CONSid BEACON_T2
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue '11000010'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
            $CONSvalue '000000000000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue '00000000'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
```

```
                $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid BEACON
$CONSid BEACON_R2
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000010'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
                $CONSvalue '000000000000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
                $CONSvalue '00000000'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Rev
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid ECHO_Req
$CONSid ECHO_Req_T1
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue '81'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Frame_Class
                $CONSvalue '04'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Frame_Type
```

```
              $CONSvalue '02'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Version_ID
              $CONSvalue '0000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Transaction_ID
              $CONSvalue '00000000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Station_ID
              $CONSvalue '0000000000000001'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Pad
              $CONSvalue '0000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid InfoField_Length
              $CONSvalue '116A'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Parameter_Type
              $CONSvalue '0011'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Parameter_Length
              $CONSvalue '1168'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Echo_data
              $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
              $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
              $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
              $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid ECHO_Resp
$CONSid ECHO_Resp_R1
$FVI
   $PDU_VALdcl $FIELDid PA
              $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
              $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
              $CONSvalue '81'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
              $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
              $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Frame_Class
              $CONSvalue '04'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Frame_Type
              $CONSvalue '03'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Version_ID
              $CONSvalue '0000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Transaction_ID
```

```
                $CONSvalue '00000000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Station_ID
               $CONSvalue '0000000000000001'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Pad
               $CONSvalue '0000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid InfoField_Length
               $CONSvalue '116A'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Parameter_Type
               $CONSvalue '0011'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Parameter_Length
               $CONSvalue '1168'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Echo_data
               $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
               $CONSvalue M_FCS_Rev
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
               $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
               $CONSvalue '010100101001010'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid ECHO_Resp
$CONSid ECHO_Resp_T1
$FVI
   $PDU_VALdcl $FIELDid PA
               $CONSvalue 'FFFFFFFFFFFFFFFFFFFF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
               $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
               $CONSvalue '81'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
               $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
               $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Frame_Class
               $CONSvalue '04'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Frame_Type
               $CONSvalue '03'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Version_ID
               $CONSvalue '0000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Transaction_ID
               $CONSvalue '00000000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Station_ID
               $CONSvalue '0000000000000001'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Pad
```

```
            $CONSvalue '0000'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid InfoField_Length
            $CONSvalue '116A'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Parameter_Type
            $CONSvalue '0011'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Parameter_Length
            $CONSvalue '1168'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid Echo_data
            $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue M_FCS_Tramt
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue '010100101101011'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_Strip
$CONSid DATA_Strip_R1
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
            $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
            $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
            $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
            $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
            $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
            $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
            $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
            $CONSvalue -
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_Strip
$CONSid DATA_Strip_R2
$FVI
   $PDU_VALdcl $FIELDid PA
            $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
```

```
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue -
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue -
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue -
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue -
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_Strip
$CONSid DATA_Strip_R3
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F269552DCF'H
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Rev
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue '010110101101010'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_Strip
$CONSid DATA_Strip_R4
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
```

```
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
                $CONSvalue IUT_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue Tester_Address
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
                $CONSvalue 'F269552DCF'H
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
                $CONSvalue M_FCS_Rev
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue '10101'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue '010110101101011'B
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_Strip
$CONSid DATA_Strip_R5
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FC
                $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid DA
                $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SA
                $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid INFO
                $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FCS
                $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid ED
                $CONSvalue -
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid FS
                $CONSvalue -
   $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid DATA_Strip
$CONSid DATA_Strip_R6
$FVI
   $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
   $End_PDU_VALdcl
   $PDU_VALdcl $FIELDid SD
```

```
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '11000001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid DA
                $CONSvalue Tester_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SA
                $CONSvalue IUT_Address
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid INFO
                $CONSvalue -
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FCS
                $CONSvalue -
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue -
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FS
                $CONSvalue -
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid TOKEN_Strip
$CONSid TOKEN_Strip_R1
$FVI
    $PDU_VALdcl $FIELDid PA
                $CONSvalue ?
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid SD
                $CONSvalue '1100010001'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid FC
                $CONSvalue '10000000'B
    $End_PDU_VALdcl
    $PDU_VALdcl $FIELDid ED
                $CONSvalue -
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$Begin_TTCN_PDUconstraint
$PDUid IDLE
$CONSid IDLE_I
$FVI
    $PDU_VALdcl $FIELDid SingleIdle
                $CONSvalue '11111'B
    $End_PDU_VALdcl
$End_FVI
$End_TTCN_PDUconstraint
$End_PDUconstraints
$End_ConstraintsPart $End_Suite
```

## Appendix C

## FDDI MAC Conformance Test

## C Language Test Generated by the TTCN Translator

```
/* NIST TTCN Translator Version 1.0  4/89 */
#include <stdio.h>
#include "suite.h"
int             _level, _time, _bufferlen;
unsigned char  *_buffer;
_verdict        Result;
char           *_timername;
char            R[10];
tsparameter()
{
  char            tempstr[80];
  printf("Enter test suite parameters.\n\n");
  printf("The address of Station Under Test.  Enter an integer.\n");
  scanf("%d", &IUT_Address);
  printf("The address of Tester. Get this value from Tester.  Enter an integer.\n");
  scanf("%d", &Tester_Address);
  printf("The operative Target Token Rotation Time.  Enter an integer.\n");
  scanf("%d", &TTRT);
  printf("Tester's Requested TTRT.  Enter an integer.\n");
  scanf("%d", &T_Req_Tester);
  printf("IUT's Requested TTRT.  Enter an integer.\n");
  scanf("%d", &T_Req_IUT);
  printf("Highest Bidding Value of Tester in Claim Token Process.  Enter an integer.\n");
  scanf("%d", &T_Bid_Max);
  printf("Lowest Bidding Value of Tester in Claim Token process.  Enter an integer.\n");
  scanf("%d", &T_Bid_Min);
  printf("Enter an hexadecimal string. e.g.12CF\n");
  scanf("%x", &M_FCS_Rev);
  printf("Enter an hexadecimal string. e.g.12CF\n");
  scanf("%x", &M_FCS_Tramt);
}
main()
{
  tsparameter();
  initialize();
  Output_trace("\n\nStarting test case FrameTransmit ");
  flush_queue();
  FrameTransmit();
  DL_POSTAMBLE();
  print_verdict("End test case FrameTransmit ");
  Output_trace("\n\nStarting test case FrameRepeat ");
  flush_queue();
  FrameRepeat();
  DL_POSTAMBLE();
  print_verdict("End test case FrameRepeat ");
  Output_trace("\n\nStarting test case FrameReceive ");
  flush_queue();
  FrameReceive();
  DL_POSTAMBLE();
  print_verdict("End test case FrameReceive ");
  Output_trace("\n\nStarting test case FrameStrip ");
  flush_queue();
  FrameStrip();
  DL_POSTAMBLE();
  print_verdict("End test case FrameStrip ");
  Output_trace("\n\nStarting test case TesterWinClaim ");
  flush_queue();
  TesterWinClaim();
  DL_POSTAMBLE();
  print_verdict("End test case TesterWinClaim ");
  Output_trace("\n\nStarting test case IUTWinClaim ");
  flush_queue();
  IUTWinClaim();
  DL_POSTAMBLE();
  print_verdict("End test case IUTWinClaim ");
  Output_trace("\n\nStarting test case TesterGotOwnBeacon ");
```

```
flush_queue();
TesterGotOwnBeacon();
DL_POSTAMBLE();
print_verdict("End test case TesterGotOwnBeacon ");
Output_trace("\n\nStarting test case IUTGotOwnBeacon ");
flush_queue();
IUTGotOwnBeacon();
DL_POSTAMBLE();
print_verdict("End test case IUTGotOwnBeacon ");
Output_trace("\n\nStarting test case EarlyToken ");
flush_queue();
EarlyToken();
DL_POSTAMBLE();
print_verdict("End test case EarlyToken ");
Output_trace("\n\nStarting test case LateToken ");
flush_queue();
LateToken();
DL_POSTAMBLE();
print_verdict("End test case LateToken ");
Output_trace("\n\nStarting test case TRTtesting ");
flush_queue();
TRTtesting();
DL_POSTAMBLE();
print_verdict("End test case TRTtesting ");
Output_trace("\n\nStarting test case TVXtesting ");
flush_queue();
TVXtesting();
DL_POSTAMBLE();
print_verdict("End test case TVXtesting ");
Output_trace("\n\nStarting test case PHInvalidR10b ");
flush_queue();
PHInvalidR10b();
DL_POSTAMBLE();
print_verdict("End test case PHInvalidR10b ");
Output_trace("\n\nStarting test case PHInvalidR20b ");
flush_queue();
PHInvalidR20b();
DL_POSTAMBLE();
print_verdict("End test case PHInvalidR20b ");
Output_trace("\n\nStarting test case PHInvalidR30b ");
flush_queue();
PHInvalidR30b();
DL_POSTAMBLE();
print_verdict("End test case PHInvalidR30b ");
Output_trace("\n\nStarting test case PHInvalidR40b ");
flush_queue();
PHInvalidR40b();
DL_POSTAMBLE();
print_verdict("End test case PHInvalidR40b ");
Output_trace("\n\nStarting test case DetectSD ");
flush_queue();
DetectSD();
DL_POSTAMBLE();
print_verdict("End test case DetectSD ");
Output_trace("\n\nStarting test case DetectFC ");
flush_queue();
DetectFC();
DL_POSTAMBLE();
print_verdict("End test case DetectFC ");
Output_trace("\n\nStarting test case DetectFrameBody1 ");
flush_queue();
DetectFrameBody1();
DL_POSTAMBLE();
print_verdict("End test case DetectFrameBody1 ");
Output_trace("\n\nStarting test case DetectFrameBody2 ");
flush_queue();
```

```
  DetectFrameBody2();
  DL_POSTAMBLE();
  print_verdict("End test case DetectFrameBody2 ");
  Output_trace("\n\nStarting test case DetectInvalidLength ");
  flush_queue();
  DetectInvalidLength();
  DL_POSTAMBLE();
  print_verdict("End test case DetectInvalidLength ");
  Output_trace("\n\nStarting test case DetectFCS ");
  flush_queue();
  DetectFCS();
  DL_POSTAMBLE();
  print_verdict("End test case DetectFCS ");
  Output_trace("\n\nStarting test case PHInvalidR50b ");
  flush_queue();
  PHInvalidR50b();
  DL_POSTAMBLE();
  print_verdict("End test case PHInvalidR50b ");
  Output_trace("\n\nStarting test case DetectTokenED1 ");
  flush_queue();
  DetectTokenED1();
  DL_POSTAMBLE();
  print_verdict("End test case DetectTokenED1 ");
  Output_trace("\n\nStarting test case DetectTokenED2 ");
  flush_queue();
  DetectTokenED2();
  DL_POSTAMBLE();
  print_verdict("End test case DetectTokenED2 ");
  exit();
}
```

```
/* NIST TTCN Translator Version 1.0  4/89 */
#include <stdio.h>
#include "sysdef.h"
typedef unsigned char BYTE;
typedef struct {
  union {
    long            v;
    char           *ptr;
  }                value;
  char             tag;
}                CONSTARG;
extern int       _level, _time, _bufferlen;
extern unsigned char *_buffer;
extern char     *_timername;
extern _verdict Result;
extern void      Implicit_send();
extern char      R[];
extern long      IUT_Address;     /* The address of Station Under Test.  */
extern long      Tester_Address; /* The address of Tester. Get this value from
                                   * Tester.  */
extern long      TTRT;            /* The operative Target Token Rotation Time.  */
extern long      T_Req_Tester;    /* Tester's Requested TTRT.  */
extern long      T_Req_IUT;       /* IUT's Requested TTRT.  */
extern long      T_Bid_Max;       /* Highest Bidding Value of Tester in Claim
                                   * Token Process.  */
extern long      T_Bid_Min;       /* Lowest Bidding Value of Tester in Claim
                                   * Token process.  */
extern long      M_FCS_Rev;
extern long      M_FCS_Tramt;
#define T_Max 167772              /* The maximum value (default value) of TRT.  */
#define T_Min 4000                /* The minimum value (default value) of TRT.  */
#define S0 0x1e                   /* Data Symbol 0  */
#define S1 0x09                   /* Data Symbol 1  */
#define S2 0x14                   /* Data Symbol 2  */
#define S3 0x15                   /* Data Symbol 3  */
#define S4 0x0a                   /* Data Symbol 4  */
#define S5 0x0b                   /* Data Symbol 5  */
#define S6 0x0e                   /* Data Symbol 6  */
#define S7 0x0f                   /* Data Symbol 7  */
#define S8 0x12                   /* Data Symbol 8  */
#define S9 0x13                   /* Data Symbol 9  */
#define sA 0x16                   /* Data Symbol A  */
#define SB 0x17                   /* Data Symbol B  */
#define SC 0x1a                   /* Data Symbol C  */
#define sD 0x1b                   /* Data Symbol D  */
#define SE 0x1c                   /* Data Symbol E  */
#define SF 0x1d                   /* Data Symbol F  */
#define SQ 0x00                   /* Line State Symbol Q  */
#define SI 0x1f                   /* Line State Symbol I  */
#define SH 0x04                   /* Line State Symbol H  */
#define SJ 0x18                   /* Starting Delimiter J  */
#define SK 0x11                   /* Starting Delimiter K  */
#define ST 0x15                   /* Ending Delimiter T  */
#define SR 0x0a                   /* Reset Symbol R  */
#define SS 0x0b                   /* Set Symbol S  */
#define S0S1S2S3S4S5S6S7 0xF269552DCF
#define SISISQSISISISISISISISISISISISI 0xFFC1FFFFFFFFFFFFFFF
#define S0SQS2S3S4S5S6S7 0xF029552DCF
#define S0SIS2S3S4S5S6S7 0xF7E9552DCF
#define S0STS2S3S4S5S6S7 0xF569552DCF
#define SQSH 0x004
#define SJSK 0x311
#define SJSQ 0x300
#define SJSI 0x31f
#define SISK 0x3f1
#define SRSRSR 0x294a
```

```
#define SRSSSS 0x296b
#define SRSQSR 0x280a
#define S0S0S0S0S0S0S0S0 0xF7BDEF7BDE
#define SSSSSR 0x2d6a
#define SSSRSR 0x2d4a
#define SQST 0x015
#define STST 0x2b5
#define SIST 0x3f5
#define S0ST 0x3d5
#define SSSSSS 0x2d6b
#define P_16 0xFFFFFFFFFFFFFFFFFFFF     /* The Preamble field in frame
                                          * transmitted by Tester.  */
typedef struct {                    /* General MAC Data Frame  */
  long           PA;                /* Preamble  */
  long           SD;                /* Starting Delimiter  */
  long           FC;                /* Frame Control  */
  long           DA;                /* Destination Address  */
  long           SA;                /* Source Address  */
  long           INFO;              /* INFOrmation  */
  long           FCS;               /* Frame Check Sequence  */
  long           ED;                /* End Delimiter  */
  long           FS;                /* Frame Status  */
  char           PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, INFO_tag, FCS_tag, ED_tag, FS_tag;
}               pdu_DATA;
extern          Send_PDU_DATA();
extern int      Is_PDU_DATA();
typedef struct {                    /* MAC Nonrestricted Token  */
  long           PA;                /* Preamble  */
  long           SD;                /* Starting Delimiter  */
  long           FC;                /* Frame Control  */
  long           ED;                /* End Delimiter  */
  char           PA_tag, SD_tag, FC_tag, ED_tag;
}               pdu_TOKEN;
extern          Send_PDU_TOKEN();
extern int      Is_PDU_TOKEN();
typedef struct {                    /* The MAC frame with the Invalid Data Length  */
  long           PA;                /* Preamble  */
  long           SD;                /* Starting Delimiter  */
  long           FC;                /* Frame Control  */
  long           DA;                /* Destination Address  */
  long           SA;                /* Source Address  */
  long           INFO;              /* INFOrmation field with the Invalid Data
                                     * Length.  */
  long           FCS;               /* Frame Check Sequence  */
  long           ED;                /* End Delimiter  */
  long           FS;                /* Frame Status  */
  char           PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, INFO_tag, FCS_tag, ED_tag, FS_tag;
}               pdu_DATA_InvLen;
extern          Send_PDU_DATA_InvLen();
extern int      Is_PDU_DATA_InvLen();
typedef struct {                    /* The MAC frame whose symbol times is
                                     * greater than TVX  */
  long           PA;                /* Preamble  */
  long           SD;                /* Starting Delimiter  */
  long           FC;                /* Frame Control  */
  long           DA;                /* Destination Address  */
  long           SA;                /* Source Address  */
  long           INFO;              /* INFOrmation field. This frame's symbol
                                     * time is greater than TVX.  */
  long           FCS;               /* Frame Check Sequence  */
  long           ED;                /* End Delimiter  */
  long           FS;                /* Frame Status  */
  char           PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, INFO_tag, FCS_tag, ED_tag, FS_tag;
}               pdu_DATA_OverMaxLen;
extern          Send_PDU_DATA_OverMaxLen();
extern int      Is_PDU_DATA_OverMaxLen();
```

```
typedef struct {                /* MAC Claim Frame  */
  long           PA;            /* Preamble  */
  long           SD;            /* Starting Delimiter  */
  long           FC;            /* Frame Control  */
  long           DA;            /* Destination Address  */
  long           SA;            /* Source Address  */
  long           INFO;          /* INFOrmation  */
  long           FCS;           /* Frame Check Sequence  */
  long           ED;            /* End Delimiter  */
  long           FS;            /* Frame Status  */
  char           PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, INFO_tag, FCS_tag, ED_tag, FS_tag;
}               pdu_CLAIM;
extern          Send_PDU_CLAIM();
extern int      Is_PDU_CLAIM();
typedef struct {                /* MAC Beacon Frame  */
  long           PA;            /* Preamble  */
  long           SD;            /* Starting Delimiter  */
  long           FC;            /* Frame Control  */
  long           DA;            /* Destination Address  */
  long           SA;            /* Source Address  */
  long           INFO;          /* INFOrmation  */
  long           FCS;           /* Frame Check Sequence  */
  long           ED;            /* End Delimiter  */
  long           FS;            /* Frame Status  */
  char           PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, INFO_tag, FCS_tag, ED_tag, FS_tag;
}               pdu_BEACON;
extern          Send_PDU_BEACON();
extern int      Is_PDU_BEACON();
typedef struct {                /* SMT ECHO request Frame  */
  long           PA;            /* Preamble  */
  long           SD;            /* Starting Delimiter  */
  long           FC;            /* Frame Control  */
  BYTE           DA[6];         /* Destination Address  */
  BYTE           SA[6];         /* Source Address  */
  long           Frame_Class;   /* To identify the function of the frame.  */
  long           Frame_Type;    /* To designate the type of the frame  */
  long           Version_ID;    /* The value will not change when upward
                                 * compatible changes are made to the SMT
                                 * frames.  */
  long           Transaction_ID;        /* To be used to pair SMT responses
                                         * with their requests.  */
  BYTE           Station_ID[8];/* The unique identifier for an FDDI station
                                 * (or concentrator).  */
  long           Pad;
  long           InfoField_Length;      /* The length of the SMT Information
                                         * field.  */
  long           Parameter_Type;
  long           Parameter_Length;      /* The length of the Echo_data.  */
  BYTE           Echo_data[1168];       /* SMT InfoField  */
  long           FCS;           /* Frame Check Sequence  */
  long           ED;            /* Ending Delimiter  */
  long           FS;            /* Frame Status  */
  char           PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, Frame_Class_tag, Frame_Type_tag, Version_ID_
}               pdu_ECHO_Req;
extern          Send_PDU_ECHO_Req();
extern int      Is_PDU_ECHO_Req();
typedef struct {                /* SMT ECHO Response Frame  */
  long           PA;            /* Preamble  */
  long           SD;            /* Starting Delimiter  */
  long           FC;            /* Frame Control  */
  BYTE           DA[6];         /* Destination Address  */
  BYTE           SA[6];         /* Source Address  */
  long           Frame_Class;   /* To identify the function of the frame.  */
  long           Frame_Type;    /* To designate the type of the frame.  */
  long           Version_ID;    /* The value will not change when upward
                                 * compatible changes are made to the SMT
```

```
                                * frames.  */
  long            Transaction_ID;       /* To be used to pair SMT responses
                                         * with their requests.  */
  BYTE            Station_ID[8];/* The unique identifier for an FDDI station
                                * (or concentrator).  */
  long            Pad;
  long            InfoField_Length;     /* The length of the SMT Information
                                         * field.  */
  long            Parameter_Type;
  long            Parameter_Length;     /* The length of the Echo_data.  */
  BYTE            Echo_data[1168];      /* SMT InfoField  */
  long            FCS;          /* Frame Check Sequence  */
  long            ED;           /* Ending Delimiter  */
  long            FS;           /* Frame Status  */
  char            PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, Frame_Class_tag, Frame_Type_tag, Version_ID_
}               pdu_ECHO_Resp;
extern          Send_PDU_ECHO_Resp();
extern int      Is_PDU_ECHO_Resp();
typedef struct {                /* MAC Data Frame Stripped  */
  long            PA;           /* Preamble  */
  long            SD;           /* Starting Delimiter  */
  long            FC;           /* Frame Control  */
  long            DA;           /* Destination Address  */
  long            SA;           /* Source Address  */
  long            INFO;         /* INFOrmation  */
  long            FCS;          /* Frame Check Sequence  */
  long            ED;           /* Ending Delimiter  */
  long            FS;           /* Frame Status  */
  char            PA_tag, SD_tag, FC_tag, DA_tag, SA_tag, INFO_tag, FCS_tag, ED_tag, FS_tag;
}               pdu_DATA_Strip;
extern          Send_PDU_DATA_Strip();
extern int      Is_PDU_DATA_Strip();
typedef struct {                /* MAC Nonrestricted Token Stripped  */
  long            PA;           /* Preamble  */
  long            SD;           /* Starting Delimiter  */
  long            FC;           /* Frame Control  */
  long            ED;           /* Ending Delimiter  */
  char            PA_tag, SD_tag, FC_tag, ED_tag;
}               pdu_TOKEN_Strip;
extern          Send_PDU_TOKEN_Strip();
extern int      Is_PDU_TOKEN_Strip();
typedef struct {
  long            SingleIdle;   /* A single Idle symbol is sent by Tester.  */
  char            SingleIdle_tag;
}               pdu_IDLE;
extern          Send_PDU_IDLE();
extern int      Is_PDU_IDLE();
extern pdu_DATA *DATA_T1();
extern pdu_DATA *DATA_R1();
extern pdu_DATA *DATA_T2();
extern pdu_DATA *DATA_R2();
extern pdu_DATA *DATA_T3();
extern pdu_DATA *DATA_T4();
extern pdu_DATA *DATA_T5();
extern pdu_DATA *DATA_T6();
extern pdu_DATA *DATA_T7_0();
extern pdu_DATA *DATA_T7_1();
extern pdu_DATA *DATA_T8();
extern pdu_DATA *DATA_T9();
extern pdu_DATA *DATA_T10();
extern pdu_DATA *DATA_T11_0();
extern pdu_DATA *DATA_R11_0();
extern pdu_DATA *DATA_T11_1();
extern pdu_DATA *DATA_R11_1();
extern pdu_DATA_InvLen *DATA_InvLen_T1();
extern pdu_DATA_OverMaxLen *DATA_OverMaxLen_T1();
extern pdu_TOKEN *TOKEN_T1();
extern pdu_TOKEN *TOKEN_R1();
extern pdu_TOKEN *TOKEN_T2();
extern pdu_TOKEN *TOKEN_T3();
extern pdu_TOKEN *TOKEN_T4();
extern pdu_CLAIM *CLAIM_T1();
extern pdu_CLAIM *CLAIM_R1();
extern pdu_CLAIM *CLAIM_T2();
extern pdu_CLAIM *CLAIM_R2();
extern pdu_CLAIM *CLAIM_T3();
extern pdu_CLAIM *CLAIM_T4();
extern pdu_CLAIM *CLAIM_R4();
extern pdu_BEACON *BEACON_T1();
extern pdu_BEACON *BEACON_R1();
extern pdu_BEACON *BEACON_T2();
extern pdu_BEACON *BEACON_R2();
extern pdu_ECHO_Req *ECHO_Req_T1();
extern pdu_ECHO_Resp *ECHO_Resp_R1();
extern pdu_ECHO_Resp *ECHO_Resp_T1();
extern pdu_DATA_Strip *DATA_Strip_R1();
extern pdu_DATA_Strip *DATA_Strip_R2();
extern pdu_DATA_Strip *DATA_Strip_R3();
extern pdu_DATA_Strip *DATA_Strip_R4();
extern pdu_DATA_Strip *DATA_Strip_R5();
extern pdu_DATA_Strip *DATA_Strip_R6();
extern pdu_TOKEN_Strip *TOKEN_Strip_R1();
extern pdu_IDLE *IDLE_I();
```

```
/* NIST TTCN Translator Version 1.0  4/89 */
#include "suite.h"
/*
 * *** Test Suite: FDDI  **** Standard Reference: FDDI MAC (X3T9.5/88-139);
 * PICS Reference: PIXIT Reference: How Used:
 * Test Methods: The Remote Test Method
 * Comments: Test Suite for FDDI MAC Conformance Verification
 */
long            IUT_Address;    /* The address of Station Under Test.  */
long            Tester_Address; /* The address of Tester. Get this value from
                                 * Tester.  */
long            TTRT;           /* The operative Target Token Rotation Time.  */
long            T_Req_Tester;   /* Tester's Requested TTRT.  */
long            T_Req_IUT;      /* IUT's Requested TTRT.  */
long            T_Bid_Max;      /* Highest Bidding Value of Tester in Claim
                                 * Token Process.  */
long            T_Bid_Min;      /* Lowest Bidding Value of Tester in Claim
                                 * Token process.  */
long            M_FCS_Rev;
long            M_FCS_Tramt;
long            a = 0;          /* A variable for the test suite.  */
pdu_DATA        DATA, *pDATA;
pdu_TOKEN       TOKEN, *pTOKEN;
pdu_DATA_InvLen DATA_InvLen, *pDATA_InvLen;
pdu_DATA_OverMaxLen DATA_OverMaxLen, *pDATA_OverMaxLen;
pdu_CLAIM       CLAIM, *pCLAIM;
pdu_BEACON      BEACON, *pBEACON;
pdu_ECHO_Req    ECHO_Req, *pECHO_Req;
pdu_ECHO_Resp   ECHO_Resp, *pECHO_Resp;
pdu_DATA_Strip  DATA_Strip, *pDATA_Strip;
pdu_TOKEN_Strip TOKEN_Strip, *pTOKEN_Strip;
pdu_IDLE        IDLE, *pIDLE;


/*
 * FrameTransmit -- Testing MAC Frame Transmission
 */
int
FrameTransmit()
{
  int               _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 762 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 765 "FDDI.MP" -- [2]! ECHO_Req */
      /* Tester's ECHO Request Frame(DA=IUT Address).  */
      pECHO_Req = ECHO_Req_T1();
      Send_PDU_ECHO_Req(pECHO_Req);
      _level += 1;
      while (TRUE) {
        /* line 769 "FDDI.MP" -- [3]! TOKEN */
        /* Tester releases Token to IUT.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 773 "FDDI.MP" -- [4]? ECHO_Resp */
          /* IUT sends ECHO Response back.  */
          pECHO_Resp = ECHO_Resp_R1();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_ECHO_Resp(pECHO_Resp, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 777 "FDDI.MP" -- [5]? TOKEN */
              /* Tester receives Token from IUT.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 783 "FDDI.MP" -- [5]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 786 "FDDI.MP" -- [5]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                   /* end of level [5] */
          }
          /* line 789 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 792 "FDDI.MP" -- [4]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                       /* end of level [4] */
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * FrameRepeat -- Testing MAC Frame Repeating
 */
int
FrameRepeat()
{
  int          _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 801 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 804 "FDDI.MP" -- [2]! DATA */
      /* Sending Tester Frame(DA<>IUT Address).  */
      pDATA = DATA_T1();
      Send_PDU_DATA(pDATA);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 808 "FDDI.MP" -- [3]? DATA */
        /* Tester gets the Tester frame repeated (A=R; C=R; E=R).  */
        pDATA = DATA_R1();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_DATA(pDATA, _buffer)) {
          _level += 1;
          SetfVerdict(PASS);
          return (TRUE);
        }
        /* line 814 "FDDI.MP" -- [3]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 817 "FDDI.MP" -- [3]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(INCONC);
          return (TRUE);
        }
      }                           /* end of level [3] */
    }                             /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                 /* end of level [1] */


/*
 * FrameReceive -- Testing MAC Frame Receiving
 */
int
FrameReceive()
{
  int            _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 825 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 828 "FDDI.MP" -- [2]! DATA */
      /* Tester sends the frame(DA=IUT Address).  */
      pDATA = DATA_T2();
      Send_PDU_DATA(pDATA);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 832 "FDDI.MP" -- [3]? DATA */
        /* Tester gets the frame (A=S; C=S; E=R) received by IUT.  */
        pDATA = DATA_R2();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_DATA(pDATA, _buffer)) {
          _level += 1;
          SetfVerdict(PASS);
          return (TRUE);
        }
        /* line 838 "FDDI.MP" -- [3]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 841 "FDDI.MP" -- [3]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                              /* end of level [3] */
    }                                /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                    /* end of level [1] */


/*
 * FrameStrip -- Testing MAC Frame Stripping
 */
int
FrameStrip()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 849 "FDDI.MP" -- [1]+  */
  /* Initializing Connection.  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 852 "FDDI.MP" -- [2]! ECHO_Req */
      /* Tester sends SMT ECHO Request frame,  */
      pECHO_Req = ECHO_Req_T1();
      Send_PDU_ECHO_Req(pECHO_Req);
      _level += 1;
      while (TRUE) {
        /* line 856 "FDDI.MP" -- [3]! TOKEN */
        /* and then issues Token.  */
```

```
pTOKEN = TOKEN_T1();
Send_PDU_TOKEN(pTOKEN);
_level += 1;
_timername = "TRT.";
while (TRUE) {
  /* line 860 "FDDI.MP" -- [4]? ECHO_Resp */
  /* IUT sends SMT ECHO Response frame,  */
  pECHO_Resp = ECHO_Resp_R1();
  _buffer = Receive_PDU(_timername);
  if (Is_PDU_ECHO_Resp(pECHO_Resp, _buffer)) {
    _level += 1;
    _timername = "TRT.";
    while (TRUE) {
      /* line 864 "FDDI.MP" -- [5]? TOKEN */
      /* and issues Token to Tester.  */
      pTOKEN = TOKEN_R1();
      _buffer = Receive_PDU(_timername);
      if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
        Start_timer("TRT.", (long) TTRT 1000);
        _level += 1;
        while (TRUE) {
          /* line 868 "FDDI.MP" -- [6]! ECHO_Resp */
          /*
           * Tester received ECHO Response and returns this frame to
           * IUT.
           */
          pECHO_Resp = ECHO_Resp_T1();
          Send_PDU_ECHO_Resp(pECHO_Resp);
          _level += 1;
          _timername = "TRT.";
          while (TRUE) {
            /* line 872 "FDDI.MP" -- [7]? DATA_Strip */
            /* IUT strips this frame.  */
            pDATA_Strip = DATA_Strip_R6();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
              _level += 1;
              SetfVerdict(PASS);
              return (TRUE);
            }
            /* line 878 "FDDI.MP" -- [7]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 881 "FDDI.MP" -- [7]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
          }            /* end of level [7] */
        }              /* end of level [6] */
      }
      /* line 884 "FDDI.MP" -- [5]? OTHERWISE */
      if (_buffer) {
        _level += 1;
        SetfVerdict(INCONC);
        return (TRUE);
      }
      /* line 887 "FDDI.MP" -- [5]? TIMEOUT */
      if (Timeout("TRT.")) {
        _level += 1;
        SetfVerdict(INCONC);
        return (TRUE);
```

```
              }
            }                     /* end of level [5] */
          }
          /* line 890 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(INCONC);
            return (TRUE);
          }
          /* line 893 "FDDI.MP" -- [4]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(INCONC);
            return (TRUE);
          }
        }                         /* end of level [4] */
      }                           /* end of level [3] */
    }                             /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                 /* end of level [1] */


/*
 * TesterWinClaim -- Testing MAC Claim Token Process 1: Tester wins Claim
 * Token
 */
int
TesterWinClaim()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 902 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
A:
    while (TRUE) {
      /* line 907 "FDDI.MP" -- [2]? TIMEOUT */
      /* Tester holds Token and sends Idle symbols until TVX expires.  */
      if (Timeout("TVX.")) {
        _level += 1;
        return (FALSE);
      }
      /* line 909 "FDDI.MP" -- [2]! IDLE */
      pIDLE = IDLE_I();
      Send_PDU_IDLE(pIDLE);
      _level += 1;
      while (TRUE) {
        /* line 912 "FDDI.MP" -- [3]   */
        if ((a == 0)) {
          _level += 1;
          /* line 914 "FDDI.MP" -- [4]? GOTO */
          goto A;
        }
        /* line 916 "FDDI.MP" -- [3]   */
        Start_timer("TRT.TRTclaim", (long) T_Max 1000);
```

```
_level += 1;
_timername = "TRT.";
while (TRUE) {
  /* line 918 "FDDI.MP" -- [4]? CLAIM */
  /* IUT issues Claim frame.  */
  pCLAIM = CLAIM_R1();
  _buffer = Receive_PDU(_timername);
  if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
    Start_timer("TRT.TRTclaim", (long) T_Max 1000);
    _level += 1;
    while (TRUE) {
      /* line 922 "FDDI.MP" -- [5]! CLAIM */
      /* Tester issues its Claim frame with highest T_Bid.  */
      pCLAIM = CLAIM_T2();
      Send_PDU_CLAIM(pCLAIM);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 926 "FDDI.MP" -- [6]? CLAIM */
        /* IUT repeats Tester Claim Frame.  */
        pCLAIM = CLAIM_R2();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
          Start_timer("TRT.", (long) TTRT 1000);
          _level += 1;
          while (TRUE) {
            /* line 930 "FDDI.MP" -- [7]! TOKEN */
            /* Tester issues Token.  */
            pTOKEN = TOKEN_T1();
            Send_PDU_TOKEN(pTOKEN);
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 934 "FDDI.MP" -- [8]? TOKEN */
              /* IUT repeats Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 940 "FDDI.MP" -- [8]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 943 "FDDI.MP" -- [8]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }           /* end of level [8] */
          }             /* end of level [7] */
        }
        /* line 946 "FDDI.MP" -- [6]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 949 "FDDI.MP" -- [6]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
```

```
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                   /* end of level [6] */
          }                     /* end of level [5] */
        }
        /* line 952 "FDDI.MP" -- [4]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 955 "FDDI.MP" -- [4]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                         /* end of level [4] */
    }                           /* end of level [3] */
  }                             /* end of level [2] */
 }
 _level = _lastlevel;
 return (FALSE);
}                               /* end of level [1] */


/*
 * IUTWinClaim -- Testing MAC Claim Token process 2: IUT wins Claim Token
 */
int
IUTWinClaim()
{
 int             _lastlevel;
 _time = 0;
 _level = 1;
 strcpy(R, "NONE");
 Result = NONE;
 _timername = (char *) 0;
 _bufferlen = -1;
 /* line 963 "FDDI.MP" -- [1]+  */
 _lastlevel = _level;
 if (INITCONNECT())
   return (TRUE);
 else if (_level > _lastlevel) {
   _level += 1;
B:
   while (TRUE) {
     /* line 966 "FDDI.MP" -- [2]? TIMEOUT */
     if (Timeout("TVX.")) {
       _level += 1;
       return (FALSE);
     }
     /* line 968 "FDDI.MP" -- [2]! IDLE */
     /* Tester holds Token and sends Idle symbols until TVX expires.  */
     pIDLE = IDLE_I();
     Send_PDU_IDLE(pIDLE);
     _level += 1;
     while (TRUE) {
       /* line 972 "FDDI.MP" -- [3]   */
       if ((a == 0)) {
         _level += 1;
         /* line 974 "FDDI.MP" -- [4]? GOTO */
         goto B;
       }
       /* line 976 "FDDI.MP" -- [3]   */
```

```
Start_timer("TRT.TRTclaim", (long) T_Max 1000);
_level += 1;
_timername = "TRT.";
while (TRUE) {
  /* line 978 "FDDI.MP" -- [4]? CLAIM */
  /* IUT issues Claim frame.  */
  pCLAIM = CLAIM_R1();
  _buffer = Receive_PDU(_timername);
  if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
    Start_timer("TRT.TRTclaim", (long) T_Max 1000);
    _level += 1;
    while (TRUE) {
      /* line 982 "FDDI.MP" -- [5]! CLAIM */
      /* Tester issues its Claim frame with lowest T_Bid.  */
      pCLAIM = CLAIM_T3();
      Send_PDU_CLAIM(pCLAIM);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 986 "FDDI.MP" -- [6]? CLAIM */
        /* IUT issues its Claim frame again.  */
        pCLAIM = CLAIM_R1();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
          _level += 1;
          while (TRUE) {
            /* line 990 "FDDI.MP" -- [7]! CLAIM */
            /* Tester repeats IUT's Claim frame.  */
            pCLAIM = CLAIM_T1();
            Send_PDU_CLAIM(pCLAIM);
            Start_timer("TRT.", (long) TTRT 1000);
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 994 "FDDI.MP" -- [8]? TOKEN */
              /* IUT issues Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 1000 "FDDI.MP" -- [8]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 1003 "FDDI.MP" -- [8]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }             /* end of level [8] */
          }               /* end of level [7] */
        }
        /* line 1006 "FDDI.MP" -- [6]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1009 "FDDI.MP" -- [6]? TIMEOUT */
        if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                     /* end of level [6] */
          }                       /* end of level [5] */
        }
        /* line 1012 "FDDI.MP" -- [4]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1015 "FDDI.MP" -- [4]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                           /* end of level [4] */
    }                             /* end of level [3] */
  }                               /* end of level [2] */
}
_level = _lastlevel;
return (FALSE);
}                                 /* end of level [1] */


/*
 * TesterGotOwnBeacon -- Testing MAC Beacon process 1: Tester receives its
 * own Beacon
 */
int
TesterGotOwnBeacon()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1024 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
C:
    while (TRUE) {
      /* line 1029 "FDDI.MP" -- [2]? TIMEOUT */
      /* Tester holds Token and sends Idle symbols until TVX expires.  */
      if (Timeout("TVX.")) {
        _level += 1;
        return (FALSE);
      }
      /* line 1031 "FDDI.MP" -- [2]! IDLE */
      pIDLE = IDLE_I();
      Send_PDU_IDLE(pIDLE);
      _level += 1;
      while (TRUE) {
        /* line 1034 "FDDI.MP" -- [3]   */
        if ((a == 0)) {
          _level += 1;
          /* line 1036 "FDDI.MP" -- [4]? GOTO */
  goto C;
}
/* line 1038 "FDDI.MP" -- [3]   */
Start_timer("TRT.TRTclaim", (long) T_Max 1000);
_level += 1;
_timername = "TRT.";
while (TRUE) {
  /* line 1040 "FDDI.MP" -- [4]? CLAIM */
  /* IUT issues Claim frame.  */
  pCLAIM = CLAIM_R1();
  _buffer = Receive_PDU(_timername);
  if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
    _level += 1;
    while (TRUE) {
      /* line 1045 "FDDI.MP" -- [5]? TIMEOUT */
      /*
       * Tester holds IUT's Claim frame to cause that the Claim Token
       * process fails.
       */
      if (Timeout("TRT.")) {
        _level += 1;
        while (TRUE) {
          /* line 1047 "FDDI.MP" -- [6]   */
          Start_timer("TRT.TRTbeacon", (long) T_Max 1000);
          _level += 1;
          _timername = "TRT.";
          while (TRUE) {
            /* line 1049 "FDDI.MP" -- [7]? BEACON */
            /* IUT issues Beacon frame.  */
            pBEACON = BEACON_R1();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_BEACON(pBEACON, _buffer)) {
              _level += 1;
              while (TRUE) {
                /* line 1053 "FDDI.MP" -- [8]! BEACON */
                /*
                 * Tester holds IUT's Beacon frame and issues its own
                 * Beacon frame.
                 */
                pBEACON = BEACON_T2();
                Send_PDU_BEACON(pBEACON);
                _level += 1;
                _timername = "TRT.";
                while (TRUE) {
                  /* line 1057 "FDDI.MP" -- [9]? BEACON */
                  /* IUT repeats Tester's Beacon frame.  */
                  pBEACON = BEACON_R2();
                  _buffer = Receive_PDU(_timername);
                  if (Is_PDU_BEACON(pBEACON, _buffer)) {
                    Start_timer("TRT.TRTclaim", (long) T_Max 1000);
                    _level += 1;
                    while (TRUE) {
                      /* line 1061 "FDDI.MP" -- [10]! CLAIM */
                      /*
                       * After Tester receives its own Beacon frame,
                       * it sends Claim frame.
                       */
                      pCLAIM = CLAIM_T4();
                      Send_PDU_CLAIM(pCLAIM);
                      _level += 1;
                      _timername = "TRT.";
                      while (TRUE) {
                        /* line 1065 "FDDI.MP" -- [11]? CLAIM */
                        /* IUT repeats Tester's Claim frame.  */
                        pCLAIM = CLAIM_R4();
                        _buffer = Receive_PDU(_timername);
        if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
          Start_timer("TRT.", (long) TTRT 1000);
          _level += 1;
          while (TRUE) {
            /* line 1069 "FDDI.MP" -- [12]! TOKEN */
            /*
             * Tester issues Token after it receives
             * its own Claim frame.
             */
            pTOKEN = TOKEN_T1();
            Send_PDU_TOKEN(pTOKEN);
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1073 "FDDI.MP" -- [13]? TOKEN */
              /* IUT repeats Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /*
               * line 1079 "FDDI.MP" -- [13]?
               * OTHERWISE
               */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 1082 "FDDI.MP" -- [13]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }   /* end of level [13] */
          }     /* end of level [12] */
        }
        /* line 1085 "FDDI.MP" -- [11]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1088 "FDDI.MP" -- [11]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      } /* end of level [11] */
    }   /* end of level [10] */
  }
  /* line 1091 "FDDI.MP" -- [9]? OTHERWISE */
  if (_buffer) {
    _level += 1;
    SetfVerdict(FAIL);
    return (TRUE);
  }
  /* line 1094 "FDDI.MP" -- [9]? TIMEOUT */
  if (Timeout("TRT.")) {
    _level += 1;
    SetfVerdict(FAIL);
```

```
                              return (TRUE);
                            }
                          }         /* end of level [9] */
                        }           /* end of level [8] */
                      }
                      /* line 1097 "FDDI.MP" -- [7]? OTHERWISE */
                      if (_buffer) {
                        _level += 1;
                        SetfVerdict(FAIL);
                        return (TRUE);
                      }
                      /* line 1100 "FDDI.MP" -- [7]? TIMEOUT */
                      if (Timeout("TRT.")) {
                        _level += 1;
                        SetfVerdict(FAIL);
                        return (TRUE);
                      }
                    }               /* end of level [7] */
                  }                 /* end of level [6] */
                }
              }                     /* end of level [5] */
            }
            /* line 1103 "FDDI.MP" -- [4]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(INCONC);
              return (TRUE);
            }
            /* line 1106 "FDDI.MP" -- [4]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(INCONC);
              return (TRUE);
            }
          }                         /* end of level [4] */
        }                           /* end of level [3] */
      }                             /* end of level [2] */
    }
    _level = _lastlevel;
    return (FALSE);
}                                   /* end of level [1] */


/*
 * IUTGotOwnBeacon -- Testing MAC Beacon process 2: IUT receives its own
 * Beacon
 */
int
IUTGotOwnBeacon()
{
  int               _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1114 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
 D:
    while (TRUE) {
```

```
/* line 1119 "FDDI.MP" -- [2]? TIMEOUT */
/* Tester holds Token and sends Idle symbols until TVX expires.  */
if (Timeout("TVX.")) {
  _level += 1;
  return (FALSE);
}
/* line 1121 "FDDI.MP" -- [2]! IDLE */
pIDLE = IDLE_I();
Send_PDU_IDLE(pIDLE);
_level += 1;
while (TRUE) {
  /* line 1124 "FDDI.MP" -- [3]   */
  if ((a == 0)) {
    _level += 1;
    /* line 1126 "FDDI.MP" -- [4]? GOTO */
    goto D;
  }
  /* line 1128 "FDDI.MP" -- [3]   */
  Start_timer("TRT.TRTclaim", (long) T_Max 1000);
  _level += 1;
  _timername = "TRT.";
  while (TRUE) {
    /* line 1130 "FDDI.MP" -- [4]? CLAIM */
    /* IUT issues Claim frame.  */
    pCLAIM = CLAIM_R1();
    _buffer = Receive_PDU(_timername);
    if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
      _level += 1;
      while (TRUE) {
        /* line 1135 "FDDI.MP" -- [5]? TIMEOUT */
        /*
         * Tester holds IUT's Claim frame to cause that the Claim frame
         * fails.
         */
        if (Timeout("TRT.")) {
          _level += 1;
          while (TRUE) {
            /* line 1137 "FDDI.MP" -- [6]   */
            Start_timer("TRT.TRTbeacon", (long) T_Max 1000);
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1139 "FDDI.MP" -- [7]? BEACON */
              /* IUT issues Beacon frame.  */
              pBEACON = BEACON_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_BEACON(pBEACON, _buffer)) {
                _level += 1;
                while (TRUE) {
                  /* line 1143 "FDDI.MP" -- [8]! BEACON */
                  /* Tester repeats IUT's Beacon frame.  */
                  pBEACON = BEACON_T1();
                  Send_PDU_BEACON(pBEACON);
                  Start_timer("TRT.TRTclaim", (long) T_Max 1000);
                  _level += 1;
                  _timername = "TRT.";
                  while (TRUE) {
                    /* line 1147 "FDDI.MP" -- [9]? CLAIM */
                    /*
                     * After IUT receives its own Beacon frame, it
                     * issues Claim frame.
                     */
                    pCLAIM = CLAIM_R1();
                    _buffer = Receive_PDU(_timername);
                    if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
                      _level += 1;
            while (TRUE) {
              /* line 1151 "FDDI.MP" -- [10]! CLAIM */
              /* Tester repeats IUT's Beacon frame.  */
              pCLAIM = CLAIM_T1();
              Send_PDU_CLAIM(pCLAIM);
              Start_timer("TRT.", (long) TTRT 1000);
              _level += 1;
              _timername = "TRT.";
              while (TRUE) {
                /* line 1155 "FDDI.MP" -- [11]? TOKEN */
                /*
                 * After IUT wins Claim Token process, it
                 * issues Token.
                 */
                pTOKEN = TOKEN_R1();
                _buffer = Receive_PDU(_timername);
                if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                  _level += 1;
                  SetfVerdict(PASS);
                  return (TRUE);
                }
                /* line 1161 "FDDI.MP" -- [11]? OTHERWISE */
                if (_buffer) {
                  _level += 1;
                  SetfVerdict(FAIL);
                  return (TRUE);
                }
                /* line 1164 "FDDI.MP" -- [11]? TIMEOUT */
                if (Timeout("TRT.")) {
                  _level += 1;
                  SetfVerdict(FAIL);
                  return (TRUE);
                }
              } /* end of level [11] */
            }   /* end of level [10] */
          }
          /* line 1167 "FDDI.MP" -- [9]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 1170 "FDDI.MP" -- [9]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }       /* end of level [9] */
      }         /* end of level [8] */
    }
    /* line 1173 "FDDI.MP" -- [7]? OTHERWISE */
    if (_buffer) {
      _level += 1;
      SetfVerdict(FAIL);
      return (TRUE);
    }
    /* line 1176 "FDDI.MP" -- [7]? TIMEOUT */
    if (Timeout("TRT.")) {
      _level += 1;
      SetfVerdict(FAIL);
      return (TRUE);
    }
  }             /* end of level [7] */
}               /* end of level [6] */
}
```

```
          }                     /* end of level [5] */
        }
        /* line 1179 "FDDI.MP" -- [4]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(INCONC);
          return (TRUE);
        }
        /* line 1182 "FDDI.MP" -- [4]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(INCONC);
          return (TRUE);
        }
      }                         /* end of level [4] */
    }                           /* end of level [3] */
  }                             /* end of level [2] */
 }
 _level = _lastlevel;
 return (FALSE);
}                               /* end of level [1] */


/*
 * EarlyToken -- Testing MAC Timed Token Rotation protocol(TTRP) 1: Tester
 * receives Early Token
 */
int
EarlyToken()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1191 "FDDI.MP" -- [1]+  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1193 "FDDI.MP" -- [2]! ECHO_Req */
      /* Tester sends SMT ECHO Request frame.  */
      pECHO_Req = ECHO_Req_T1();
      Send_PDU_ECHO_Req(pECHO_Req);
      _level += 1;
      while (TRUE) {
        /* line 1197 "FDDI.MP" -- [3]! TOKEN */
        /* Tester issues Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1201 "FDDI.MP" -- [4]? ECHO_Resp */
          /* IUT returns ECHO Response frame.  */
          pECHO_Resp = ECHO_Resp_R1();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_ECHO_Resp(pECHO_Resp, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1205 "FDDI.MP" -- [5]? TOKEN */
              /* IUT returns Token (TRT<TTRT).  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 1211 "FDDI.MP" -- [5]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 1214 "FDDI.MP" -- [5]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                   /* end of level [5] */
          }
          /* line 1217 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 1220 "FDDI.MP" -- [4]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                       /* end of level [4] */
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * LateToken -- Testing MAC Timed Token Rotation Protocol(TTRP) 2: Tester
 * receives Late Token
 */
int
LateToken()
{
  int                _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1228 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1231 "FDDI.MP" -- [2]! ECHO_Req */
/* Tester sends SMT ECHO Request frame,  */
pECHO_Req = ECHO_Req_T1();
Send_PDU_ECHO_Req(pECHO_Req);
_level += 1;
while (TRUE) {
  /* line 1235 "FDDI.MP" -- [3]? TIMEOUT */
  if (Timeout("TRT.")) {
    _level += 1;
    while (TRUE) {
      /* line 1237 "FDDI.MP" -- [4]   */
      Start_timer("TRT.", (long) TTRT 1000);
      _level += 1;
      while (TRUE) {
        /* line 1239 "FDDI.MP" -- [5]! TOKEN */
        /* and issues Late Token when TTRT<TRT<2xTTRT.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1243 "FDDI.MP" -- [6]? TOKEN */
          /* IUT returns Token.  */
          pTOKEN = TOKEN_R1();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
            Start_timer("TRT.", (long) TTRT 1000);
            _level += 1;
            while (TRUE) {
              /* line 1247 "FDDI.MP" -- [7]! ECHO_Req */
              /* Tester sends SMT ECHO Request frame again,  */
              pECHO_Req = ECHO_Req_T1();
              Send_PDU_ECHO_Req(pECHO_Req);
              _level += 1;
              while (TRUE) {
                /* line 1251 "FDDI.MP" -- [8]? TIMEOUT */
                if (Timeout("TRT.")) {
                  _level += 1;
                  while (TRUE) {
                    /* line 1253 "FDDI.MP" -- [9]   */
                    Start_timer("TRT.", (long) TTRT 1000);
                    _level += 1;
                    while (TRUE) {
                      /* line 1255 "FDDI.MP" -- [10]! TOKEN */
                      /*
                       * and issues Late Token again when
                       * TTRT<TRT<2xTTRT.
                       */
                      pTOKEN = TOKEN_T1();
                      Send_PDU_TOKEN(pTOKEN);
                      _level += 1;
                      _timername = "TRT.";
                      while (TRUE) {
                        /* line 1259 "FDDI.MP" -- [11]? TOKEN */
                        /* IUT returns Token.  */
                        pTOKEN = TOKEN_R1();
                        _buffer = Receive_PDU(_timername);
                        if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                          _level += 1;
                          SetfVerdict(PASS);
                          return (TRUE);
                        }
                        /* line 1265 "FDDI.MP" -- [11]? OTHERWISE */
                        if (_buffer) {
                          _level += 1;
                          SetfVerdict(FAIL);
                          return (TRUE);
```

```
                               }
                               /* line 1268 "FDDI.MP" -- [11]? TIMEOUT */
                               if (Timeout("TRT.")) {
                                 _level += 1;
                                 SetfVerdict(FAIL);
                                 return (TRUE);
                               }
                             }   /* end of level [11] */
                           }     /* end of level [10] */
                         }       /* end of level [9] */
                       }
                     }           /* end of level [8] */
                   }             /* end of level [7] */
                 }
                 /* line 1271 "FDDI.MP" -- [6]? OTHERWISE */
                 if (_buffer) {
                   _level += 1;
                   SetfVerdict(FAIL);
                   return (TRUE);
                 }
                 /* line 1274 "FDDI.MP" -- [6]? TIMEOUT */
                 if (Timeout("TRT.")) {
                   _level += 1;
                   SetfVerdict(FAIL);
                   return (TRUE);
                 }
               }                 /* end of level [6] */
             }                   /* end of level [5] */
           }                     /* end of level [4] */
         }
       }                         /* end of level [3] */
     }                           /* end of level [2] */
   }
   _level = _lastlevel;
   return (FALSE);
}                                /* end of level [1] */


/*
 * TRTtesting -- Testing MAC Monitoring function 1: TRT Monitoring Function
 */
int
TRTtesting()
{
  int               _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1283 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1286 "FDDI.MP" -- [2]! ECHO_Req */
      /* Tester sends SMT ECHO Request frame.  */
      pECHO_Req = ECHO_Req_T1();
      Send_PDU_ECHO_Req(pECHO_Req);
      _level += 1;
      while (TRUE) {
        /* line 1291 "FDDI.MP" -- [3]? TIMEOUT */
/* Causing TRT expired.  */
if (Timeout("TRT.")) {
  _level += 1;
  while (TRUE) {
    /* line 1293 "FDDI.MP" -- [4]   */
    Start_timer("TRT.", (long) TTRT 1000);
    _level += 1;
    while (TRUE) {
      /* line 1296 "FDDI.MP" -- [5]? TIMEOUT */
      /* Causing TRT expired twice and Token never has been seen.  */
      if (Timeout("TRT.")) {
        _level += 1;
        while (TRUE) {
          /* line 1298 "FDDI.MP" -- [6]   */
          Start_timer("TRT.TRTclaim", (long) T_Max 1000);
          _level += 1;
          _timername = "TRT.";
          while (TRUE) {
            /* line 1300 "FDDI.MP" -- [7]? CLAIM */
            /* IUT initiates Claim Token process.  */
            pCLAIM = CLAIM_R1();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
              _level += 1;
              while (TRUE) {
                /* line 1304 "FDDI.MP" -- [8]! CLAIM */
                /* Tester repeats Claim frame.  */
                pCLAIM = CLAIM_T1();
                Send_PDU_CLAIM(pCLAIM);
                Start_timer("TRT.", (long) TTRT 1000);
                _level += 1;
                _timername = "TRT.";
                while (TRUE) {
                  /* line 1308 "FDDI.MP" -- [9]? TOKEN */
                  /* IUT issues Token.  */
                  pTOKEN = TOKEN_R1();
                  _buffer = Receive_PDU(_timername);
                  if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                    _level += 1;
                    SetfVerdict(PASS);
                    return (TRUE);
                  }
                  /* line 1314 "FDDI.MP" -- [9]? OTHERWISE */
                  if (_buffer) {
                    _level += 1;
                    SetfVerdict(FAIL);
                    return (TRUE);
                  }
                  /* line 1317 "FDDI.MP" -- [9]? TIMEOUT */
                  if (Timeout("TRT.")) {
                    _level += 1;
                    SetfVerdict(FAIL);
                    return (TRUE);
                  }
                }       /* end of level [9] */
              }         /* end of level [8] */
            }
            /* line 1320 "FDDI.MP" -- [7]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 1323 "FDDI.MP" -- [7]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
```

```
                      SetfVerdict(FAIL);
                      return (TRUE);
                    }
                  }             /* end of level [7] */
                }               /* end of level [6] */
              }
            }                   /* end of level [5] */
          }                     /* end of level [4] */
        }
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * TVXtesting -- Testing MAC Monitoring function 2: TVX Monitoring
 */
int
TVXtesting()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1331 "FDDI.MP" -- [1]+  */
  /* Initializing Connection(test group)  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    _timername = (char *) 0;
    _time = 0;
    _bufferlen = -1;
    while (TRUE) {
      /* line 1334 "FDDI.MP" -- [2]+  */
      /* Subtree 1  */
      _lastlevel = _level;
      if (TVX_SUBTREE1())
        return (TRUE);
      else if (_level > _lastlevel) {
        _level += 1;
        _timername = (char *) 0;
        _time = 0;
        _bufferlen = -1;
        while (TRUE) {
          /* line 1337 "FDDI.MP" -- [3]+  */
          /* Initializing Connection(test group)  */
          _lastlevel = _level;
          if (INITCONNECT())
            return (TRUE);
          else if (_level > _lastlevel) {
            _level += 1;
            _timername = (char *) 0;
            _time = 0;
            _bufferlen = -1;
            while (TRUE) {
              /* line 1340 "FDDI.MP" -- [4]+  */
              /* Subtree 2  */
              _lastlevel = _level;
              if (TVX_SUBTREE2())
                return (TRUE);
              else if (_level > _lastlevel) {
                _level += 1;
                return (FALSE);
              }
              _level = _lastlevel;
            }                     /* end of level [4] */
          }
          _level = _lastlevel;
        }                         /* end of level [3] */
      }
      _level = _lastlevel;
    }                             /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                 /* end of level [1] */
int
TVX_SUBTREE1()
{
  int             _lastlevel;
  /* line 1344 "FDDI.MP" -- [1]! DATA_OverMaxLen */
  /*
   * Tester sends tester frame(DA=IUT Address; Frame symbol time>TVX), to
   * cause TVX expire.
   */
  pDATA_OverMaxLen = DATA_OverMaxLen_T1();
  Send_PDU_DATA_OverMaxLen(pDATA_OverMaxLen);
  _level += 1;
  while (TRUE) {
    /* line 1348 "FDDI.MP" -- [2]   */
    Start_timer("TRT.TRTclaim", (long) T_Max 1000);
    _level += 1;
    _timername = "TRT.";
    while (TRUE) {
      /* line 1350 "FDDI.MP" -- [3]? CLAIM */
      /* IUT initiates Claim Token process.  */
      pCLAIM = CLAIM_R1();
      _buffer = Receive_PDU(_timername);
      if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
        _level += 1;
        while (TRUE) {
          /* line 1354 "FDDI.MP" -- [4]! CLAIM */
          /* Tester repeats the Claim frame sent by IUT.  */
          pCLAIM = CLAIM_T1();
          Send_PDU_CLAIM(pCLAIM);
          Start_timer("TRT.", (long) TTRT 1000);
          _level += 1;
          _timername = "TRT.";
          while (TRUE) {
            /* line 1358 "FDDI.MP" -- [5]? TOKEN */
            /* IUT issues Token.  */
            pTOKEN = TOKEN_R1();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
              _level += 1;
              SetfVerdict(PASS);
              return (FALSE);
            }
            /* line 1364 "FDDI.MP" -- [5]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (FALSE);
            }
            /* line 1367 "FDDI.MP" -- [5]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (FALSE);
            }
          }                       /* end of level [5] */
        }                         /* end of level [4] */
      }
      /* line 1370 "FDDI.MP" -- [3]? OTHERWISE */
      if (_buffer) {
        _level += 1;
        SetfVerdict(FAIL);
        return (FALSE);
      }
      /* line 1373 "FDDI.MP" -- [3]? TIMEOUT */
      if (Timeout("TRT.")) {
        _level += 1;
        SetfVerdict(FAIL);
        return (FALSE);
      }
    }                             /* end of level [3] */
  }                               /* end of level [2] */
}                                 /* end of level [1] */
int
TVX_SUBTREE2()
{
  int             _lastlevel;
E:
  if (timername == 0)
    _timername = "TVX.";
  /* line 1378 "FDDI.MP" -- [1]? TIMEOUT */
  /* Holds Token and sends Idle symbols until TVX expires.  */
  if (Timeout("TVX.")) {
    _level += 1;
    return (FALSE);
  }
  /* line 1380 "FDDI.MP" -- [1]! IDLE */
  pIDLE = IDLE_I();
  Send_PDU_IDLE(pIDLE);
  _level += 1;
  while (TRUE) {
    /* line 1383 "FDDI.MP" -- [2]   */
    if ((a == 0)) {
      _level += 1;
      /* line 1385 "FDDI.MP" -- [3]? GOTO */
      goto E;
    }
    /* line 1387 "FDDI.MP" -- [2]   */
    Start_timer("TRT.TRTclaim", (long) T_Max 1000);
    _level += 1;
    _timername = "TRT.";
    while (TRUE) {
      /* line 1389 "FDDI.MP" -- [3]? CLAIM */
      /* IUT starts Claim Token process.  */
      pCLAIM = CLAIM_R1();
      _buffer = Receive_PDU(_timername);
      if (Is_PDU_CLAIM(pCLAIM, _buffer)) {
        _level += 1;
        while (TRUE) {
          /* line 1393 "FDDI.MP" -- [4]! CLAIM */
          pCLAIM = CLAIM_T1();
          Send_PDU_CLAIM(pCLAIM);
          Start_timer("TRT.", (long) TTRT 1000);
          _level += 1;
          _timername = "TRT.";
          while (TRUE) {
            /* line 1396 "FDDI.MP" -- [5]? TOKEN */
            pTOKEN = TOKEN_R1();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
              _level += 1;
              SetfVerdict(RESULT);
              return (TRUE);
            }
            /* line 1401 "FDDI.MP" -- [5]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 1404 "FDDI.MP" -- [5]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
          }                     /* end of level [5] */
        }                       /* end of level [4] */
      }
      /* line 1407 "FDDI.MP" -- [3]? OTHERWISE */
      if (_buffer) {
        _level += 1;
        SetfVerdict(FAIL);
        return (TRUE);
      }
      /* line 1410 "FDDI.MP" -- [3]? TIMEOUT */
      if (Timeout("TRT.")) {
        _level += 1;
        SetfVerdict(FAIL);
        return (TRUE);
      }
    }                           /* end of level [3] */
  }                             /* end of level [2] */
}                               /* end of level [1] */


/*
 * PHInvalidR10b -- Testing MAC Frame Error Detection 1: R(10b) -- a
 * transition at MAC receive state machine
 */
int
PHInvalidR10b()
{
  int            _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1419 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1422 "FDDI.MP" -- [2]! DATA */
      /* Tester sends the frame(DA=IUT Address; A Invalid Symbol in PA)  */
      pDATA = DATA_T3();
      Send_PDU_DATA(pDATA);
      _level += 1;
      while (TRUE) {
        /* line 1426 "FDDI.MP" -- [3]! TOKEN */
        /* followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1430 "FDDI.MP" -- [4]? DATA_Strip */
          /* Idles returned  */
          pDATA_Strip = DATA_Strip_R1();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1434 "FDDI.MP" -- [5]? TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 1440 "FDDI.MP" -- [5]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 1443 "FDDI.MP" -- [5]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                   /* end of level [5] */
          }
          /* line 1446 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 1449 "FDDI.MP" -- [4]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                       /* end of level [4] */
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * PHInvalidR20b -- Testing MAC Frame Error Detection 2: R(20b) -- a
 * transition at MAC receive state machine
 */
```

```
int
PHInvalidR20b()
{
  int               _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1457 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1460 "FDDI.MP" -- [2]! DATA */
      /*
       * Tester sends the frame(DA=IUT Address; a Invalid Symbol after J in
       * SD)
       */
      pDATA = DATA_T4();
      Send_PDU_DATA(pDATA);
      _level += 1;
      while (TRUE) {
        /* line 1464 "FDDI.MP" -- [3]! TOKEN */
        /* followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1468 "FDDI.MP" -- [4]? DATA_Strip */
          /* PA returned,  */
          pDATA_Strip = DATA_Strip_R1();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1472 "FDDI.MP" -- [5]? TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 1478 "FDDI.MP" -- [5]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 1481 "FDDI.MP" -- [5]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                   /* end of level [5] */
          }
          /* line 1484 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 1487 "FDDI.MP" -- [4]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                           /* end of level [4] */
      }                             /* end of level [3] */
    }                               /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                   /* end of level [1] */


/*
 * PHInvalidR30b -- Testing MAC Frame Error Detection 3: R(30b) -- a
 * transition at MAC receive state machine
 */
int
PHInvalidR30b()
{
  int               _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1495 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1498 "FDDI.MP" -- [2]! DATA */
      /* Tester sends the frame(DA=IUT Address; a Invalid Symbol in INFO)  */
      pDATA = DATA_T5();
      Send_PDU_DATA(pDATA);
      _level += 1;
      while (TRUE) {
        /* line 1502 "FDDI.MP" -- [3]! TOKEN */
        /* followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1506 "FDDI.MP" -- [4]? DATA_Strip */
          /* PA,SD,FC,DA,SA and Idles returned,  */
          pDATA_Strip = DATA_Strip_R2();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1510 "FDDI.MP" -- [5]? TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_R1();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
              _level += 1;
              SetfVerdict(PASS);
              return (TRUE);
            }
            /* line 1516 "FDDI.MP" -- [5]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 1519 "FDDI.MP" -- [5]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
          }                     /* end of level [5] */
        }
        /* line 1522 "FDDI.MP" -- [4]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1525 "FDDI.MP" -- [4]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                         /* end of level [4] */
    }                           /* end of level [3] */
  }                             /* end of level [2] */
 }
 _level = _lastlevel;
 return (FALSE);
}                               /* end of level [1] */


/*
 * PHInvalidR40b -- Testing MAC Frame Error Detection 4: R(40b) -- a
 * transition at MAC receive state machine
 */
int
PHInvalidR40b()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1533 "FDDI.MP" -- [1]+  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1535 "FDDI.MP" -- [2]! DATA */
      /* Tester sends the frame(DA=IUT Address; a Invalid Symbol in FS)  */
      pDATA = DATA_T6();
      Send_PDU_DATA(pDATA);
      _level += 1;
      while (TRUE) {
        /* line 1539 "FDDI.MP" -- [3]! TOKEN */
        /* followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1543 "FDDI.MP" -- [4]? DATA_Strip */
          /* PA, SD, FC, DA, SA, INFO, FCS, ED and FS returned(E=S, A=S, C=S)  */
          pDATA_Strip = DATA_Strip_R4();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1547 "FDDI.MP" -- [5]? TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 1553 "FDDI.MP" -- [5]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 1556 "FDDI.MP" -- [5]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                   /* end of level [5] */
          }
          /* line 1559 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 1562 "FDDI.MP" -- [4]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                       /* end of level [4] */
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * DetectSD -- Testing MAC Frame Error Detection 5: Detecting SD
 */
int
DetectSD()
```

```
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1570 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1573 "FDDI.MP" -- [2]! DATA */
      /* Tester sends the frame(DA=IUT Address; an Idle Symbol after J)  */
      pDATA = DATA_T7_0();
      Send_PDU_DATA(pDATA);
      _level += 1;
      while (TRUE) {
        /* line 1577 "FDDI.MP" -- [3]! TOKEN */
        /* Followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1581 "FDDI.MP" -- [4]? DATA_Strip */
          /* PA and Idles returned.  */
          pDATA_Strip = DATA_Strip_R1();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1585 "FDDI.MP" -- [5]? TOKEN */
              /* Followed by Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (FALSE);
              }
              /* line 1591 "FDDI.MP" -- [5]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (FALSE);
              }
              /* line 1594 "FDDI.MP" -- [5]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (FALSE);
              }
            }                   /* end of level [5] */
          }
          /* line 1597 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (FALSE);
          }
```

```
/* line 1600 "FDDI.MP" -- [4]? TIMEOUT */
if (Timeout("TRT.")) {
  _level += 1;
  SetfVerdict(FAIL);
  while (TRUE) {
    /* line 1602 "FDDI.MP" -- [5]! DATA */
    /*
     * Tester sends the frame(DA=IUT Address; other symbol(not K
     * symbol) before K in SD)
     */
    pDATA = DATA_T7_1();
    Send_PDU_DATA(pDATA);
    Start_timer("TRT.", (long) TTRT 1000);
    _level += 1;
    while (TRUE) {
      /* line 1606 "FDDI.MP" -- [6]! TOKEN */
      /* followed by Token.  */
      pTOKEN = TOKEN_T1();
      Send_PDU_TOKEN(pTOKEN);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 1610 "FDDI.MP" -- [7]? DATA_Strip */
        /* PA and Idle returned,  */
        pDATA_Strip = DATA_Strip_R1();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
          _level += 1;
          _timername = "TRT.";
          while (TRUE) {
            /* line 1614 "FDDI.MP" -- [8]? TOKEN */
            /* followed by Token.  */
            pTOKEN = TOKEN_R1();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
              _level += 1;
              SetfVerdict(RESULT);
              return (TRUE);
            }
            /* line 1620 "FDDI.MP" -- [8]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 1623 "FDDI.MP" -- [8]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
          }           /* end of level [8] */
        }
        /* line 1626 "FDDI.MP" -- [7]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1629 "FDDI.MP" -- [7]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }              /* end of level [7] */
```

```
              }                   /* end of level [6] */
            }                     /* end of level [5] */
          }
        }                         /* end of level [4] */
      }                           /* end of level [3] */
    }                             /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                 /* end of level [1] */


/*
 * DetectFC -- Testing MAC Frame Error Detection 6: Detecting FC
 */
int
DetectFC()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1637 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1640 "FDDI.MP" -- [2]! DATA */
      /*
       * Tester sends the frame(DA=IUT Address; a symbol sequence(not Idle(s)
       * or nn) after K in FC),
       */
      pDATA = DATA_T8();
      Send_PDU_DATA(pDATA);
      _level += 1;
      while (TRUE) {
        /* line 1644 "FDDI.MP" -- [3]! TOKEN */
        /* followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1648 "FDDI.MP" -- [4]? DATA_Strip */
          /* PA, SD and Idles returned,  */
          pDATA_Strip = DATA_Strip_R5();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1652 "FDDI.MP" -- [5]? TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
          /* line 1658 "FDDI.MP" -- [5]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 1661 "FDDI.MP" -- [5]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                   /* end of level [5] */
      }
      /* line 1664 "FDDI.MP" -- [4]? OTHERWISE */
      if (_buffer) {
        _level += 1;
        SetfVerdict(FAIL);
        return (TRUE);
      }
      /* line 1667 "FDDI.MP" -- [4]? TIMEOUT */
      if (Timeout("TRT.")) {
        _level += 1;
        SetfVerdict(FAIL);
        return (TRUE);
      }
    }                       /* end of level [4] */
   }                        /* end of level [3] */
  }                         /* end of level [2] */
 }
 _level = _lastlevel;
 return (FALSE);
}                           /* end of level [1] */


/*
 * DetectFrameBody1 -- Testing MAC Frame Error DEtection 7: Detecting Frame
 * Body 1
 */
int
DetectFrameBody1()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1675 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1678 "FDDI.MP" -- [2]! DATA */
      /* Tester sends the frame(DA=IUT Address; an Idle symbol in INFO),  */
      pDATA = DATA_T9();
      Send_PDU_DATA(pDATA);
      _level += 1;
      while (TRUE) {
        /* line 1682 "FDDI.MP" -- [3]! TOKEN */
        /* followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1686 "FDDI.MP" -- [4]? DATA_Strip */
          /* PA, SD, FC, DA, SA and Idles returned,  */
          pDATA_Strip = DATA_Strip_R2();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1690 "FDDI.MP" -- [5]? TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
              if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                _level += 1;
                SetfVerdict(PASS);
                return (TRUE);
              }
              /* line 1696 "FDDI.MP" -- [5]? OTHERWISE */
              if (_buffer) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
              /* line 1699 "FDDI.MP" -- [5]? TIMEOUT */
              if (Timeout("TRT.")) {
                _level += 1;
                SetfVerdict(FAIL);
                return (TRUE);
              }
            }                   /* end of level [5] */
          }
          /* line 1702 "FDDI.MP" -- [4]? OTHERWISE */
          if (_buffer) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
          /* line 1705 "FDDI.MP" -- [4]? TIMEOUT */
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                       /* end of level [4] */
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * DetectFrameBody2 -- Testing MAC Frame Error Detection 8: Detecting Frame
 * Body 2
 */
int
DetectFrameBody2()
{
  int           _lastlevel;
  _time = 0;
  _level = 1;
```

```
strcpy(R, "NONE");
Result = NONE;
_timername = (char *) 0;
_bufferlen = -1;
/* line 1713 "FDDI.MP" -- [1]+  */
/* Initializing Connection  */
_lastlevel = _level;
if (INITCONNECT())
  return (TRUE);
else if (_level > _lastlevel) {
  _level += 1;
  while (TRUE) {
    /* line 1716 "FDDI.MP" -- [2]! DATA */
    /*
     * Tester sends the frame(DA=IUT Address; a symbol(not Idle or data) in
     * INFO),
     */
    pDATA = DATA_T10();
    Send_PDU_DATA(pDATA);
    _level += 1;
    while (TRUE) {
      /* line 1720 "FDDI.MP" -- [3]! TOKEN */
      /* followed by Token.  */
      pTOKEN = TOKEN_T1();
      Send_PDU_TOKEN(pTOKEN);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 1724 "FDDI.MP" -- [4]? DATA_Strip */
        /* PA, SD, FC, DA, SA and Idles returned,  */
        pDATA_Strip = DATA_Strip_R2();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
          _level += 1;
          _timername = "TRT.";
          while (TRUE) {
            /* line 1728 "FDDI.MP" -- [5]? TOKEN */
            /* followed by Token  */
            pTOKEN = TOKEN_R1();
            _buffer = Receive_PDU(_timername);
            if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
              _level += 1;
              SetfVerdict(PASS);
              return (TRUE);
            }
            /* line 1734 "FDDI.MP" -- [5]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 1737 "FDDI.MP" -- [5]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
          }                   /* end of level [5] */
        }
        /* line 1740 "FDDI.MP" -- [4]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1743 "FDDI.MP" -- [4]? TIMEOUT */
```

```
          if (Timeout("TRT.")) {
            _level += 1;
            SetfVerdict(FAIL);
            return (TRUE);
          }
        }                           /* end of level [4] */
      }                             /* end of level [3] */
    }                               /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                   /* end of level [1] */


/*
 * DetectInvalidLength -- Testing Frame Error Detection 9: Detecting Invalid
 * Data Length Frame
 */
int
DetectInvalidLength()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1751 "FDDI.MP" -- [1]+  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1753 "FDDI.MP" -- [2]! DATA_InvLen */
      /*
       * Tester sends the frame (DA=IUT Address; a Invalid Data Length
       * Frame),
       */
      pDATA_InvLen = DATA_InvLen_T1();
      Send_PDU_DATA_InvLen(pDATA_InvLen);
      _level += 1;
      while (TRUE) {
        /* line 1757 "FDDI.MP" -- [3]! TOKEN */
        /* followed by Token.  */
        pTOKEN = TOKEN_T1();
        Send_PDU_TOKEN(pTOKEN);
        _level += 1;
        _timername = "TRT.";
        while (TRUE) {
          /* line 1761 "FDDI.MP" -- [4]? DATA_Strip */
          /*
           * PA, SD, FC, DA, SA, INFO, FCS, ED and FS returned(A=S, E=S,
           * C=R),
           */
          pDATA_Strip = DATA_Strip_R3();
          _buffer = Receive_PDU(_timername);
          if (Is_PDU_DATA_Strip(pDATA_Strip, _buffer)) {
            _level += 1;
            _timername = "TRT.";
            while (TRUE) {
              /* line 1765 "FDDI.MP" -- [5]? TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_R1();
              _buffer = Receive_PDU(_timername);
            if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
              _level += 1;
              SetfVerdict(PASS);
              return (TRUE);
            }
            /* line 1771 "FDDI.MP" -- [5]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 1774 "FDDI.MP" -- [5]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
          }                     /* end of level [5] */
        }
        /* line 1777 "FDDI.MP" -- [4]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1780 "FDDI.MP" -- [4]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                         /* end of level [4] */
    }                           /* end of level [3] */
  }                             /* end of level [2] */
 }
 _level = _lastlevel;
 return (FALSE);
}                               /* end of level [1] */


/*
 * DetectFCS -- Testing Frame Error Detection 10: Detecting FCS Errors
 */
int
DetectFCS()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1788 "FDDI.MP" -- [1]+  */
  /* Intializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1791 "FDDI.MP" -- [2]! DATA */
      /* Tester sends the frame(DA=IUT Address; FCS errors),  */
      pDATA = DATA_T11_0();
      Send_PDU_DATA(pDATA);
      _level += 1;
```

```
while (TRUE) {
  /* line 1795 "FDDI.MP" -- [3]! TOKEN */
  /* followed by Token.  */
  pTOKEN = TOKEN_T1();
  Send_PDU_TOKEN(pTOKEN);
  _level += 1;
  _timername = "TRT.";
  while (TRUE) {
    /* line 1799 "FDDI.MP" -- [4]? DATA */
    /* IUT repeats Tester's frame(A=S,E=S,C=R),  */
    pDATA = DATA_R11_0();
    _buffer = Receive_PDU(_timername);
    if (Is_PDU_DATA(pDATA, _buffer)) {
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 1803 "FDDI.MP" -- [5]? TOKEN */
        /* Token returned.  */
        pTOKEN = TOKEN_R1();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
          _level += 1;
          while (TRUE) {
            /* line 1807 "FDDI.MP" -- [6]! DATA */
            /* Tester sends the frame(DA<>IUT Address; FCS error),  */
            pDATA = DATA_T11_1();
            Send_PDU_DATA(pDATA);
            _level += 1;
            while (TRUE) {
              /* line 1811 "FDDI.MP" -- [7]! TOKEN */
              /* followed by Token.  */
              pTOKEN = TOKEN_T1();
              Send_PDU_TOKEN(pTOKEN);
              _level += 1;
              _timername = "TRT.";
              while (TRUE) {
                /* line 1815 "FDDI.MP" -- [8]? DATA */
                /* IUT repeats Tester's frame(A=R, E=S, C=R)  */
                pDATA = DATA_R11_1();
                _buffer = Receive_PDU(_timername);
                if (Is_PDU_DATA(pDATA, _buffer)) {
                  _level += 1;
                  _timername = "TRT.";
                  while (TRUE) {
                    /* line 1819 "FDDI.MP" -- [9]? TOKEN */
                    /* IUT returns Token.  */
                    pTOKEN = TOKEN_R1();
                    _buffer = Receive_PDU(_timername);
                    if (Is_PDU_TOKEN(pTOKEN, _buffer)) {
                      _level += 1;
                      SetfVerdict(PASS);
                      return (TRUE);
                    }
                    /* line 1825 "FDDI.MP" -- [9]? OTHERWISE */
                    if (_buffer) {
                      _level += 1;
                      SetfVerdict(FAIL);
                      return (TRUE);
                    }
                    /* line 1828 "FDDI.MP" -- [9]? TIMEOUT */
                    if (Timeout("TRT.")) {
                      _level += 1;
                      SetfVerdict(FAIL);
                      return (TRUE);
                    }
                  }       /* end of level [9] */
```

```
                    }
                    /* line 1831 "FDDI.MP" -- [8]? OTHERWISE */
                    if (_buffer) {
                      _level += 1;
                      SetfVerdict(FAIL);
                      return (TRUE);
                    }
                    /* line 1834 "FDDI.MP" -- [8]? TIMEOUT */
                    if (Timeout("TRT.")) {
                      _level += 1;
                      SetfVerdict(FAIL);
                      return (TRUE);
                    }
                  }             /* end of level [8] */
                }               /* end of level [7] */
              }                 /* end of level [6] */
            }
            /* line 1837 "FDDI.MP" -- [5]? OTHERWISE */
            if (_buffer) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
            /* line 1840 "FDDI.MP" -- [5]? TIMEOUT */
            if (Timeout("TRT.")) {
              _level += 1;
              SetfVerdict(FAIL);
              return (TRUE);
            }
          }                     /* end of level [5] */
        }
        /* line 1843 "FDDI.MP" -- [4]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1846 "FDDI.MP" -- [4]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                         /* end of level [4] */
    }                           /* end of level [3] */
  }                             /* end of level [2] */
 }
 _level = _lastlevel;
 return (FALSE);
}                               /* end of level [1] */


/*
 * PHInvalidR50b -- Testing MAC Token Error Detection 1: R(50b) -- a
 * transition at MAC receive state machine
 */
int
PHInvalidR50b()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1855 "FDDI.MP" -- [1]+  */
  /* Initializing Connection  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1858 "FDDI.MP" -- [2]! TOKEN */
      /* Tester sends the Token (a Invalid Symbol in ED),  */
      pTOKEN = TOKEN_T2();
      Send_PDU_TOKEN(pTOKEN);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 1862 "FDDI.MP" -- [3]? TOKEN_Strip */
        /* PA,SD,FC and Idles returned.  */
        pTOKEN_Strip = TOKEN_Strip_R1();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_TOKEN_Strip(pTOKEN_Strip, _buffer)) {
          _level += 1;
          SetfVerdict(PASS);
          return (TRUE);
        }
        /* line 1868 "FDDI.MP" -- [3]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1871 "FDDI.MP" -- [3]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * DetectTokenED1 -- Testing 1 for ED of Token
 */
int
DetectTokenED1()
{
  int            _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1879 "FDDI.MP" -- [1]+  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1881 "FDDI.MP" -- [2]! TOKEN */
      /* Tester sends Token(an Idle Symbol in ED)  */
      pTOKEN = TOKEN_T3();
      Send_PDU_TOKEN(pTOKEN);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 1885 "FDDI.MP" -- [3]? TOKEN_Strip */
        /* PA, SD, FC and Idles returned.  */
        pTOKEN_Strip = TOKEN_Strip_R1();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_TOKEN_Strip(pTOKEN_Strip, _buffer)) {
          _level += 1;
          SetfVerdict(PASS);
          return (FALSE);
        }
        /* line 1891 "FDDI.MP" -- [3]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (FALSE);
        }
        /* line 1894 "FDDI.MP" -- [3]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (FALSE);
        }
      }                          /* end of level [3] */
    }                            /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                                /* end of level [1] */


/*
 * DetectTokenED2 -- Testing 2 for the ED of Token
 */
int
DetectTokenED2()
{
  int             _lastlevel;
  _time = 0;
  _level = 1;
  strcpy(R, "NONE");
  Result = NONE;
  _timername = (char *) 0;
  _bufferlen = -1;
  /* line 1902 "FDDI.MP" -- [1]+  */
  _lastlevel = _level;
  if (INITCONNECT())
    return (TRUE);
  else if (_level > _lastlevel) {
    _level += 1;
    while (TRUE) {
      /* line 1904 "FDDI.MP" -- [2]! TOKEN */
      /* Tester sends the Token(any symbol(not Idles or T) in ED)  */
      pTOKEN = TOKEN_T4();
      Send_PDU_TOKEN(pTOKEN);
      _level += 1;
      _timername = "TRT.";
      while (TRUE) {
        /* line 1908 "FDDI.MP" -- [3]? TOKEN_Strip */
        /* PA, SD, FC and Idles returned  */
        pTOKEN_Strip = TOKEN_Strip_R1();
        _buffer = Receive_PDU(_timername);
        if (Is_PDU_TOKEN_Strip(pTOKEN_Strip, _buffer)) {
          _level += 1;
          SetfVerdict(PASS);
          return (TRUE);
        }
        /* line 1914 "FDDI.MP" -- [3]? OTHERWISE */
        if (_buffer) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
        /* line 1917 "FDDI.MP" -- [3]? TIMEOUT */
        if (Timeout("TRT.")) {
          _level += 1;
          SetfVerdict(FAIL);
          return (TRUE);
        }
      }                         /* end of level [3] */
    }                           /* end of level [2] */
  }
  _level = _lastlevel;
  return (FALSE);
}                               /* end of level [1] */


/*
 * INITCONNECT -- to complete ring initialization and form a Token path for
 * test cases
 */
int
INITCONNECT()
{
  int             _lastlevel;
  /* line 1927 "FDDI.MP" -- [1]   */
  a = 0;
  _level += 1;
  return (FALSE);
}                               /* end of level [1] */
/*
```

```
 * ***Constraints Declarations**** */
pdu_DATA        *
DATA_T1()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = Tester_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA        *
DATA_R1()
{
  DATA.PA = 0;
  DATA.PA_tag = 6;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = Tester_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Rev;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA        *
DATA_T2()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
```

```
    DATA.FS = 0x294a;
    DATA.FS_tag = 0;
    return (&DATA);
  }
  pdu_DATA      *
  DATA_R2()
  {
    DATA.PA = 0;
    DATA.PA_tag = 6;
    DATA.SD = 0x311;
    DATA.SD_tag = 0;
    DATA.FC = 0xc1;
    DATA.FC_tag = 0;
    DATA.DA = IUT_Address;
    DATA.DA_tag = 0;
    DATA.SA = Tester_Address;
    DATA.SA_tag = 0;
    bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
    DATA.INFO_tag = 0;
    DATA.FCS = M_FCS_Rev;
    DATA.FCS_tag = 0;
    DATA.ED = 0x15;
    DATA.ED_tag = 0;
    DATA.FS = 0x296b;
    DATA.FS_tag = 0;
    return (&DATA);
  }
  pdu_DATA      *
  DATA_T3()
  {
    bitcpy(DATA.PA, 5, cvttobit("0xFFC1FFFFFFFFFFFFFFFF", 80), 80);
    DATA.PA_tag = 0;
    DATA.SD = 0x311;
    DATA.SD_tag = 0;
    DATA.FC = 0xc1;
    DATA.FC_tag = 0;
    DATA.DA = IUT_Address;
    DATA.DA_tag = 0;
    DATA.SA = Tester_Address;
    DATA.SA_tag = 0;
    bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
    DATA.INFO_tag = 0;
    DATA.FCS = M_FCS_Tramt;
    DATA.FCS_tag = 0;
    DATA.ED = 0x15;
    DATA.ED_tag = 0;
    DATA.FS = 0x294a;
    DATA.FS_tag = 0;
    return (&DATA);
  }
  pdu_DATA      *
  DATA_T4()
  {
    bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
    DATA.PA_tag = 0;
    DATA.SD = 0x300;
    DATA.SD_tag = 0;
    DATA.FC = 0xc1;
    DATA.FC_tag = 0;
    DATA.DA = IUT_Address;
    DATA.DA_tag = 0;
    DATA.SA = Tester_Address;
    DATA.SA_tag = 0;
    bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
    DATA.INFO_tag = 0;
    DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA        *
DATA_T5()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF029552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA        *
DATA_T6()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x280a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA        *
DATA_T7_0()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x31f;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_T7_1()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x3f1;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_T8()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0x004;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_T9()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF7E9552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_T10()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF569552DCF", 40), 40);
  DATA.INFO_tag = 0;
  DATA.FCS = M_FCS_Tramt;
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_T11_0()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  bitcpy(DATA.FCS, 5, cvttobit("0xF7BDEF7BDE", 40), 40);
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_R11_0()
{
  DATA.PA = 0;
  DATA.PA_tag = 6;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = IUT_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  bitcpy(DATA.FCS, 5, cvttobit("0xF7BDEF7BDE", 40), 40);
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x2d6a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_T11_1()
{
  bitcpy(DATA.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA.PA_tag = 0;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = Tester_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  bitcpy(DATA.FCS, 5, cvttobit("0xF7BDEF7BDE", 40), 40);
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x294a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA       *
DATA_R11_1()
{
  DATA.PA = 0;
  DATA.PA_tag = 6;
  DATA.SD = 0x311;
  DATA.SD_tag = 0;
  DATA.FC = 0xc1;
  DATA.FC_tag = 0;
  DATA.DA = Tester_Address;
  DATA.DA_tag = 0;
  DATA.SA = Tester_Address;
  DATA.SA_tag = 0;
  bitcpy(DATA.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA.INFO_tag = 0;
  bitcpy(DATA.FCS, 5, cvttobit("0xF7BDEF7BDE", 40), 40);
  DATA.FCS_tag = 0;
  DATA.ED = 0x15;
  DATA.ED_tag = 0;
  DATA.FS = 0x2d4a;
  DATA.FS_tag = 0;
  return (&DATA);
}
pdu_DATA_InvLen *
DATA_InvLen_T1()
{
  DATA_InvLen.PA = P_16;
  DATA_InvLen.PA_tag = 0;
  DATA_InvLen.SD = 0x311;
  DATA_InvLen.SD_tag = 0;
  DATA_InvLen.FC = 0xc1;
  DATA_InvLen.FC_tag = 0;
  DATA_InvLen.DA = IUT_Address;
  DATA_InvLen.DA_tag = 0;
  DATA_InvLen.SA = Tester_Address;
  DATA_InvLen.SA_tag = 0;
  bitcpy(DATA_InvLen.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA_InvLen.INFO_tag = 0;
  DATA_InvLen.FCS = M_FCS_Tramt;
  DATA_InvLen.FCS_tag = 0;
  DATA_InvLen.ED = 0x15;
  DATA_InvLen.ED_tag = 0;
  DATA_InvLen.FS = 0x294a;
  DATA_InvLen.FS_tag = 0;
  return (&DATA_InvLen);
}
pdu_DATA_OverMaxLen *
DATA_OverMaxLen_T1()
{
  bitcpy(DATA_OverMaxLen.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  DATA_OverMaxLen.PA_tag = 0;
  DATA_OverMaxLen.SD = 0x311;
  DATA_OverMaxLen.SD_tag = 0;
  DATA_OverMaxLen.FC = 0xc1;
  DATA_OverMaxLen.FC_tag = 0;
  DATA_OverMaxLen.DA = IUT_Address;
  DATA_OverMaxLen.DA_tag = 0;
  DATA_OverMaxLen.SA = Tester_Address;
  DATA_OverMaxLen.SA_tag = 0;
  bitcpy(DATA_OverMaxLen.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA_OverMaxLen.INFO_tag = 0;
  DATA_OverMaxLen.FCS = M_FCS_Tramt;
  DATA_OverMaxLen.FCS_tag = 0;
  DATA_OverMaxLen.ED = 0x15;
  DATA_OverMaxLen.ED_tag = 0;
  DATA_OverMaxLen.FS = 0x294a;
  DATA_OverMaxLen.FS_tag = 0;
  return (&DATA_OverMaxLen);
}
pdu_TOKEN     *
TOKEN_T1()
{
  bitcpy(TOKEN.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  TOKEN.PA_tag = 0;
  TOKEN.SD = 0x311;
  TOKEN.SD_tag = 0;
  TOKEN.FC = 0x80;
  TOKEN.FC_tag = 0;
  TOKEN.ED = 0x2b5;
  TOKEN.ED_tag = 0;
  return (&TOKEN);
}
pdu_TOKEN     *
TOKEN_R1()
{
  TOKEN.PA = 0;
  TOKEN.PA_tag = 6;
  TOKEN.SD = 0x311;
  TOKEN.SD_tag = 0;
  TOKEN.FC = 0x80;
  TOKEN.FC_tag = 0;
  TOKEN.ED = 0x2b5;
  TOKEN.ED_tag = 0;
  return (&TOKEN);
}
pdu_TOKEN      *
TOKEN_T2()
{
  bitcpy(TOKEN.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  TOKEN.PA_tag = 0;
  TOKEN.SD = 0x311;
  TOKEN.SD_tag = 0;
  TOKEN.FC = 0x80;
  TOKEN.FC_tag = 0;
  TOKEN.ED = 0x015;
  TOKEN.ED_tag = 0;
  return (&TOKEN);
}
pdu_TOKEN      *
TOKEN_T3()
{
  bitcpy(TOKEN.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  TOKEN.PA_tag = 0;
  TOKEN.SD = 0x311;
  TOKEN.SD_tag = 0;
  TOKEN.FC = 0x80;
  TOKEN.FC_tag = 0;
  TOKEN.ED = 0x3f5;
  TOKEN.ED_tag = 0;
  return (&TOKEN);
}
pdu_TOKEN      *
TOKEN_T4()
{
  bitcpy(TOKEN.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  TOKEN.PA_tag = 0;
  TOKEN.SD = 0x311;
  TOKEN.SD_tag = 0;
  TOKEN.FC = 0x80;
  TOKEN.FC_tag = 0;
  TOKEN.ED = 0x3d5;
  TOKEN.ED_tag = 0;
  return (&TOKEN);
}
pdu_CLAIM      *
CLAIM_T1()
{
  bitcpy(CLAIM.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  CLAIM.PA_tag = 0;
  CLAIM.SD = 0x311;
  CLAIM.SD_tag = 0;
  CLAIM.FC = 0xc3;
  CLAIM.FC_tag = 0;
  CLAIM.DA = IUT_Address;
  CLAIM.DA_tag = 0;
  CLAIM.SA = IUT_Address;
  CLAIM.SA_tag = 0;
  CLAIM.INFO = T_Req_IUT;
  CLAIM.INFO_tag = 0;
  CLAIM.FCS = M_FCS_Tramt;
  CLAIM.FCS_tag = 0;
  CLAIM.ED = 0x15;
  CLAIM.ED_tag = 0;
  CLAIM.FS = 0x294a;
  CLAIM.FS_tag = 0;
  return (&CLAIM);
}
pdu_CLAIM      *
CLAIM_R1()
```

```
{
  CLAIM.PA = 0;
  CLAIM.PA_tag = 6;
  CLAIM.SD = 0x311;
  CLAIM.SD_tag = 0;
  CLAIM.FC = 0xc3;
  CLAIM.FC_tag = 0;
  CLAIM.DA = IUT_Address;
  CLAIM.DA_tag = 0;
  CLAIM.SA = IUT_Address;
  CLAIM.SA_tag = 0;
  CLAIM.INFO = T_Req_IUT;
  CLAIM.INFO_tag = 0;
  CLAIM.FCS = M_FCS_Rev;
  CLAIM.FCS_tag = 0;
  CLAIM.ED = 0x15;
  CLAIM.ED_tag = 0;
  CLAIM.FS = 0x294a;
  CLAIM.FS_tag = 0;
  return (&CLAIM);
}
pdu_CLAIM      *
CLAIM_T2()
{
  bitcpy(CLAIM.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  CLAIM.PA_tag = 0;
  CLAIM.SD = 0x311;
  CLAIM.SD_tag = 0;
  CLAIM.FC = 0xc3;
  CLAIM.FC_tag = 0;
  CLAIM.DA = Tester_Address;
  CLAIM.DA_tag = 0;
  CLAIM.SA = Tester_Address;
  CLAIM.SA_tag = 0;
  CLAIM.INFO = T_Bid_Max;
  CLAIM.INFO_tag = 0;
  CLAIM.FCS = M_FCS_Tramt;
  CLAIM.FCS_tag = 0;
  CLAIM.ED = 0x15;
  CLAIM.ED_tag = 0;
  CLAIM.FS = 0x294a;
  CLAIM.FS_tag = 0;
  return (&CLAIM);
}
pdu_CLAIM      *
CLAIM_R2()
{
  CLAIM.PA = 0;
  CLAIM.PA_tag = 6;
  CLAIM.SD = 0x311;
  CLAIM.SD_tag = 0;
  CLAIM.FC = 0xc3;
  CLAIM.FC_tag = 0;
  CLAIM.DA = Tester_Address;
  CLAIM.DA_tag = 0;
  CLAIM.SA = Tester_Address;
  CLAIM.SA_tag = 0;
  CLAIM.INFO = T_Bid_Max;
  CLAIM.INFO_tag = 0;
  CLAIM.FCS = M_FCS_Rev;
  CLAIM.FCS_tag = 0;
  CLAIM.ED = 0x15;
  CLAIM.ED_tag = 0;
  CLAIM.FS = 0x294a;
  CLAIM.FS_tag = 0;
  return (&CLAIM);
```

```
}
pdu_CLAIM      *
CLAIM_T3()
{
  bitcpy(CLAIM.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  CLAIM.PA_tag = 0;
  CLAIM.SD = 0x311;
  CLAIM.SD_tag = 0;
  CLAIM.FC = 0xc3;
  CLAIM.FC_tag = 0;
  CLAIM.DA = Tester_Address;
  CLAIM.DA_tag = 0;
  CLAIM.SA = Tester_Address;
  CLAIM.SA_tag = 0;
  CLAIM.INFO = T_Bid_Min;
  CLAIM.INFO_tag = 0;
  CLAIM.FCS = M_FCS_Tramt;
  CLAIM.FCS_tag = 0;
  CLAIM.ED = 0x15;
  CLAIM.ED_tag = 0;
  CLAIM.FS = 0x294a;
  CLAIM.FS_tag = 0;
  return (&CLAIM);
}
pdu_CLAIM      *
CLAIM_T4()
{
  bitcpy(CLAIM.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  CLAIM.PA_tag = 0;
  CLAIM.SD = 0x311;
  CLAIM.SD_tag = 0;
  CLAIM.FC = 0xc3;
  CLAIM.FC_tag = 0;
  CLAIM.DA = Tester_Address;
  CLAIM.DA_tag = 0;
  CLAIM.SA = Tester_Address;
  CLAIM.SA_tag = 0;
  CLAIM.INFO = T_Req_Tester;
  CLAIM.INFO_tag = 0;
  CLAIM.FCS = M_FCS_Tramt;
  CLAIM.FCS_tag = 0;
  CLAIM.ED = 0x15;
  CLAIM.ED_tag = 0;
  CLAIM.FS = 0x294a;
  CLAIM.FS_tag = 0;
  return (&CLAIM);
}
pdu_CLAIM      *
CLAIM_R4()
{
  CLAIM.PA = 0;
  CLAIM.PA_tag = 6;
  CLAIM.SD = 0x311;
  CLAIM.SD_tag = 0;
  CLAIM.FC = 0xc3;
  CLAIM.FC_tag = 0;
  CLAIM.DA = Tester_Address;
  CLAIM.DA_tag = 0;
  CLAIM.SA = Tester_Address;
  CLAIM.SA_tag = 0;
  CLAIM.INFO = T_Req_Tester;
  CLAIM.INFO_tag = 0;
  CLAIM.FCS = M_FCS_Rev;
  CLAIM.FCS_tag = 0;
  CLAIM.ED = 0x15;
  CLAIM.ED_tag = 0;
  CLAIM.FS = 0x294a;
  CLAIM.FS_tag = 0;
  return (&CLAIM);
}
pdu_BEACON     *
BEACON_T1()
{
  bitcpy(BEACON.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  BEACON.PA_tag = 0;
  BEACON.SD = 0x311;
  BEACON.SD_tag = 0;
  BEACON.FC = 0xc2;
  BEACON.FC_tag = 0;
  bitcpy(BEACON.DA, 5, cvttobit("0x000000000000", 48), 48);
  BEACON.DA_tag = 0;
  BEACON.SA = IUT_Address;
  BEACON.SA_tag = 0;
  BEACON.INFO = 0x00;
  BEACON.INFO_tag = 0;
  BEACON.FCS = M_FCS_Tramt;
  BEACON.FCS_tag = 0;
  BEACON.ED = 0x15;
  BEACON.ED_tag = 0;
  BEACON.FS = 0x294a;
  BEACON.FS_tag = 0;
  return (&BEACON);
}
pdu_BEACON     *
BEACON_R1()
{
  BEACON.PA = 0;
  BEACON.PA_tag = 6;
  BEACON.SD = 0x311;
  BEACON.SD_tag = 0;
  BEACON.FC = 0xc2;
  BEACON.FC_tag = 0;
  bitcpy(BEACON.DA, 5, cvttobit("0x000000000000", 48), 48);
  BEACON.DA_tag = 0;
  BEACON.SA = IUT_Address;
  BEACON.SA_tag = 0;
  BEACON.INFO = 0x00;
  BEACON.INFO_tag = 0;
  BEACON.FCS = M_FCS_Rev;
  BEACON.FCS_tag = 0;
  BEACON.ED = 0x15;
  BEACON.ED_tag = 0;
  BEACON.FS = 0x294a;
  BEACON.FS_tag = 0;
  return (&BEACON);
}
pdu_BEACON     *
BEACON_T2()
{
  bitcpy(BEACON.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  BEACON.PA_tag = 0;
  BEACON.SD = 0x311;
  BEACON.SD_tag = 0;
  BEACON.FC = 0xc2;
  BEACON.FC_tag = 0;
  bitcpy(BEACON.DA, 5, cvttobit("0x000000000000", 48), 48);
  BEACON.DA_tag = 0;
  BEACON.SA = Tester_Address;
  BEACON.SA_tag = 0;
  BEACON.INFO = 0x00;
  BEACON.INFO_tag = 0;
  BEACON.FCS = M_FCS_Tramt;
```

```
    BEACON.FCS_tag = 0;
    BEACON.ED = 0x15;
    BEACON.ED_tag = 0;
    BEACON.FS = 0x294a;
    BEACON.FS_tag = 0;
    return (&BEACON);
  }
  pdu_BEACON     *
  BEACON_R2()
  {
    BEACON.PA = 0;
    BEACON.PA_tag = 6;
    BEACON.SD = 0x311;
    BEACON.SD_tag = 0;
    BEACON.FC = 0xc2;
    BEACON.FC_tag = 0;
    bitcpy(BEACON.DA, 5, cvttobit("0x000000000000", 48), 48);
    BEACON.DA_tag = 0;
    BEACON.SA = Tester_Address;
    BEACON.SA_tag = 0;
    BEACON.INFO = 0x00;
    BEACON.INFO_tag = 0;
    BEACON.FCS = M_FCS_Rev;
    BEACON.FCS_tag = 0;
    BEACON.ED = 0x15;
    BEACON.ED_tag = 0;
    BEACON.FS = 0x294a;
    BEACON.FS_tag = 0;
    return (&BEACON);
  }
  pdu_ECHO_Req   *
  ECHO_Req_T1()
  {
    bitcpy(ECHO_Req.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
    ECHO_Req.PA_tag = 0;
    ECHO_Req.SD = 0x311;
    ECHO_Req.SD_tag = 0;
    ECHO_Req.FC = 0x81;
    ECHO_Req.FC_tag = 0;
    bitcpy(ECHO_Req.DA, 48, IUT_Address, 32);
    ECHO_Req.DA_tag = 0;
    bitcpy(ECHO_Req.SA, 48, Tester_Address, 32);
    ECHO_Req.SA_tag = 0;
    ECHO_Req.Frame_Class = 0x04;
    ECHO_Req.Frame_Class_tag = 0;
    ECHO_Req.Frame_Type = 0x02;
    ECHO_Req.Frame_Type_tag = 0;
    ECHO_Req.Version_ID = 0x0000;
    ECHO_Req.Version_ID_tag = 0;
    ECHO_Req.Transaction_ID = 0x00000000;
    ECHO_Req.Transaction_ID_tag = 0;
    bitcpy(ECHO_Req.Station_ID, 64, cvttobit("0x0000000000000001", 64), 64);
    ECHO_Req.Station_ID_tag = 0;
    ECHO_Req.Pad = 0x0000;
    ECHO_Req.Pad_tag = 0;
    ECHO_Req.InfoField_Length = 0x116A;
    ECHO_Req.InfoField_Length_tag = 0;
    ECHO_Req.Parameter_Type = 0x0011;
    ECHO_Req.Parameter_Type_tag = 0;
    ECHO_Req.Parameter_Length = 0x1168;
    ECHO_Req.Parameter_Length_tag = 0;
    bitcpy(ECHO_Req.Echo_data, 9344, cvttobit("0xF269552DCF", 40), 40);
    ECHO_Req.Echo_data_tag = 0;
    ECHO_Req.FCS = M_FCS_Tramt;
    ECHO_Req.FCS_tag = 0;
    ECHO_Req.ED = 0x15;
  ECHO_Req.ED_tag = 0;
  ECHO_Req.FS = 0x294a;
  ECHO_Req.FS_tag = 0;
  return (&ECHO_Req);
}
pdu_ECHO_Resp  *
ECHO_Resp_R1()
{
  ECHO_Resp.PA = 0;
  ECHO_Resp.PA_tag = 6;
  ECHO_Resp.SD = 0x311;
  ECHO_Resp.SD_tag = 0;
  ECHO_Resp.FC = 0x81;
  ECHO_Resp.FC_tag = 0;
  bitcpy(ECHO_Resp.DA, 48, Tester_Address, 32);
  ECHO_Resp.DA_tag = 0;
  bitcpy(ECHO_Resp.SA, 48, IUT_Address, 32);
  ECHO_Resp.SA_tag = 0;
  ECHO_Resp.Frame_Class = 0x04;
  ECHO_Resp.Frame_Class_tag = 0;
  ECHO_Resp.Frame_Type = 0x03;
  ECHO_Resp.Frame_Type_tag = 0;
  ECHO_Resp.Version_ID = 0x0000;
  ECHO_Resp.Version_ID_tag = 0;
  ECHO_Resp.Transaction_ID = 0x00000000;
  ECHO_Resp.Transaction_ID_tag = 0;
  bitcpy(ECHO_Resp.Station_ID, 64, cvttobit("0x0000000000000001", 64), 64);
  ECHO_Resp.Station_ID_tag = 0;
  ECHO_Resp.Pad = 0x0000;
  ECHO_Resp.Pad_tag = 0;
  ECHO_Resp.InfoField_Length = 0x116A;
  ECHO_Resp.InfoField_Length_tag = 0;
  ECHO_Resp.Parameter_Type = 0x0011;
  ECHO_Resp.Parameter_Type_tag = 0;
  ECHO_Resp.Parameter_Length = 0x1168;
  ECHO_Resp.Parameter_Length_tag = 0;
  bitcpy(ECHO_Resp.Echo_data, 9344, cvttobit("0xF269552DCF", 40), 40);
  ECHO_Resp.Echo_data_tag = 0;
  ECHO_Resp.FCS = M_FCS_Rev;
  ECHO_Resp.FCS_tag = 0;
  ECHO_Resp.ED = 0x15;
  ECHO_Resp.ED_tag = 0;
  ECHO_Resp.FS = 0x294a;
  ECHO_Resp.FS_tag = 0;
  return (&ECHO_Resp);
}
pdu_ECHO_Resp  *
ECHO_Resp_T1()
{
  bitcpy(ECHO_Resp.PA, 5, cvttobit("0xFFFFFFFFFFFFFFFFFFFF", 80), 80);
  ECHO_Resp.PA_tag = 0;
  ECHO_Resp.SD = 0x311;
  ECHO_Resp.SD_tag = 0;
  ECHO_Resp.FC = 0x81;
  ECHO_Resp.FC_tag = 0;
  bitcpy(ECHO_Resp.DA, 48, Tester_Address, 32);
  ECHO_Resp.DA_tag = 0;
  bitcpy(ECHO_Resp.SA, 48, IUT_Address, 32);
  ECHO_Resp.SA_tag = 0;
  ECHO_Resp.Frame_Class = 0x04;
  ECHO_Resp.Frame_Class_tag = 0;
  ECHO_Resp.Frame_Type = 0x03;
  ECHO_Resp.Frame_Type_tag = 0;
  ECHO_Resp.Version_ID = 0x0000;
  ECHO_Resp.Version_ID_tag = 0;
  ECHO_Resp.Transaction_ID = 0x00000000;
  ECHO_Resp.Transaction_ID_tag = 0;
  bitcpy(ECHO_Resp.Station_ID, 64, cvttobit("0x0000000000000001", 64), 64);
  ECHO_Resp.Station_ID_tag = 0;
  ECHO_Resp.Pad = 0x0000;
  ECHO_Resp.Pad_tag = 0;
  ECHO_Resp.InfoField_Length = 0x116A;
  ECHO_Resp.InfoField_Length_tag = 0;
  ECHO_Resp.Parameter_Type = 0x0011;
  ECHO_Resp.Parameter_Type_tag = 0;
  ECHO_Resp.Parameter_Length = 0x1168;
  ECHO_Resp.Parameter_Length_tag = 0;
  bitcpy(ECHO_Resp.Echo_data, 9344, cvttobit("0xF269552DCF", 40), 40);
  ECHO_Resp.Echo_data_tag = 0;
  ECHO_Resp.FCS = M_FCS_Tramt;
  ECHO_Resp.FCS_tag = 0;
  ECHO_Resp.ED = 0x15;
  ECHO_Resp.ED_tag = 0;
  ECHO_Resp.FS = 0x296b;
  ECHO_Resp.FS_tag = 0;
  return (&ECHO_Resp);
}
pdu_DATA_Strip *
DATA_Strip_R1()
{
  DATA_Strip.PA = 0;
  DATA_Strip.PA_tag = 6;
  DATA_Strip.SD = 0;
  DATA_Strip.SD_tag = 8;
  DATA_Strip.FC = 0;
  DATA_Strip.FC_tag = 8;
  DATA_Strip.DA = 0;
  DATA_Strip.DA_tag = 8;
  DATA_Strip.SA = 0;
  DATA_Strip.SA_tag = 8;
  DATA_Strip.INFO = 0;
  DATA_Strip.INFO_tag = 8;
  DATA_Strip.FCS = 0;
  DATA_Strip.FCS_tag = 8;
  DATA_Strip.ED = 0;
  DATA_Strip.ED_tag = 8;
  DATA_Strip.FS = 0;
  DATA_Strip.FS_tag = 8;
  return (&DATA_Strip);
}
pdu_DATA_Strip *
DATA_Strip_R2()
{
  DATA_Strip.PA = 0;
  DATA_Strip.PA_tag = 6;
  DATA_Strip.SD = 0x311;
  DATA_Strip.SD_tag = 0;
  DATA_Strip.FC = 0xc1;
  DATA_Strip.FC_tag = 0;
  DATA_Strip.DA = IUT_Address;
  DATA_Strip.DA_tag = 0;
  DATA_Strip.SA = Tester_Address;
  DATA_Strip.SA_tag = 0;
  DATA_Strip.INFO = 0;
  DATA_Strip.INFO_tag = 8;
  DATA_Strip.FCS = 0;
  DATA_Strip.FCS_tag = 8;
  DATA_Strip.ED = 0;
  DATA_Strip.ED_tag = 8;
  DATA_Strip.FS = 0;
  DATA_Strip.FS_tag = 8;
  return (&DATA_Strip);
```

```
}
pdu_DATA_Strip *
DATA_Strip_R3()
{
  DATA_Strip.PA = 0;
  DATA_Strip.PA_tag = 6;
  DATA_Strip.SD = 0x311;
  DATA_Strip.SD_tag = 0;
  DATA_Strip.FC = 0xc1;
  DATA_Strip.FC_tag = 0;
  DATA_Strip.DA = IUT_Address;
  DATA_Strip.DA_tag = 0;
  DATA_Strip.SA = Tester_Address;
  DATA_Strip.SA_tag = 0;
  bitcpy(DATA_Strip.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA_Strip.INFO_tag = 0;
  DATA_Strip.FCS = M_FCS_Rev;
  DATA_Strip.FCS_tag = 0;
  DATA_Strip.ED = 0x15;
  DATA_Strip.ED_tag = 0;
  DATA_Strip.FS = 0x2d6a;
  DATA_Strip.FS_tag = 0;
  return (&DATA_Strip);
}
pdu_DATA_Strip *
DATA_Strip_R4()
{
  DATA_Strip.PA = 0;
  DATA_Strip.PA_tag = 6;
  DATA_Strip.SD = 0x311;
  DATA_Strip.SD_tag = 0;
  DATA_Strip.FC = 0xc1;
  DATA_Strip.FC_tag = 0;
  DATA_Strip.DA = IUT_Address;
  DATA_Strip.DA_tag = 0;
  DATA_Strip.SA = Tester_Address;
  DATA_Strip.SA_tag = 0;
  bitcpy(DATA_Strip.INFO, 5, cvttobit("0xF269552DCF", 40), 40);
  DATA_Strip.INFO_tag = 0;
  DATA_Strip.FCS = M_FCS_Rev;
  DATA_Strip.FCS_tag = 0;
  DATA_Strip.ED = 0x15;
  DATA_Strip.ED_tag = 0;
  DATA_Strip.FS = 0x2d6b;
  DATA_Strip.FS_tag = 0;
  return (&DATA_Strip);
}
pdu_DATA_Strip *
DATA_Strip_R5()
{
  DATA_Strip.PA = 0;
  DATA_Strip.PA_tag = 6;
  DATA_Strip.SD = 0x311;
  DATA_Strip.SD_tag = 0;
  DATA_Strip.FC = 0;
  DATA_Strip.FC_tag = 8;
  DATA_Strip.DA = 0;
  DATA_Strip.DA_tag = 8;
  DATA_Strip.SA = 0;
  DATA_Strip.SA_tag = 8;
  DATA_Strip.INFO = 0;
  DATA_Strip.INFO_tag = 8;
  DATA_Strip.FCS = 0;
  DATA_Strip.FCS_tag = 8;
  DATA_Strip.ED = 0;
  DATA_Strip.ED_tag = 8;
  DATA_Strip.FS = 0;
  DATA_Strip.FS_tag = 8;
  return (&DATA_Strip);
}
pdu_DATA_Strip *
DATA_Strip_R6()
{
  DATA_Strip.PA = 0;
  DATA_Strip.PA_tag = 6;
  DATA_Strip.SD = 0x311;
  DATA_Strip.SD_tag = 0;
  DATA_Strip.FC = 0xc1;
  DATA_Strip.FC_tag = 0;
  DATA_Strip.DA = Tester_Address;
  DATA_Strip.DA_tag = 0;
  DATA_Strip.SA = IUT_Address;
  DATA_Strip.SA_tag = 0;
  DATA_Strip.INFO = 0;
  DATA_Strip.INFO_tag = 8;
  DATA_Strip.FCS = 0;
  DATA_Strip.FCS_tag = 8;
  DATA_Strip.ED = 0;
  DATA_Strip.ED_tag = 8;
  DATA_Strip.FS = 0;
  DATA_Strip.FS_tag = 8;
  return (&DATA_Strip);
}
pdu_TOKEN_Strip *
TOKEN_Strip_R1()
{
  TOKEN_Strip.PA = 0;
  TOKEN_Strip.PA_tag = 6;
  TOKEN_Strip.SD = 0x311;
  TOKEN_Strip.SD_tag = 0;
  TOKEN_Strip.FC = 0x80;
  TOKEN_Strip.FC_tag = 0;
  TOKEN_Strip.ED = 0;
  TOKEN_Strip.ED_tag = 8;
  return (&TOKEN_Strip);
}
pdu_IDLE        *
IDLE_I()
{
  IDLE.SingleIdle = 0x1f;
  IDLE.SingleIdle_tag = 0;
  return (&IDLE);
}
```

# Appendix D

## FDDI MAC Conformance Test
## Host Specific Library Routines

```
#include <stdio.h>
#include <ctype.h>
#include <sys/time.h>
#include <sys/signal.h>
#include "sysdef.h"

struct itimerval ttime;
long            tdur;
int             _timeout;
extern int     *pint;
extern char    *flagptr;
extern FILE    *fp;

flush_queue()
{
  pint = (int *) flagptr;
  *pint = 256;
  while (*pint == 256);
}


Start_timer(tname, dur)
  char            *tname;
  long             dur;
{
  extern           wakeup();

  printf("Start timer %s\n", tname);
  fprintf(fp, "Start timer %s\n", tname);
  tdur = dur;
  _timeout = FALSE;
  signal(SIGALRM, wakeup);
  ualarm(tdur * 1000);
}

Timeout(oname)
  char            *oname;
{
  return _timeout;
}

Read_Timer(rname)
  char            *rname;
{
  printf("Read timer %s.\n", rname);
  fprintf(fp, "Read timer %s.\n", rname);
  getitimer(ITIMER_REAL, &ttime);
  return (tdur - ((ttime.it_value.tv_sec + ttime.it_value.tv_usec / 1000000) *
                   1000));
}

Cancel_Timer()
{
  printf("Cancel timer\n");
  fprintf(fp, "Cancel timer\n");
  ualarm(0);
}

wakeup()
{
  _timeout = TRUE;
}

bitcmp(str1, sz1, str2, sz2)
  char            *str1, *str2;
  int              sz1, sz2;
```

```
{
  int              i, j, fs, ss, idnum;
  char            *fst, *sst;

  if (sz1 < sz2) {
    fs = (sz1 + 7) / 8;
    ss = (sz2 + 7) / 8;
    idnum = 1;
    fst = str1;
    sst = str2;
  } else {
    fs = (sz2 + 7) / 8;
    ss = (sz1 + 7) / 8;
    idnum = -1;
    fst = str2;
    sst = str1;
  }
  for (i = 0; i < ss - fs; i++)
    if (sst[i])
      return PM((0 - sst[i]) * idnum);
  for (j = i; j < ss; j++)
    if (fst[j - i] != sst[j])
      return PM((fst[j - i] - sst[j]) * idnum);
  return 0;
}

bitcpy(str1, size1, str2, size2)
  char            *str1, *str2;
  int              size1, size2;
{
  int              i, j, result;
  int              tmp1, tmp2;

  tmp1 = (size1 + 7) / 8;
  tmp2 = (size2 + 7) / 8;
  if (tmp1 < tmp2)
    result = tmp1;
  else
    result = tmp2;
  for (i = 0; i < result; i++)
    str1[tmp1 - i - 1] = str2[tmp2 - i - 1];
  for (j = result; j < tmp1; j++)
    str1[tmp1 - j - 1] = '\0';
}

unsigned char  *
cvttobit(hexnum, hexsize)
  char            *hexnum;
  int              hexsize;
{
  unsigned char   globhex[80];
  int              tmpint, tmpsize;

  if (strncmp(hexnum, "0x", 2)) {
    tmpsize = (hexsize + 7) / 8;
    sscanf(hexnum, "%d", &tmpint);
    sprintf(hexnum, "%0*x", tmpsize, tmpint);
    cnvrtfromhex(hexnum, globhex, hexsize);
  } else
    cnvrtfromhex(hexnum + 2, globhex, hexsize);
  return globhex;
}

cnvrtfromhex(snum, hexchar, size)
  char            snum[];
unsigned char   hexchar[];
```

```
int               size;
{
  int               i, j, k, dgts, nibl[2];

  dgts = (size + 7) / 8;
  for (i = dgts - 1, j = strlen(snum) - 1; (i >= 0) && (j >= 0); i--, j = j - 2) {
    for (k = 0; k < 2; k++) {
      if ((k == 1) && (j - k < 0)) {
        nibl[1] = 0;
        break;
      }
      if ((snum[j - k] >= '0') && (snum[j - k] <= '9'))
        nibl[k] = snum[j - k] - '0';
      else {
        if (islower(snum[j - k]))
          snum[j - k] = snum[j - k] - 'a' + 'A';
        if ((snum[j - k] >= 'A') && (snum[j - k] <= 'F'))
          nibl[k] = snum[j - k] - 'A' + 10;
        else {
          printf("Bad number. Try again.\n");
          scanf("%s", snum);
          i = dgts;
          j = strlen(snum) + 1;
          k = 2;
        }
      }
    }
    hexchar[i] = nibl[0] + 16 * nibl[1];
  }
  for (j = i; j >= 0; j--)
    hexchar[j] = 0;
}

SetpVerdict(pverd)
  _verdict          pverd;
{
  if ((Result == NONE) || (Result == PASS))
    Result = pverd;
  else if ((pverd == FAIL) || (pverd == INCONC))
    Result = pverd;
  else
    printf("Verdict not accpetable\n");
  fprintf(fp, "Verdict not accpetable\n");
  switch (Result) {
  case NONE:
    strcpy(R, "NONE");
    break;
  case FAIL:
    strcpy(R, "FAIL");
    break;
  case PASS:
    strcpy(R, "PASS");
    break;
  case INCONC:
    strcpy(R, "INCONCLUSIVE");
    break;
  }
}

SetfVerdict(fverd)
  _verdict          fverd;
{
  if (fverd != RESULT)
    if ((Result == NONE) || (Result == PASS))
      Result = fverd;
    else if (fverd == FAIL)
```

```
      Result = fverd;
    else if ((Result == INCONC) && (fverd == INCONC));
    else {
      printf("Illegal verdict\n");
      fprintf(fp, "Illegal verdict\n");
    }
  else if (Result == NONE) {
    printf("Illegal verdict\n");
    fprintf(fp, "Illegal verdict\n");
  }
}

/*
 * print_verdict -- output a message and the final verdict on the test log.
 *
 */

print_verdict(message)
  char           *message;
{
  printf("%s -- ", message);
  fprintf(fp, "%s -- ", message);
  switch (Result) {
  case NONE:
    printf("No verdict.\n");
    fprintf(fp, "No verdict.\n");
    break;
  case FAIL:
    printf("Test Failed.\n");
    fprintf(fp, "Test Failed.\n");
    break;
  case PASS:
    printf("Test Passed.\n");
    fprintf(fp, "Test Passed.\n");
    break;
  case INCONC:
    printf("Verdict is inconclusive.\n");
    fprintf(fp, "Verdict is inconclusive.\n");
    break;
  }
}

/*
 * Implicit_send -- output a message to operator console instructing the
 * tests operator to initiate manual operations on the IUT.
 */

Implicit_send(message)
  char           *message;
{
  printf("Test coordination, force IUT to send %s\n", message);
  fprintf(fp, "Test coordination, force IUT to send %s\n", message);
}

/*
 * Output_trace -- Print a message on the test trace log.
 *
 */

Output_trace(message)
  char           *message;
{
  printf("%s\n", message);      /* output to console */
  fprintf(fp, "%s\n", message); /* output to file */
}
```

| | |
|---|---|
| NIST-114A (REV. 3-89) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY **BIBLIOGRAPHIC DATA SHEET** | 1. PUBLICATION OR REPORT NUMBER NISTIR 90-4267 |
| | 2. PERFORMING ORGANIZATION REPORT NUMBER |
| | 3. PUBLICATION DATE JULY 1990 |

4. TITLE AND SUBTITLE

A Conformance Test for FDDI Medium Access Control (MAC)

5. AUTHOR(S)

Zuqiu Liu and William E. Burr

| | |
|---|---|
| 6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY GAITHERSBURG, MD 20899 | 7. CONTRACT/GRANT NUMBER |
| | 8. TYPE OF REPORT AND PERIOD COVERED |

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

☐ DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The Fiber Distributed Data Interface (FDDI) is an emerging standard for a 100 MBit/s fiber optic token ring Local Area Network. The FDDI Medium Access Control (MAC) data link layer protocol standard specifies a complex protocol which controls the normal operation of the FDDI network. This report contains a proposed test for the conformance of implementations of FDDI MAC. The tests are written in the Combined Tree and Tabular Notation (TTCN) and automatically converted to a C language program by a TTCN to C translator.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

conformance test; data link layer protocol; FDDI; fiber optic network; MAC; medium access control; token ring network; TTCN

| | |
|---|---|
| 13. AVAILABILITY | 14. NUMBER OF PRINTED PAGES |
| [✓ X] UNLIMITED | 187 |
| ☐ FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). | |
| ☐ ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402. | 15. PRICE A09 |
| [✓ X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161. | |

ELECTRONIC FORM